ГИС Аксиома API для Python

Версия 4.4

ООО ЭСТИ

10 февраля, 2023

Оглавление

1 О компоненте

API - программный интерфейс приложения - совокупность классов, процедур и функций, благодаря которым одна компьютерная программа может взаимодействовать с другой программой.

С помощью API для ГИС «Аксиома» на языке Python можно взаимодействовать с ГИС «Аксиома», используя ее функционал для решения задач. Далее понятия «API для Аксиомы.ГИС на языке Python», «Аксиома.ГИС для Python», «Аксиома API» и просто «API» будут относиться к действительно описываемой программной библиотеке.

API использует PySide2, он же Qt for Python. Он идеально подходит для коммерческого использования. Любая ваша разработка с его использованием будет совместима с LGPL и её можно лицензировать на свое усмотрение.

Документация API состоит из двух частей:

- руководство разработчика;
- справочник функций.

Внимание: Нельзя одновременно использовать **axipy** и старое API **axioma**.

Внимание: Нельзя одновременно использовать **PySide2** и **PyQt5**. **axipy** может быть использован только с **PySide2**.

1.1 Как читать эту документацию

Данная документация составлена таким образом, чтобы описать общие принципы и подходы решения тех или иных базовых задач, необходимых при обработке геопространственных данных. Более полное описание всевозможных классов, свойств, методов, исключений, функций и их параметров содержится в справочнике функций.

Документ логически структурирован. Рекомендуется читать его в прямом порядке от начала до конца, чтобы получить общее представление о возможностях, предоставляемых API.

2 Инициализация

API может быть использован следующими способами:

- в приложении ГИС «Аксиома» из панели «Консоль Python»;
- в модуле, который будет загружен в ГИС «Аксиома»;
- как SDK - Software development kit, независимо от приложения Аксиома.ГИС.

Примечание: Для использования API в качестве SDK требуется платная лицензия.

В первых двух случаях можно сразу приступать к работе, так как API будет инициализировано за вас. В последнем случае перед началом использования его необходимо самостоятельно инициализировать. Если забыть это сделать, то при попытке использования будет вызвано исключение, которое напомнит выполнить инициализацию.

Например:

```
from axipy.cs import CoordSystem


try:
    crs = CoordSystem.from_epsg(4326)
except RuntimeError as error:
    print(f"Поймано исключение: {error}")
```

```
>>> Поймано исключение: axipy is not initialized
```

Для инициализации API следует вызвать метод `axipy.init_axioma()`:

```
from axipy import init_axioma
init_axioma()  # инициализация

from axipy.cs import CoordSystem
crs = CoordSystem.from_epsg(4326)
print(crs.name)
```

```
>>> 'Долгота / Широта (WGS 84)'
```

Допускается (но не рекомендуется) импортировать сразу все классы и функции, чтобы упростить процедуру импорта.

```
from axipy import *  # импортируем все типы ГИС "Аксиома"
                     # в текущее пространство имен

crs = CoordSystem.from_epsg(4088)
print(crs.name)
```

```
>>> 'World Equidistant Cylindrical (Sphere)'
```

Приведем пример простейшего приложения, где открывается файл с данными и далее карта показывается в окне просмотра. Пример запускается не из под ГИС Аксиома.

```
import axipy

# Инициализируем ядро
app = axipy.init_axioma()
# Открываем таблицу
table_world = axipy.provider_manager.openfile('world.tab')
# Добавляем ее в каталог
axipy.data_manager.add(table_world)
# Создаем слой
layer_world = axipy.Layer.create(table_world)
# Создаем карту и показываем ее в окне
map_world = axipy.Map([ layer_world ])
mapview = axipy.view_manager.create_mapview(map_world)
# Изменяем размер окна
mapview.position = axipy.Rect(10, 10, 500, 500)
# показываем
mapview.show()
app.exec_()
```

Если эту же задачу необходимо выполнить в рамках запущенной Аксиомы, код будет выглядеть следующим образом:

```
import axipy

table_world = axipy.provider_manager.openfile('world.tab')
layer_world = axipy.Layer.create(table_world)
map_world = axipy.Map([ layer_world ])
mapview = axipy.view_manager.create_mapview(map_world)
mapview.position = axipy.Rect(10, 10, 500, 500)
```

2.1 Системные переменные

Системные переменные, если это требуется, необходимо устанавливать до проведения инициализации ядра.

2.1.1 AXIOMA_LOG_LEVEL

Системная переменная AXIOMA_LOG_LEVEL управляет уровнем логирования в системе. Допустимый диапазон значений (0...3) в сторону уменьшения количества выводимой информации. По умолчанию устанавливается средний уровень - «2». Если установить значение «-1», то логирование будет отключено.

2.1.2 AXIOMA_LANGUAGE

Системная переменная AXIOMA_LANGUAGE устанавливает язык интерфейса. Допустимые значения «ru» и «en».

```
from axipy import init_axioma
import os

os.environ["AXIOMA_LOG_LEVEL"] = "1"
os.environ["AXIOMA_LANGUAGE"] = "en"
init_axioma()  # инициализация
```

2.2 Системные службы

При работе для более удобного доступа к функциям созданы готовые экземпляры (менеджеры). Перечислим их.

Таблица 1: Доступные менеджеры.

Объект	Класс объекта	Описание
axipy.da.data_manager	axipy.da.DataManager	Хранилище объектов данных
axipy.gui.view_manager	axipy.gui.ViewManager	Менеджер содержимого окон
axipy.da.provider_manager	axipy.da.ProviderManager	Открытия/создания объектов данных
axipy.gui.selection_manager	axipy.gui.SelectionManager	Доступ к выделенным объектам
axipy.concurrent.task_manager	axipy.concurrent.TaskManager	Менеджер, запускающий пользовательские задачи
axipy.gui.active_tool_panel	axipy.gui.ActiveToolPanel	Панель активного инструмента

3

Системы Координат

Термины «проекция» и «координатная система» иногда используются один вместо другого, но, на самом деле, понятия, которые они отражают, различны.

Проекция – это уравнения или наборы уравнений, которые содержат математические параметры для карты. Точное число и природа параметров зависят от типа проекции. Проекция – это метод уменьшения искажений карты, вызванных кривизной земной поверхности или, точнее говоря, проекция компенсирует недостатки отображения карты на плоскости в двух измерениях, в то время как координаты существуют в трёх измерениях.

Координатная система – когда параметрам проекции присваиваются определенные значения, они становятся системой координат. Система координат – это набор параметров, описывающих координаты, одна из которых является проекцией.

Системы Координат (СК) представлены типом `axipy.cs.CoordSystem`. Объекты типа `CoordSystem` могут быть созданы, используя:

- код EPSG - European Petroleum Survey Group;
- строку MapInfo PRJ
- строку WKT - Well-known text;
- строку PROJ;
- единиц измерения - для создания СК в план-схеме;

Примечание: Полный список доступных функций создания систем координат содержится в документации к классу `axipy.cs.CoordSystem`.

Список 1: Например

```
merc = CoordSystem.from_epsg(3395)
print(merc.name)
'''
>>> Меркатора WGS84
'''
```

Функция `from_string()` позволяет создавать СК из “универсального представления” - строки с префиксом типа, двоеточием и значением. Возможные префиксы: proj, wkt,

epsg, prj.

Список 2: Например

```
crs = CoordSystem.from_string('prj:Earth Projection 12, 62, "m", 0')
print(crs.name)
'''
>>> Робинсона NAD27
'''
```

3.1 Трансформация координат

Координаты любой точки земной поверхности в разных системах координат будут различаться, переход от одной системы координат к другой осуществляется с помощью специальных формул преобразований и набора параметров,используемых в этих формулах.

Функционал по переходу координат из одной СК в другую выделен в отельный класс `axipy.cs.CoordTransformer`. Он позволяет преобразовывать координаты между двумя СК.

Список 3: Например

```
transformer = CoordTransformer('epsg:4326', 'epsg:26953')
coordinate = (55.76, 37.6)
result = transformer.transform(coordinate)
print(f"Point({result.x}, {result.y})")
'''
>>> Point(8513601.095442554, 9873107.576049749)
'''
```

4 Объекты данных

Разные типы данных обобщены одним абстрактным типом `axipy.da.DataObject`. Он образует иерархию объектов данных различного типа: таблица, растр, грид, чертеж, панорама, и так далее.

5 Провайдеры данных

За каждый тип данных отвечает провайдер данных axipy.da.ProviderManager. Провайдер владеет информацией о том, как представлять конкретный тип объектов данных и как ими манипулировать. Класс axipy.da.ProviderManager содержит все зарегистрированные провайдеры данных.

Каждый провайдер имеет свои особенности и возможности, но общие концепции, такие как открытие и создание, выделены в общий интерфейс axipy.da.DataProvider. В то же время конкретный провайдер может иметь дополнительные функции и параметры, свойственные только ему.

Для получения списка загруженных провайдеров есть функция axipy.da.ProviderManager.loaded_providers().

Например:

```
provider_manager.loaded_providers()
```

```
{'CsvDataProvider': 'Файловый провайдер: Текст с разделителями',
  'DwgDgnFileProvider': 'Провайдер DWG и DGN',
  'GdalDataProvider': 'Растровый провайдер GDAL',
  'MifMidDataProvider': 'Провайдер данных MIF-MID',
  'OgrDataProvider': 'Векторный провайдер OGR',
  'ShapeDataProvider': 'Shapefile',
  'PgDataProvider': 'PostgreSQL',
  'OracleDataProvider': 'oracle',
  'MsSqlDataProvider': 'MS SQL Server',
  'RestDataProvider': 'ArcGIS REST',
  'SqliteDataProvider': 'Векторный провайдер sqlite',
  'TabDataProvider': 'MapInfo',
  'TmsDataProvider': 'Тайловые сервисы',
  'WfsDataProvider': 'Web Feature Service',
  'WmsDataProvider': 'Web Map Service',
  'WmtsDataProvider': 'Web Map Tile Service',
  'XlsDataProvider': 'Провайдер чтения файлов Excel'}
```

Для открытия разных источников данных может потребоваться разная информация. Например, для подключения к базе данных нужно указать имя пользователя и пароль и прочее. Поэтому для большинства провайдеров данных определены функции задания источников данных axipy.da.DataProvider.get_source() с дополнительными

параметрами. Например, axipy.da.CsvDataProvider.get_source(), axipy.da.PostgreDataProvider.get_source() и другие.

Например:

```
source = provider_manager.csv.get_source('path/to/mydata.csv', delimiter=';')
table = source.open()
```

Что эквивалентно:

```
table = provider_manager.csv.open('path/to/mydata.csv', delimiter=';')
```

Или:

```
table = provider_manager.openfile('path/to/mydata.csv', delimiter=';')
```

См.также:

Подробнее в документации axipy.da.ProviderManager.

5.1 Открытие/Создание

Открытие и создание объектов данных axipy.da.DataObject выполняется провайдерами данных. Класс axipy.da.ProviderManager представляет коллекцию зарегистрированных провайдеров данных.

5.1.1 Открытие

Самый простой способ открыть объект данных из файла - использовать функцию axipy.da.ProviderManager.openfile. Функция сама найдет подходящий провайдер данных для открытия.

Совет: Это предпочтительный способ.

Список 1: Пример открытия

```
# filepath = 'path/to/file.tab'
table = provider_manager.openfile(filepath)
```

Примечание: При открытии данных они попадают в единый каталог axipy.da.DataManager.

При необходимости указать больше параметров для открытия, или если параметры по умолчанию не подходят, можно явно использовать провайдер данных. Необходимый провайдер данных можно получить из коллекции зарегистрированных провайдеров данных axipy.da.ProviderManager.

Список 2: Пример открытия конкретным провайдером

```
# csv_filepath = 'path/to/file.csv'
csv_table = provider_manager.csv.open(csv_filepath, delimiter='\t')
```

Самый сложный способ - открытие через словарь dict. Здесь нужно заранее знать все параметры, их допустимые значения и идентификатор провайдера данных.

Список 3: Пример открытия из словаря

```
"""Открывает csv файл через определение ``definition``.
Исключительно в качестве примера. Открывать csv проще провайдером.
"""
# csv_filepath = 'path/to/file.csv'
definition = {
    'src': csv_filepath,
    'delimiter': '\t',
    'charset': 'utf8',
    'hasNamesRow': True,
    'provider': 'CsvDataProvider'
}
csv_table = provider_manager.open(definition)
```

При открытии таблиц из СУБД задаются как параметры подключения, так и требуемая таблица или текст запроса.

Список 4: Пример открытия данный из БД Postgres с указанием имени таблицы.

```
definition = provider_manager.postgre.get_source(host='192.168.0.2', db_name='test',␣
↪user='test', password='pass', dataobject='world')
table = provider_manager.open(definition)
```

Список 5: Пример открытия данный из БД oracle с указанием текста SQL запроса.

```
definition = provider_manager.oracle.get_source(host='192.168.0.2', db_name='ORCL',␣
↪user='test', password='pass', sql='select * from world')
table = provider_manager.open(definition)
```

5.1.2 Создание

Аналогично, самый простой способ создания файлов - с помощью функции axipy.da.ProviderManager.createfile:

Список 6: Пример создания

```
# filepath = 'path/to/file.tab'
schema = Schema(
    Attribute.string('country', 30),
    Attribute.string('capital', 30),
    coordsystem='prj:Earth Projection 1, 104'
)
table = provider_manager.createfile(filepath, schema)
```

Примечание: При создании объекта данных он автоматически открывается.

Для задания дополнительных параметров или использования специализированных возможностей провайдера данных, можно явно вызывать методы конкретного провайдера.

Список 7: Пример создания конкретным провайдером

```
schema = Schema(
    Attribute.string('country', 30),
    Attribute.string('capital', 30),
)
temp_table = provider_manager.shp.open_temporary(schema)
```

Если и этот способ не подходит для решения какой-то задачи, можно создавать объекты данных из словаря. Это самый сложный и непрактичный способ.

5.2 Импорт/Экспорт

5.2.1 Экспорт

Некоторые форматы данных поддерживаются ГИС «Аксиома» только на импорт и/или экспорт. Не для всех форматом можно создать, открывать и редактировать данные, используя транзакционную модель редактирования, являющуюся основной для ГИС «Аксиома». Тем не менее экспорт и создание очень близки по назначению, поэтому они объединены и представлены одним типом `axipy.da.Destination` - назначение объекта данных.

Так для некоторых типов экспорт `axipy.da.Destination.export()` является единственной возможностью вывода, в то время как для других - это дополнительная возможность к имеющейся `axipy.da.Destination.create_open()` в случаях, когда открытие и редактирование не требуется.

Экспортировать можно отдельные записи, таблицы или целые источники данных.

Список 8: Пример экспорта таблицы

```
# Открываем исходную таблицу
table_src = provider_manager.openfile(input_filepath)
# Формируем целевую и производим экспорт
destination = provider_manager.csv.get_destination(output_filepath, Schema())
destination.export_from_table(table_src, copy_schema=True)
```

Если требуется создать таблицу и занести в нее некоторую информацию, но при этом формат данных не поддерживается на изменение, эту проблему можно решить, используя временную таблицу как буфер. Т.е. формируем данные в памяти, а потом производим экспорт этой временной таблицы в нужный нам формат.

Список 9: Пример формирования файла csv на базе временной таблицы

```
# Создаем временную таблицу
definition = {
    'src': '',
    'schema': attr.schema(
        attr.integer('id'),
        attr.string('name')
    )
}
table_src = provider_manager.create(definition)
# Добавляем в нее записи
table_src.insert(Feature({'id': '1', 'name': 'Name1'}))
table_src.insert(Feature({'id': '2', 'name': 'Name2'}))
# Создаем целевую таблицу и производим экспорт
destination = provider_manager.csv.get_destination('outfile.csv', Schema())
destination.export_from_table(table_src, copy_schema=True)
```

5.2.2 Импорт

Источник данных `axipy.da.Source` - это зеркальный тип назначения объекта данных `axipy.da.Destination`. Так же как для назначения, некоторые типы данных поддерживают только импорт, и не могут быть напрямую открыты с помощью `axipy.da.Source.open`. А другие типы поддерживают и открытие и импорт для случаев, когда открытие и редактирование не требуется.

Список 10: Пример экспорта источника

```
source = provider_manager.tab.get_source(input_tabfile)
destination.export_from(source)
```

См.также:

Чтобы узнать, какие типы поддерживают импорт и экспорт, обратитесь к описанию конкретных провайдеров данных `axipy.da.ProviderManager`.

5.3 Таблицы

Для того, чтобы мы могли работать как с географической, так и с атрибутивной информацией, данные в ГИС «Аксиома» организованы в виде групп файлов, имеющих общее имя, но разные расширения.

Пользователь оперирует только с одним файлом из этой группы – так называемым «табличным» файлом, которые имеет расширение TAB. Все файлы из группы автоматически создаются, обновляются и поддерживаются самой программой ГИС «Аксиома». Таблица `axipy.da.Table` является наследником класса объекта данных `axipy.da.DataObject`.

5.3.1 Открытие таблиц

Работа с источником данных начинается с открытия объекта данных с помощью функции `openfile()` объекта `axipy.da.provider_manager`. Для таблиц возвращаемый объект данных будет типа `axipy.da.Table`.

```
table = provider_manager.openfile('../path/to/datadir/worldcap.tab')
```

Некоторые форматы могут содержать несколько таблиц в одном файле, например GeoPackage. В таком случае нужно указать в параметре `dataobject=`, какую таблицу из файла вы хотите открыть.

```
table = provider_manager.openfile('../path/to/datadir/example.gpkg', dataobject='world
↪')
```

У таблицы можно узнать провайдер, которым она была открыта - свойство `axipy.da.DataObject.provider`:

```
print(table.provider)
```

```
>>> 'SqliteDataProvider'
```

5.3.1.1 Источники данных и дополнительные параметры

Источником данных может быть не только файл. Например, это может быть База Данных. Также для открытия или создания некоторых объектов данных может понадобиться указать дополнительные параметры, применимые только для этого типа источника.

Для открытия таких объектов используйте конкретный провайдер данных из коллекции провайдеров `axipy.da.ProviderManager`. Он содержит все методы, параметры и допустимые значения, для работы с конкретным типом объектов данных.

Если по какой-то причине использовать конкретный провайдер данных не удается, то используется функция `open()`, которая принимает словарь со всеми необходимыми параметрами.

Пример открытия таблицы из базы данных PostgreSQL:

Примечание: Исключительно в качестве примера. Намного проще напрямую использовать провайдер данных `axipy.da.PostgreDataProvider`.

```
definition = {
    "src": "<Адрес сервера БД>",
    "port": 5432,
    "db": "<Имя базы данных>",
    "user": "<Имя прользователя>",
    "password": "<Пароль>",
    "dataobject": '"DataAxi"."World"',
    "provider": "PgDataProvider"
}
table = provider_manager.open(definition)
```

5.3.1.2 Открытие источников с множеством таблиц

Для получения всех доступных таблиц одного источника используется функция axipy.da.ProviderManager.read_contents():

```
contents = axipy.provider_manager.read_contents('../path/to/datadir/example.gpkg')
print(contents)
```

```
>>> ['world', 'worldcap']
```

Для источников с единственной таблицей список будет содержать одно имя:

```
contents = axipy.provider_manager.read_contents({'src': '../path/to/datadir/worldcap.
↪tab'})
print(contents)
```

```
>>> ['worldcap']
```

5.3.2 Схема таблицы

Записи таблицы имеют фиксированную структуру, повторяющую столбцы таблицы. Схема представлена типом axipy.da.Schema. Свойство axipy.da.Schema.coordsystem указывает на то, что таблица является пространственной. Атрибуты axipy.da.Attribute перечислены в том же порядке, что и столбцы таблицы.

Совет: Схему можно рассматривать как список list атрибутов axipy.da.Attribute. Манипулировать атрибутами схемы можно также, как элементами списка.

Схему таблицы можно получить, используя свойство axipy.da.Table.schema.

Например:

```
schema = table.schema()
```

Схема имеет простую стандартную структуру и с ней просто работать. Например, можно легко вывести все имена атрибутов:

```
schema.attribute_names
# эквивалентно
[attr.name for attr in schema]
```

5.3.2.1 Атрибуты схемы

Атрибут представлен типом axipy.da.Attribute. Его главные параметры - это имя name и тип typedef. Тип атрибута представляется строкой. В нем может быть указана максимальная длина - для строк и десятичного типа через двоеточие, например, `string:254`. И точность - для десятичного типа через точку, например, `decimal:7.3`.

Доступные типы:

Тип	Описание
string	строка
int	целое число
double	вещественное число с плавающей запятой
decimal	вещественное число с фиксированной запятой
bool	логическое значение
date	дата
time	время
datetime	дата и время

Специальный атрибут Система Координат `coordsystem` содержит значение СК и указывает на то, что таблица пространственная - может содержать геометрию и стиль.

См.также:

Подробнее в главе Системы Координат.

Создание схемы таблицы. Вспомогательные функции

Класс `axipy.da.Attribute` содержит вспомогательные функции `axipy.da.Attribute.string()` и другие для создания атрибутов; для создания схемы таблицы используется конструктор - `axipy.da.Schema`.

Например, так можно создать схему таблицы:

```
schema = Schema(
    Attribute.string('Столица', 25),
    Attribute.string('Capital', 25),
    Attribute.string('Страна', 30),
    Attribute.string('Country', 30),
    Attribute.decimal('Cap_Pop', 8, 5),
    coordsystem='prj:Earth Projection 12, 62, "m", 0'
)
```

Свойства `axipy.da.Attribute.length` и `axipy.da.Attribute.precision` позволяют получить длину и точность типа. Список всех свойств описан в справочнике на тип `axipy.da.Attribute`.

5.3.2.2 Чтение записей

Объект таблица `axipy.da.Table` дает возможность работать с записями `axipy.da.Feature`. Метод `axipy.da.Table.items()` возвращает итератор (`iterator`) по записям.

См.также:

Подробнее о записях в главе Записи.

```
features = table.items()
for feature in features:
    # обработка записи
    ...
```

Возвращается именно Итератор. При чтении он движется вперед и в конце становится пустым. Чтобы начать чтение сначала, нужно создать новый итератор.

5.3.3 Создание таблиц

Создавать таблицы несколько сложнее, чем открывать готовые. Для них нужно определить схему, СК, Провайдер и пр. Все эти параметры были рассмотрены выше.

Провайдер может быть задан явно:

```
newtable = provider_manager.tab.create_open('../path/to/datadir/newtable.tab', schema)
```

или найден автоматически из расширения файла:

```
newtable = provider_manager.createfile('../path/to/datadir/newtable.tab', schema)
```

См.также:

axipy.da.ProviderManager.tab, axipy.da.ProviderManager.createfile().

Добавим в таблицу несколько записей из вселенной Властелин Колец:

```
features_to_insert = [
    Feature({'country': 'Мордор', 'capital': 'Барад-Дур'}),
    Feature(country='Гондор', capital='Минас Тирит'),  # создание с использованием␣
↪**kwargs
    Feature({'country': 'Рохан'}),  # не обязательно подавать все значения, они будут␣
↪пустыми
]
newtable.insert(features_to_insert)
```

Иногда может потребоваться массовая вставка записей в таблицу. В случае, если это делать по одной записи, то после каждой вставки будут отрабатываться события на изменение данных, которые в свою очередь используются в инструментах и прочих GUI компонентах. Для того, чтобы это избежать предлагается два метода. Рассмотрим их на примере вставки 1000 записей в таблицу:

1. С предварительных занесением в список list и последующей единовременной вставкой:

```
table = provider_manager.openfile('sample.tab')
point = Point(1,1)
pstyle = PointStyle()
features = []
for i in range (1, 1000):
  fpoint = Feature({}, geometry = point, style = pstyle )
  features.append(fpoint)
table.insert(features)
table.commit()
```

Но при использовании данного метода при большом количестве данных есть вероятность перерасхода памяти. Этого можно избежать, если воспользоваться вторым подходом.

2. Использование функции-генератора. При этом читаемые данные сразу же используются при вставке. Этот метод можно использовать, как пример, если необходимо зачитать данные из текстового файла с неподдерживаемым форматом.

```
table = provider_manager.openfile('sample.tab')
point = Point(1,1)
pstyle = PointStyle()
```

(continues on next page)

(продолжение с предыдущей страницы)

```
def generate_features(): # функция-генератор
    for i in range(1, 1000):
        yield Feature(geometry = point, style = pstyle)

table.insert(generate_features())
table.commit()
```

Или же это можно записать в другом виде (результат аналогичный):

```
table = provider_manager.openfile('sample.tab')
point = Point(1,1)
pstyle = PointStyle()
generator = ( Feature(geometry = point, style = pstyle) for i in range(1, 1000) )
table.insert(generator)
table.commit()
```

5.3.4 Редактирование таблиц

Один из возможных способов редактирования таблиц - открыть исходную таблицу, создать целевую таблицу с той же или другой структурой. И при чтении записей из исходной таблицы редактировать их и записывать в целевую. В результате получится отредактированная копия.

Например, для таблицы `world` необходимо оставить только колонки “Страна” и “Население”; и оставить только те страны, население которых больше `100 миллионов`.

Вот один из вариантов, как это можно реализовать.

```
def is_over_100million(feature) -> bool:
    return feature['Население'] > 100_000_000


orig_table = provider_manager.openfile('../path/to/datadir/world.tab')
schema = Schema(
    Attribute.string('Страна'),
    Attribute.integer('Население'),
)
copy_table = provider_manager.createfile('../path/to/edited_world.tab', schema)
orig_features = orig_table.items()

filtered_features = filter(is_over_100million, orig_features)
copy_table.insert(filtered_features)
```

При редактировании таблицы так-же присутствует возможность более гибкого управления производимыми в таблице изменениями. Допустимо выполнение отката как назад (отмена последних изменений) `axipy.da.Table.undo()`, так и откат вперед (возврат после выполнения отката назад) `axipy.da.Table.redo()`.

```
table.insert(feature1)
table.insert(feature2)
if table.can_undo:
   table.undo()
```

В рассмотренном примере будет вставлена только feature1.

5.3.5 Запросы

SQL-запросы являются мощным инструментом обработки данных. При выполнении запроса к таблицам образуется отдельный объект данных. При открытии данных они попадают в единый каталог axipy.da.DataManager. Запрос выполняется относительно всех таблиц, находящихся в этом каталоге, с помощью метода axipy.da.DataManager.query().

```
query_text = 'SELECT * FROM world WHERE Население > 100000000'
query_table = provider_manager.query(query_text)
```

Список поддерживаемых функций можно найти в руководстве пользователя ГИС «Аксиома» и в диалоге построения SQL-запросов самого приложения ГИС «Аксиома».

5.3.5.1 Интерфейс Qt

Более низкоуровневый доступ к данным возможен через стандартный Qt интерфейс PySide2.QtSql и вспомогательный axipy.sql. Это позволяет:

- избежать лишних округлений и нормализации значений, так как нет схемы таблицы axipy.da.Schema и атрибутов axipy.da.Attribute;
- не конвертировать значения, так как нет приведения к axipy.da.Feature;
- выделять меньше ресурсов, так как результат читается по одной записи без создания объекта данных axipy.da.Table;
- проще интегрировать с другим Qt кодом.

Примечание: Доступ к колонкам с геометрией и стилем осуществляется через значения axipy.sql.geometry_uid и axipy.sql.style_uid.

Все открытые таблицы образуют базу данных, которую можно получить функцией axipy.sql.get_database().

```
db = axipy.sql.get_database()
query = QSqlQuery(db)
query.exec_('SELECT * FROM world WHERE Население > 100000000')
while query.next():
    print(query.value(0))
```

В случае, если запрос выполняется не из оболочки ГИС «Аксиома», а в отдельном приложении, то перед тем, как выполнить запрос, необходимо в каталоге зарегистрировать все таблицы, участвующие в запросе:

```
table = provider_manager.openfile('world.tab')
data_manager.add(table)
query_table = provider_manager.query('select * from world', table)
```

Обработка ошибок

При выполнении запроса возвращается признак успешности выполнения. Если выполнение оказалось неуспешным, функция PySide2.QtSql.QSqlQuery.lastError() может показать последнюю ошибку. Стандартная обработка ошибок в PySide2.QtSql выглядит следующим образом:

```
q = QSqlQuery(database)
success = q.exec_(sql_text)
if not success:
    # Передаем ошибку выше
    raise RuntimeError(q.lastError().text())
```

5.4 Растры

Растровое изображение – это цифровое представление рисунка, фотографии или иного графического материала в виде набора точек растра.

Класс axipy.da.Raster представляет растровый объект данных.

5.4.1 Открытие растров, расположенных в файловой системе

Для открытия растровых файлов используйте функцию openfile() объекта axipy.da.provider_manager.

```
raster = axipy.provider_manager.openfile('path/to/raster.tab')
```

5.4.2 Открытие из СУБД

Так-же поддерживается открытие данных из СУБД PostgreSQL и Oracle. Данный функционал реализован через передачу строки в формате библиотеки GDAL. Пример открытия растра из БД PostgreSQL и показа его на карте:

```
raster = provider_manager.gdal.open("PG:host=server_name dbname='test' user='postgres
↪' password='postgres' schema='public' table=table_name")
raster_layer = Layer.create(raster)
print(raster_layer)
map = Map([ raster_layer ])
mapview = view_manager.create_mapview(map)
```

Пример открытия растра из БД Oracle:

```
raster = provider_manager.gdal.open("GeoRaster:user/password@XE,GDAL_RDT,1")
```

где user/password@XE - строка соединения с БД, GDAL_RDT,1 - источник, который необходимо открыть. Подробнее см gdalinfo.

5.4.3 Открытие данных, расположенных на WEB-ресурсах

Пример открытия тайлового сервера TMS. При этом передается шаблон URL:

```
raster = provider_manager.tms.open('https://maps.axioma-gis.ru/osm/{LEVEL}/{ROW}/{COL}
↪.png')
```

Пример открытия WMTS:

```
raster = provider_manager.wmts.open('https://basemap.at/wmts/1.0.0/WMTSCapabilities.
↪xml', 'geolandbasemap')
```

5.4.4 Операции с растрами

Растры по определению имеют пространственную привязку - координатную систему `axipy.da.Raster.coordsystem` и точки привязки `axipy.da.Raster.get_gcps()`. Таким образом:

```
Изображение + Пространственная привязка = Растр
```

5.4.4.1 Регистрация

Для того, чтобы «привязать» растр (задать изображению пространственную привязку), можно воспользоваться функцией `register()` модуля `axipy.da.raster`.

Например так можно привязать растр, используя эквивалентную матрицу преобразования (`QTransform` по умолчанию). Т.е. точка на изображении (x, y) в пикселях будет привязана к точке в пространстве (x, y) в единицах Системы Координат.

```
from axipy import CoordSystem, Unit
from axipy.da.raster import register, GCP
from PySide2.QtGui import QTransform

matrix = QTransform()
coordsystem = CoordSystem.from_units(Unit.m)
register(imagefile, matrix, coordsystem)
```

Чтобы привязать растр по-другому, можно передать другую матрицу трансформации (аффинное преобразование). Предварительно ее придется узнать или посчитать. Для расчета матрицы преобразования достаточно 3 точек привязки `axipy.da.raster.GCP`. Обычно по углам изображения. Функция `axipy.da.raster.register()` также может это сделать.

```
from axipy import CoordSystem
from axipy.da.raster import register, GCP

gcps = [
    GCP((0, 0), (0, 1000)),
    GCP((100, 0), (1000, 1000)),
    GCP((0, 100), (0, 0)),
]
coordsystem = CoordSystem.from_string('prj:NonEarth 0,7')
register(imagefile, gcps, coordsystem)
```

5.4.4.2 Трансформация

Операция трансформации `axipy.da.raster.transform()` растрового файла требуется для устранения или компенсации искажений, возникающих при создании растра. Требуется, когда аффинного преобразования недостаточно.

Список 11: Пример использования

```
coordsystem = CoordSystem.from_epsg(4326)
gcps = [
    GCP((0, 0), (0, 0)),
    GCP((200, 0), (30, 30)),
    GCP((200, 200), (60, 0)),
]
transform(rasterfile, outputfile, gcps, coordsystem)
```

См.также:

Руководство пользователя ГИС Аксиома раздел «Растровые изображения»

6 Записи

Запись `Feature`, которая получается при чтении из таблицы, во многом повторяет словарь Python `dict`. В ней содержатся пары (Имя столбца: Значение). Причем значение приводится к типу столбца. Например, если это числовое поле int, а его значение 42, то запись будет содержать число 42, а не строку “42”.

Прочитанная запись никак не ссылается на таблицу и является копией. Присвоенная к переменной запись будет доступна после продвижения итератора или даже после закрытия таблицы. Но поэтому и изменения, внесенные в эту запись-копию, никак не повлияют на значения в таблице.

6.1 Атрибуты

Доступ атрибутам осуществляется по имени или номеру. Так же как для словаря, если атрибут с заданным именем не существует, то вызывается исключение `KeyError`. Если атрибут с заданным номером не существует, то вызывается исключение `IndexError`.

```
feature = next(table.items())
try:
    value = feature['attr_name']
except KeyError as e:
    print(f'Поймано исключение: {e}')
```

```
>>> Поймано исключение: "Key 'unknown_key' not found"
```

Можно задать значение по умолчанию при чтении атрибута, который может не существовать. Если его не задать, то значение по умолчанию считается равным `None`.

```
value = feature.get('attr_name', 0.0)
```

Проверка существования атрибута с помощью ключевого слова `in` так же, как в словаре:

```
if 'attr_name' in feature:
    ...
```

6.1.1 Геометрический атрибут

Доступ к специальному атрибуту Геометрия axipy.da.Geometry производится через свойство axipy.da.Feature.geometry.

Или можно использовать специальное наименование GEOMETRY_ATTR, представляющее имя геометрического атрибута:

```
geometry = feature.geometry
# эквивалентно
geom = feature[GEOMETRY_ATTR]
```

Проверка существования has_geometry():

```
if feature.has_geometry():
    ...
# эквивалентно
if GEOMETRY_ATTR in feature:
    ...
```

6.1.2 Стиль для геометрического атрибута

Стиль axipy.da.Style может содержаться в виде атрибута. Доступ к нему производится по специальному наименованию `STYLE_ATTR или свойствам axipy.da.Feature.style и has_style().

```
style = feature.style
# эквивалентно
style = feature[STYLE_ATTR]
```

```
feature.has_style()
# эквивалентно
STYLE_ATTR in feature
```

6.2 Идентификаторы записей

Записи имеют свойство axipy.da.Feature.id. Это значение зависит от типа данных. При этом не гарантируется порядок, начальное значение или отсутствие разрывов между соседними записями. Также идентификатор необязательно является числом.

7 Геометрия

Геометрический объект (геометрия) представляет собой описательную структуру данных, на базе которой формируется графическое представление векторного элемента. Оно может быть частью визуального представления объектов реального мира. В зависимости от целей, геометрия может содержать данные о системе координат, в которой она создана.

7.1 Типы

ГИС «Аксиома» поддерживает следующие типы геометрических объектов:

Простые типы геометрии:

- Точка
- Полилиния
- Полигон

Коллекции:

- Смешанная коллекция
- Коллекция точек
- Коллекция полилиний
- Коллекция полигонов

Так же возможна работа с типами данных MapInfo:

- Линия
- Прямоугольник
- Скругленный прямоугольник
- Эллипс
- Дуга
- Текст

Рассмотрим подробнее геометрические типы. Переменные из примеров создания объектов будут использованы ниже в примерах более общего характера. Более общие характеристики объектов, а так же операции над ними будут рассмотрены ниже.

7.1.1 Точка

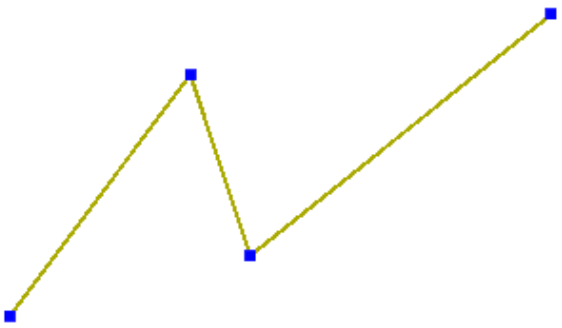
Точка представлена классом `axipy.da.Point`. Для нее характерно отсутствие таких параметров, как площади и линейных размеров. Создадим точку с координатами (10, 10) в СК Широта/Долгота. С целью визуального восприятия, результат представим в виде WKT строки (`axipy.da.Geometry.wkt`).

```
cs = CoordSystem.from_prj("1, 104")
point = Point(10, 10, cs)
print('Точка ({}, {})'.format(point.x, point.y))
'''
>>> Точка (10, 10)
'''
```

7.1.2 Полилиния

Представлена классом `axipy.da.LineString` в виде непрерывной линии, соединяющей последовательность узлов. Для нее так же характерно отсутствие площади, но присутствует длина. Точки полилинии хранятся в виде списка `list` [`axipy.utl.Pnt`], то при задании, помимо передачи в конструктор такого списка, так же допустимо указание координат в виде пар координат `tuple`. Нумерация точек при доступе по индексу начинается с 0. Доступны через свойство `axipy.da.LineString.points` Приведем пример: создадим полилинию без СК. В этом же примере заменим 3-ю вершину на другое значение и выведем полученный результат в виде wkt `axipy.da.Geometry.wkt`:

```
ls = LineString([(1, 1), (4, 5), (5, 2), (10, 6)])
ls.points[2] = (6, 3)
print(ls.wkt)
'''
>>> LINESTRING (1 1, 4 5, 6 3, 10 6)
'''
```



Так же присутствует возможность изменения существующих параметров полилинии. Добавим точку в позицию 1 и удалим точку 2:

```
ls.points.insert(1, (3,6))
ls.points.remove(2)
print(ls.wkt)
'''
>>> LINESTRING (1 1, 3 6, 6 3, 10 6)
'''
```

Поддерживается инициализация через существующий итератор. Смоделируем данную ситуацию: создадим итератор на базе точек первой полилинии и на его основе создадим полилинию:

```
itr = (a for a in ls.points)
ls_it = LineString(itr)
```

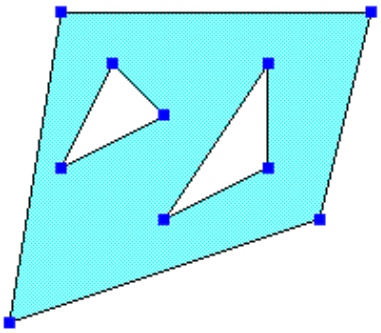
7.1.3 Полигон

Представлен классом axipy.da.Polygon. Полигон представляет собой площадной объект, или другими словами часть плоскости, ограниченная замкнутой полилинией. Кроме внешней границы, полигон может иметь одну или несколько внутренних (дырок). Ему присущи такие свойства, как длина (периметр) и площадь. Точки хранятся по аналогии с полилинией. И инициализация в конструкторе или при добавлении дырки в полигон производится по подобному принципу. Характерной особенностью является тот факт, что все контуры замкнуты и последняя точка совпадает с первой. Это касается как формы полигона, так и его дырок. В качестве примера создадим полигон:

```
polygon = Polygon((1,1), (2,7), (8,7), (7,3))
print(polygon.wkt)
'''
>>> POLYGON ((1 1, 2 7, 8 7, 7 3, 1 1))
'''
```

И добавим в него две дырки:

```
polygon.holes.append((2,4), (3,6), (4,5))
polygon.holes.append((4,3), (6,6), (6,4))
print(polygon.wkt)
'''
>>> POLYGON ((1 1, 2 7, 8 7, 7 3, 1 1), (2 4, 3 6, 4 5, 2 4), (4 3, 6 6, 6 4, 4 3))
'''
```



Как уже указывалось выше, работа с точками для полигона производится аналогично

с полилинией. Это же касается и дырок, только доступ производится через свойство axipy.da.Polygon.holes Обновим значение третьей точки для второй дырки.

```
polygon.holes[1][2] = (6,3)
print(polygon.wkt)
'''
>>> POLYGON ((1 1, 2 7, 8 7, 7 3, 1 1), (2 4, 3 6, 4 5, 2 4), (4 3, 6 6, 6 3, 4 3))
'''
```

7.1.4 Смешанная коллекция

Представлена классом axipy.da.GeometryCollection. Это нетипизированная коллекция. Может содержать внутри себя геометрии различных типов за исключением коллекций. Попробуем создать коллекцию и добавить в нее последовательно точку, полилинию и полигон. Стоит обратить внимание, что если добавляется последовательность точек, то она рассматривается как полилиния, в тоже время для указания полигона необходимо явно указать принадлежность к классу. Доступ к элементам коллекции производится по индексу, начиная со значения 0.

```
coll = GeometryCollection()
coll.append(1,2) # Точка
coll.append((3,4), (5, 5), (10, 0)) # Полилиния
coll.append(Polygon((3,4), (5, 5), (10, 0))) # Полигон
print(coll.wkt)
'''
>>> GEOMETRYCOLLECTION (POINT (1 2), LINESTRING (3 4, 5 5, 10 0), POLYGON ((3 4, 5 5,␣
↪10 0, 3 4)))
'''
```

Удалим из коллекции полилинию и полигон, а после этого добавим объекты, созданные в предыдущих примерах. Точку поменяем простой заменой по индексу:

```
coll.remove(2)
coll.remove(1)
coll.append(polygon)
coll.append(ls)
coll[0] = point
print(coll.wkt)
'''
>>> GEOMETRYCOLLECTION (POINT (10 10), POLYGON ((1 1, 2 7, 8 7, 7 3, 1 1), (2 4, 3 6,␣
↪4 5, 2 4), (4 3, 6 6, 6 3, 4 3)), LINESTRING (1 1, 3 6, 6 3, 10 6))
'''
```

При замене элемента с заданием координат работают такие же принципы, как и в конструкторе. Обновим полилинию и полигон:

```
coll[2] = [(101, 102), (103, 104), (105, 106)]
coll[1] = Polygon((101, 102), (103, 104), (105, 106))
print(coll.wkt)
'''
>>> GEOMETRYCOLLECTION (POINT (10 10), POLYGON ((101 102, 103 104, 105 106, 101 102)),
↪ LINESTRING (101 102, 103 104, 105 106))
'''
```

Поменяем первую (она же последняя) точку полигона:

```
coll[1].points[0] = (0,0)
print(coll.wkt)
'''
>>> GEOMETRYCOLLECTION (POINT (10 10), POLYGON ((0 0, 103 104, 105 106, 0 0)),␣
↪LINESTRING (101 102, 103 104, 105 106))
'''
```

Далее рассмотрим типизированные коллекции. Принципы работы аналогичны нетипизированным, за исключением того, что позволено хранение геометрий только одного типа.

7.1.5 Коллекция точек

Представлена классом axipy.da.MultiPoint. Это типизированная коллекция. Допустимо хранение только точек. Создадим коллекцию точек и добавим в нее 2 элемента разными способами:

```
mpoint = MultiPoint()
mpoint.append(11,11)
mpoint.append((12, 12))
mpoint.append(point)
print(mpoint.wkt)
'''
>>> MULTIPOINT (11 11, 12 12, 10 10)
'''
```

7.1.6 Коллекция полилиний

Представлена классом axipy.da.MultiLineString. Это типизированная коллекция. Допустимо хранение только полилиний.

```
mls = MultiLineString() # Создадим саму коллекцию.
mls.append((11, 12), (13, 14), (15, 16)) # Добавим как объект
mls.append([(21, 22), (23, 24), (25, 26)]) # Добавим как перечень точек
print(mls.wkt)
'''
>>> MULTILINESTRING ((11 12, 13 14, 15 16), (21 22, 23 24, 25 26))
'''
```

7.1.7 Коллекция полигонов

Представлена классом axipy.da.MultiPolygon. Это типизированная коллекция. Допустимо хранение только полигонов. Отдельно стоит отметить, что при добавлении/изменения геометрий в виде перечня точек, в отличие от смешанной коллекции и коллекции полилиний, в данном случае создается полигон. Рассмотрим на примере:

```
mpoly = MultiPolygon()
mpoly.append((1, 2), (3, 4), (5, 6), (7, 8))
mpoly.append(polygon) # Добавим ранее созданный с дыркой
print(mpoly.wkt)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
''''
>>> MULTIPOLYGON (((1 2, 3 4, 5 6, 7 8, 1 2)), ((0 0, 1 10, 14 15, 11 5, 10 2, 0 0),␣
↪(2 2, 44 44, 5 3, 2 2), (2 2, 2 4, 5 3, 2 2)))
'''
```

Далее рассмотрим объекты MapInfo. Одна из особенностей заключается в том, что они нестандартные, и, как следствие, не поддерживают WKT представление.

7.1.8 Линия

Представлена классом `axipy.da.Line`. Линейный объект. Описывается двумя точками: начальным и конечным узлом. Создадим линию, передав в конструктор пару точек. После этого последовательно поменяем координаты начала и конца линии.

```
line = Line((11, 11), (21, 21))
line.begin = (12, 12)
line.end = (120, 120)
```

7.1.9 Прямоугольник

Представлен классом `axipy.mi.Rectangle`. Площадной объект. Описывается минимальными и максимальными значениями по координатам X и Y. Создадим прямоугольник, задав параметры через конструктор. Затем поменяем предельные значения по X координате. Результат проконтролируем выводом значения `axipy.da.Geometry.bounds` геометрии.

```
rectangle = Rectangle(0, 0, 40, 20)
rectangle.xmin = 10
rectangle.xmax = 50
print(rectangle.bounds)
'''
>>> (10.0 0.0) (50.0 20.0)
'''
```

7.1.10 Скругленный прямоугольник

Представлен классом `axipy.mi.RoundRectangle`. Так же является площадным объектом. Отличительной особенностью от прямоугольника является наличие скруглений на углах. В остальном данные объекты аналогичны. Во избежании путаницы с параметрами, в конструкторе задается `axipy.utl.Rect` и радиусы скругления:

```
rrectangle = RoundRectangle([0, 0, 40, 20], 0.2, 0.2)
rrectangle.xRadius = 0.3
print(repr(rrectangle))
'''
>>> RoundRectangle xmin=0.0 ymin=0.0 xmax=40.0 ymax=20.0 xraduis=0.3 yraduis=0.2
'''
```

7.1.11 Эллипс

Представлен классом axipy.mi.Ellipse. Является площадным объектом. Создадим эллипс, передав в конструктор его Rect. После этого поменяем свойства.

```
ellipse = Ellipse([0,0,22,33])
ellipse.center = (10,10) # Переопределим центр
ellipse.majorSemiAxis = 10 # Задание большой полуоси
ellipse.minorSemiAxis = 5 # Задание малой полуоси
print(ellipse.bounds)
'''
>>> (0.0 5.0) (20.0 15.0)
'''
```

7.1.12 Дуга

Представлена классом axipy.mi.Arc. Линейный объект. Задается в конструкторе через axipy.utl.Rect и, дополнительно, начальный и конечный угол дуги. Создадим объект, затем выведем основные свойства:

```
arc = Arc(Rect(0,0,20,30), 45, 270)
print('center={} start={} end={}'.format(arc.center, arc.startAngle, arc.endAngle))
'''
>>> center=(10.0 15.0) start=45.0 end=270.0
'''
```

7.1.13 Текст

Представлен классом axipy.mi.Text. В отличие от описанных выше типов, его геометрические свойства определяются не только параметрами класса, но и параметрами его оформления.

```
rv = view_manager.create_reportview()
rect = Rect(8, 6, 11, 7)
text = Text("Пример", rect, view=rv)
geomItem = GeometryReportItem()
geomItem.geometry = text
geomItem.style = Style.for_geometry(text)
rv.report.items.add(geomItem)
```

Рассмотрим общие свойства геометрии.

7.2 Геометрические свойства

В зависимости от типа объекта, для него могут быть получены свойства. Продемонстрируем использование некоторых из них:

```
polygon = Polygon((0, 0), (0, 10), (10, 10), (10, 0))
print(f'Название: {polygon.name}')
print(f'Площадь: {polygon.get_area()}')
print(f'Периметр: {polygon.get_length()}')
print(f'Ограничивающий прямоугольник: {polygon.bounds}')
'''
>>> Название: Полигон
>>> Площадь: 100.0
>>> Периметр: 40.0
>>> Ограничивающий прямоугольник: (0.0 0.0) (10.0 10.0)
'''
```

7.2.1 Сериализация

ГИС «Аксиома» позволяет производить сериализацию геометрию в форматы текстового WKT или бинарного вида WKB. Подробнее

Рассмотрим на примере точечного объекта сначала для случая WKT. Будем использовать метод `axipy.da.Geometry.from_wkt()` для получения объекта из WKT и свойство `axipy.da.Geometry.wkt` для получения WKT представления полученного объекта:

```
polygon = Geometry.from_wkt('POLYGON ((10 10, 100 100, 100 10, 10 10))')
point = Geometry.from_wkt('SRID=4326;POINT (10 10)')
crs = CoordSystem.from_epsg(4326)
pline = Geometry.from_wkt('LINESTRING (30 10, 10 30, 40 40)', crs)
```

Аналогично проделаем для WKB. При этом используются, соответственно, `axipy.da.Geometry.from_wkb()` и `axipy.da.Geometry.wkb`:

```
wkb = b'\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00$@\x00\x00\x00\x00\x00\x00$@'
pnt = Geometry.from_wkb(wkb)
```

7.3 Преобразования

Для перепроецирование в другую СК используется метод `axipy.da.Geometry.reproject()`. Заметим, что исходный объект должен содержать свою СК. В рамках примера перепроецируем точку из СК Широта/Долгота в проекцию Меркатора:

```
csLL = CoordSystem.from_prj("1, 104")
csMercator = CoordSystem.from_prj("10, 104, 7, 0")
point_ll = Point(10,10, csLL)
point_merc = point_ll.reproject(csMercator)
print('point result: {}'.format(point_merc.wkt))
'''
>>> point result: POINT (1113194.90793274 1111475.10285222)
'''
```

Более простые преобразования типа сдвига axipy.da.Geometry.shift(), масштабирования axipy.da.Geometry.scale() и поворот axipy.da.Geometry.scale() игнорируют указанную СК и данные операции производятся в текущих координатах объекта. Рассмотрим на примерах:

```
polygon = Polygon((1,1), (2,7), (8,7), (7,3))
polygon_shift = polygon.shift(5,5)
# Сдвинем на величину 5 по X и Y
print('polygon shift: {}'.format(polygon_shift.wkt))
polygon_scale = polygon.scale(5,5)
# Отмасштабируем координаты относительно центра
print('polygon scale: {}'.format(polygon_scale.wkt))
# Повернем на 90 градусов против часовой стрелки относительно точки (10, 40)
polygon_rotated = polygon.rotate((10, 40), 90)
print('polygon rotate: {}'.format(polygon_rotated.wkt))
'''
>>> polygon shift: POLYGON ((6 6, 7 12, 13 12, 12 8, 6 6))
>>> polygon scale: POLYGON ((-13 -11, -8 19, 22 19, 17 -1, -13 -11))
>>> polygon rotate: POLYGON ((49 31, 43 32, 43 38, 47 37, 49 31))
'''
```

Для более сложных аффинных преобразований можно использовать axipy.da.Geometry.affine_transform() с указанием матрицы преобразований QTransform

7.4 Пространственные операции

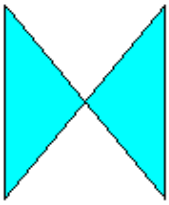
7.4.1 Нормализация объекта

Для проверки валидности геометрии предусмотрено свойство axipy.da.Geometry.is_valid. Если геометрия не проходит тест на валидность (данное свойство False), то для его нормализации используется метод axipy.da.Geometry.normalize(). Краткую аннотацию причины почему геометрия недействительна, можно воспользовавшись свойством axipy.da.Geometry.is_valid_reason.

Создадим заведомо неправильный полигон и попробуем его исправить:

```
poly_bad = Polygon([(1, 1), (6, 7), (6, 1), (1, 7)])
poly_norm = poly_bad.normalize()
print('Validate source: {} ({})'.format(poly_bad.is_valid, poly_bad.is_valid_reason))
print('Validate destination: {}'.format(poly_norm.is_valid))
print('Wkt:{}'.format(poly_norm.wkt))
'''
>>> Validate source: False (Self-intersection[3.5 4])
>>> Validate destination: True
>>> Wkt:MULTIPOLYGON (((3.5 4, 1 1, 1 7, 3.5 4)), ((3.5 4, 6 7, 6 1, 3.5 4)))
'''
```

В результате мы получили коллекцию из двух полигонов.



7.4.2 Клонирование объекта

Для создания копии объекта используется метод axipy.da.Geometry.clone():

```
point1 = Point(10, 10)
point2 = point1.clone()
point1.x = 12
print('>>>', point1.wkt, point2.wkt)
'''
>>> POINT (12 10) POINT (10 10)
'''
```

7.4.3 Логические операции

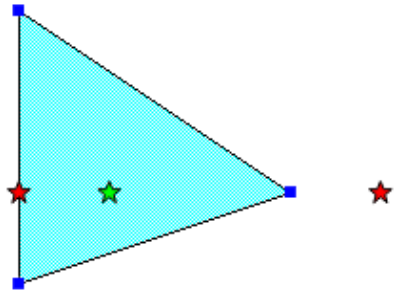
Пространственные отношения, возвращающие логический True или False:

Точное совпадение геометрий производится посредством axipy.da.Geometry.equals(), если же необходимо произвести приблизительное сравнение, то используем метод axipy.da.Geometry.almost_equals():

```
polygon1 = Polygon((1, 1), (2, 7), (7, 3))
polygon2 = Polygon((1, 1.1), (2, 7), (7, 3))
print('Точное сравнение:', polygon1.equals(polygon2))
print('Сравнение с точностью 0.2:',polygon1.almost_equals(polygon2, 0.2))
'''
>>> Точное сравнение: False
>>> Сравнение с точностью 0.2: True
'''
```

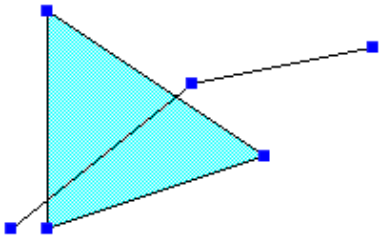
Проверка на попадание axipy.da.Geometry.contains()

```
poly1 = Polygon((1, 1), (1, 7), (7, 3))
point1 = Point(3,3)
point2 = Point(9,3)
point3 = Point(1,3)
print('Точка внутри:', poly1.contains(point1))
print('Точка снаружи:', poly1.contains(point2))
print('Точка на грани:', poly1.contains(point3))
'''
>>> Точка внутри: True
>>> Точка снаружи: False
>>> Точка на грани: False
'''
```



Проверка на частичное пересечение axipy.da.Geometry.crosses()

```
poly1 = Polygon((1, 1), (1, 7), (7, 3))
sl = LineString ((0, 1), (5, 5),  (10, 6))
print(poly1.crosses(sl))
'''
>>> True
'''
```



Проверка на отсутствие соприкосновений axipy.da.Geometry.disjoint()

```
print(poly1.disjoint(sl))
'''
>>> False
'''
```

Проверка пересечений объектов axipy.da.Geometry.intersects()

Пересечение геометрий, если результат отличен от анализируемых данных axipy.da.Geometry.overlaps()

```
poly2 = Polygon((5,1), (4,4), (10,3))
print(poly1.overlaps(poly2))
'''
>>> True
'''
```

Проверка касания axipy.da.Geometry.touches()

```
point3 = Point(1,5)
print('Точка на грани:', poly1.touches(point3))
'''
>>> True
'''
```

Функция, обратная contains axipy.da.Geometry.within()

```
print('Точка внутри:', Point(2,5).within(poly1))
''''
>>> True
'''
```

Охват одной геометрии другой axipy.da.Geometry.covers()

7.4.4 Отношения DE-9IM

Функция axipy.da.Geometry.relate() проверяет все DE-9IM отношения между объектами. Предикаты выше являются их частными случаями.

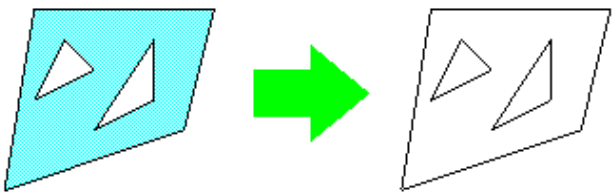
```
print('relate:', poly1.relate(Point(10,10)))
'''
relate: FF2FF10F2
'''
```

7.4.5 Операции над объектами

В данном разделе рассмотрим операции, результатом выполнения которых будут новые объекты.

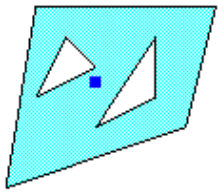
Получение границ геометрии в виде полилинии axipy.da.Geometry.boundary()

```
poly = Geometry.from_wkt('POLYGON ((1 1, 2 7, 8 7, 7 3, 1 1), (2 4, 3 6, 4 5, 2 4),␣
↪(4 3, 6 6, 6 4, 4 3))')
poly_boundary = poly.boundary()
print(poly_boundary.wkt)
'''
>>> MULTILINESTRING ((1 1, 2 7, 8 7, 7 3, 1 1), (2 4, 3 6, 4 5, 2 4), (4 3, 6 6, 6 4,␣
↪4 3))
'''
```



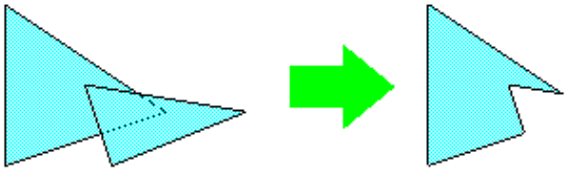
Центроид объекта axipy.da.Geometry.centroid()

```
centroid = poly.centroid()
print(centroid.wkt)
'''
>>> POINT (4 4.5)
'''
```



Вычитание объектов axipy.da.Geometry.difference()

```
poly1 = Polygon((1, 1), (1, 7), (7, 3))
poly2 = Polygon((5,1), (4,4), (10,3))
print(poly1.difference(poly2).wkt)
'''
>>> POLYGON ((1 1, 1 7, 6 3.67, 4 4, 4.6 2.2, 1 1))
'''
```



Пересечение объектов axipy.da.Geometry.intersection()

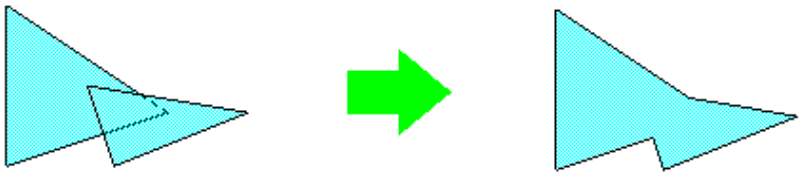
```
print(poly1.intersection(poly2).wkt)
'''
>>> POLYGON ((6 3.67, 7 3, 4.6 2.2, 4 4, 6 3.67))
'''
```

Обратное пересечение объектов axipy.da.Geometry.symmetric_difference()

```
print(poly1.symmetric_difference(poly2).wkt)
'''
>>> MULTIPOLYGON (((1 1, 1 7, 6 3.67, 4 4, 4.6 2.2, 1 1)), ((4.6 2.2, 7 3, 6 3.67, 10␣
↪3, 5 1, 4.6 2.2)))
'''
```

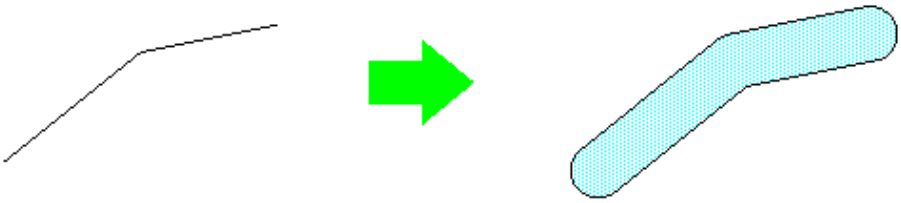
Объединение объектов `axipy.da.Geometry.union()`

```
print(poly1.union(poly2).wkt)
'''
>>> POLYGON ((1 1, 1 7, 6 3.67, 10 3, 5 1, 4.6 2.2, 1 1))
'''
```



Построение буфера `axipy.da.Geometry.buffer()`

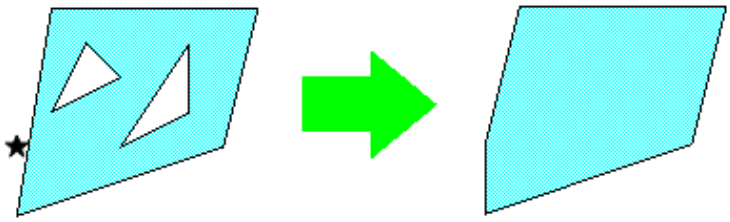
```
sl = LineString ((0, 1), (5, 5),  (10, 6))
buf = sl.buffer(1)
```



Границы объекта `axipy.da.Geometry.convex_hull()`

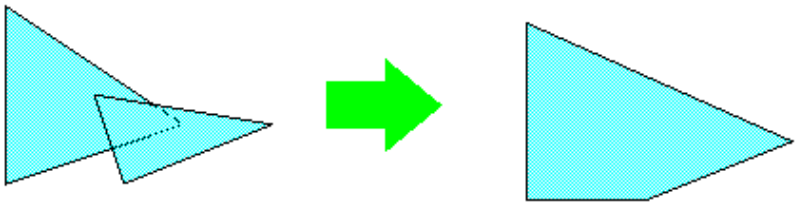
Рассмотрим на примере коллекции:

```
coll = GeometryCollection()
coll.append(polygon)
coll.append(Point(1,5))
print(coll.convex_hull().wkt)
'''
POLYGON ((1 1, 1 5, 2 7, 8 7, 7 3, 1 1))
'''
```



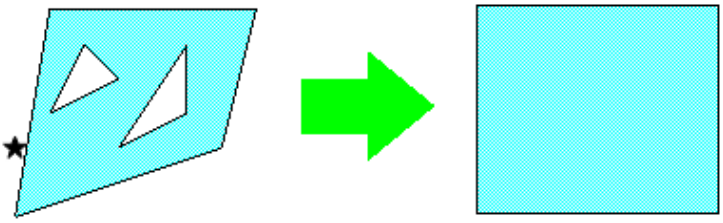
Пример с внутренними углами:

```
print(poly1.union(poly2).convex_hull().wkt)
'''
POLYGON ((1 1, 1 7, 10 3, 5 1, 1 1))
'''
```



Прямоугольные границы объекта `axipy.da.Geometry.envelope()`

```
print(coll.envelope().wkt)
'''
POLYGON ((1 1, 8 1, 8 7, 1 7, 1 1))
'''
```



7.4.6 Конвертация объектов

Список 1: Пример конвертации полигона в полилинию.

```
polygon = Polygon((0, 0), (0, 10), (10, 10))
print(polygon.to_linestring().wkt)
'''
>>> LINESTRING (0 0, 0 10, 10 10, 0 0)
'''
```

Список 2: Пример конвертации полилинии в полигон.

```
ls = LineString((0, 0), (0, 10), (10, 10))
print(ls.to_polygon().wkt)
'''
>>> POLYGON ((0 0, 0 10, 10 10, 0 0))
'''
```

8

Стиль

Стиль представляет собой описательную структуру, которая в свою очередь используется при оформлении геометрического объекта `Geometry` при его отрисовке. Стиль представлен базовым классом `axipy.da.Style`, а так-же его наследниками.

При работе с табличными данными стиль, при наличии в ней геометрического атрибута, может определяться тремя разными способами:

- Содержаться в специальной колонке таблицы в виде атрибута. В данном случае для геометрии каждой записи таблицы назначается соответствующий ей стиль.
- Определяется для колонки на уровне таблицы. В данном случае геометрия всех записей будет иметь одинаковое оформление.
- В таблице присутствует только геометрия. В данном случае стиль при отрисовке слоя будет браться как значение по умолчанию.

Рассмотрим в дальнейшем первый вариант. Для выполнения последующих примеров создадим таблицу в памяти и зарегистрируем ее в системе:

```
definition = {
    'src': '',
    'schema': Schema(
        Attribute.string('id', 60),
        coordsystem='prj:1, 104'
    )
}
table = provider_manager.create(definition)
```

Далее попробуем различными методами добавить геометрию в эту таблицу. На примере точечного объекта. Создадим точечный объект, и, если стиль оформления не имеет значения, назначим ему наиболее подходящий для данного типа (в нашем случае точки) объекта, просто передав туда нашу геометрию:

```
point = Point(10,8)
pstyle = Style.for_geometry(point)
print(pstyle.to_mapinfo())
'''
  >>> Symbol (36, 255, 12, "Map Symbols", 0,0)
'''
```

Для иллюстрации результата сформируем на базе созданных объектов геометрии и стиля запись и добавим его в ранее созданную таблицу. Итог покажем на карте:

```
fpoint = Feature(
    geometry=point,
    style=pstyle
)
table.insert([fpoint])
m = Map([table])
view = view_manager.create_mapview(m)
```

В результате получим точку на карте:

Так же доступно создание из строки формата MapBasic. Для этого используется метод `axipy.da.Style.from_mapinfo()`. Если же для существующего стиля необходимо получить строку MapBasic, то используется метод `axipy.da.Style.to_mapinfo()`. Создадим для нашей точки стиль на базе представления MapBasic. Строка формирования стиля точки будет выглядеть так:

```
pstyle = Style.from_mapinfo('Symbol (35, 255, 20)')
```

Пример добавления точки, но с использованием стиля, на основании растрового символа:

```
pstyle = PointStyle.create_mi_picture("GLOB1-32.bmp")
print(pstyle)
fpoint = Feature(
    geometry=point,
    style=pstyle
)
'''
>>> Symbol ("GLOB1-32.bmp", 0, 12, 0)
'''
```

Далее вставим в таблицу по аналогии, рассмотренной выше. Результат:

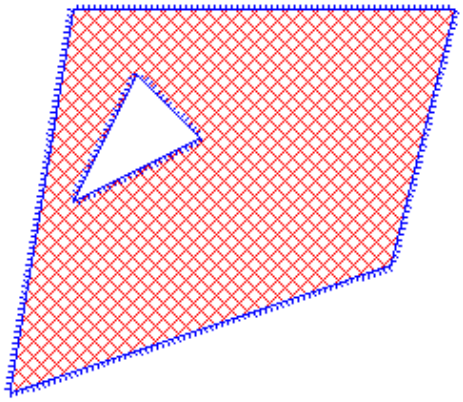
Теперь рассмотрим объект типа полигон.

```
from PySide2.QtCore import Qt

polygon = Polygon((1,1), (2,7), (8,7), (7,3))
polygon.holes.append((2,4), (3,6), (4,5))
polystyle = PolygonStyle()
polystyle.set_pen(pattern=48, color=Qt.blue)
polystyle.set_brush(pattern=8, color=Qt.red)
print(polystyle)
fpolygon = Feature(
    geometry=polygon,
    style=polystyle
)
```

(continues on next page)

(продолжение с предыдущей страницы)

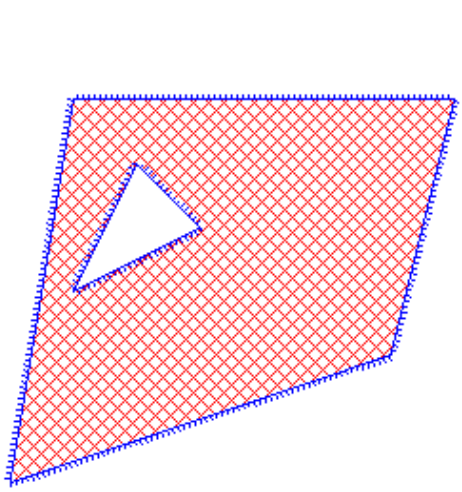
```
table.insert([fpolygon])
'''
>>> Pen (1, 48, 255)  Brush (8, 16711680, 0)
'''
```



Для сложных разнородных объектов применяются стили, которые внутри себя содержат другие стили для каждого типа используемых объектов. Для этого используется класс `axipy.da.CollectionStyle`. Из ранее созданных полигона и точки сделаем коллекцию посредством объединения. И, одновременно с этим, на базе ранее созданных стилей сделаем сложный стиль:

```
collstyle = CollectionStyle()
collstyle.for_point(pstyle)
collstyle.for_polygon(polystyle)
print(collstyle)
collection = polygon.union(point)
fcollection = Feature(
    geometry=collection,
    style=collstyle
)
table.insert([fcollection])
'''
>>>  Symbol ("GLOB1-32.bmp", 0, 12, 0) Pen (1, 48, 255)  Brush (8, 16711680, 0)
'''
```

В результате получим следующий объект:



9 Отображение данных

9.1 Слой

Для отображения данных таблицы или растра необходимо создать на основе этого источника данных слой `axipy.render.Layer`.

```
from axipy.render import Layer

layer = Layer.create(table)
```

В зависимости от типа передаваемого объекта будет создан векторный или растровый слой.

9.1.1 Свойства подписей

Настройки автоматического подписывания определяются классом `Label` свойством `label`.

Список 1: Пример использования

```
world = generate_layer_for_geometry_table()
# Зададим в качестве формулы метки атрибут "Страна" и запретим перекрытие меток друг␣
↪другом:
world.label.text = "Страна"
world.label.placementPolicy = LabelOverlap.DisallowOverlap
# Задание стиля оформления слоя
style_lay = Style.from_mapinfo("Pen (1, 2, 0) Brush (8, 255) Symbol (33,255,14)")
world.overrideStyle = style_lay
# Для сброса переопределения достаточно задать значение None
world.overrideStyle = None
```

Настройки автоматического подписывания `Label` повторяют соответствующие настройки в интерфейсе ГИС Аксиома диалога «Свойства слоя» > «Подписи».

См.также:

Подробнее в разделе «Подписывание» руководства пользователя для ГИС Аксиома.

9.2 Карта

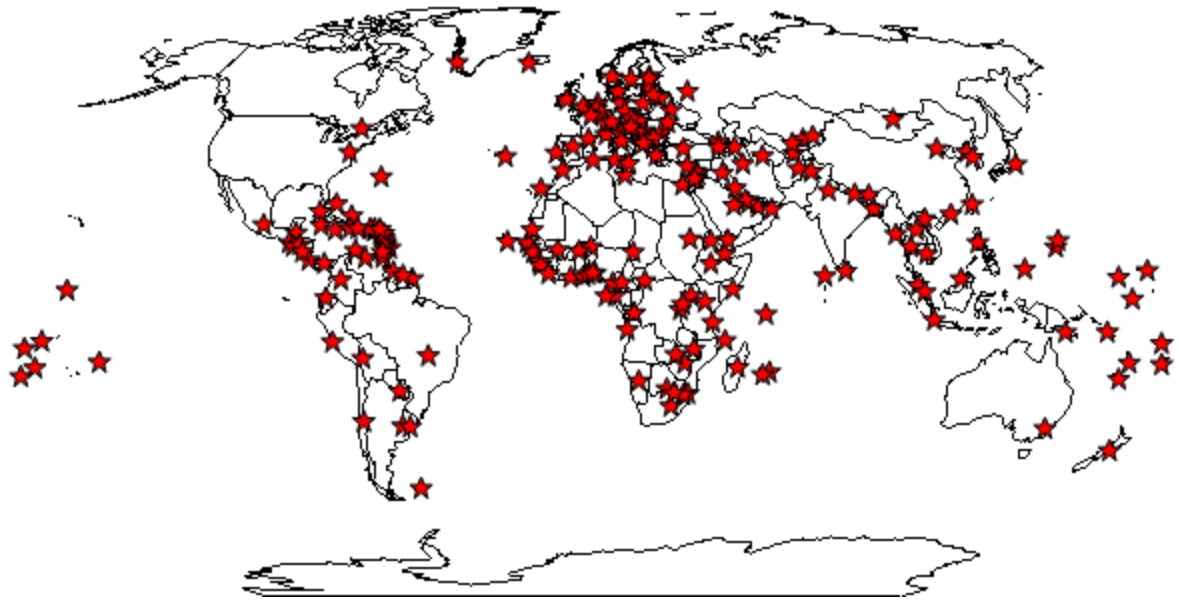
Совокупность слоев образует карту axipy.render.Map. Порядок отрисовки слоев прямо зависит от их расположения в карте. То есть первый слой будет на самом верху, а последний - в самом низу.

Создадим карту с двумя слоями. Передадим в карту список таблиц - из них автоматически создадутся слои с параметрами по умолчанию.

```
import axipy

world = axipy.provider_manager.openfile('../path/to/datadir/example.gpkg', dataobject=
↪'world')
capital = axipy.provider_manager.openfile('../path/to/datadir/worldcap.tab')
map = axipy.Map([capital, world])
```

Карту можно вывести в изображение - PySide2.QtGui.QImage, которое, например, можно в качестве результата сохранить в файл.

```
image = map.to_image(600, 300)
```



Полученную картинку можно сохранить как растр в файловой системе. Формат файла будет определяться его расширением:

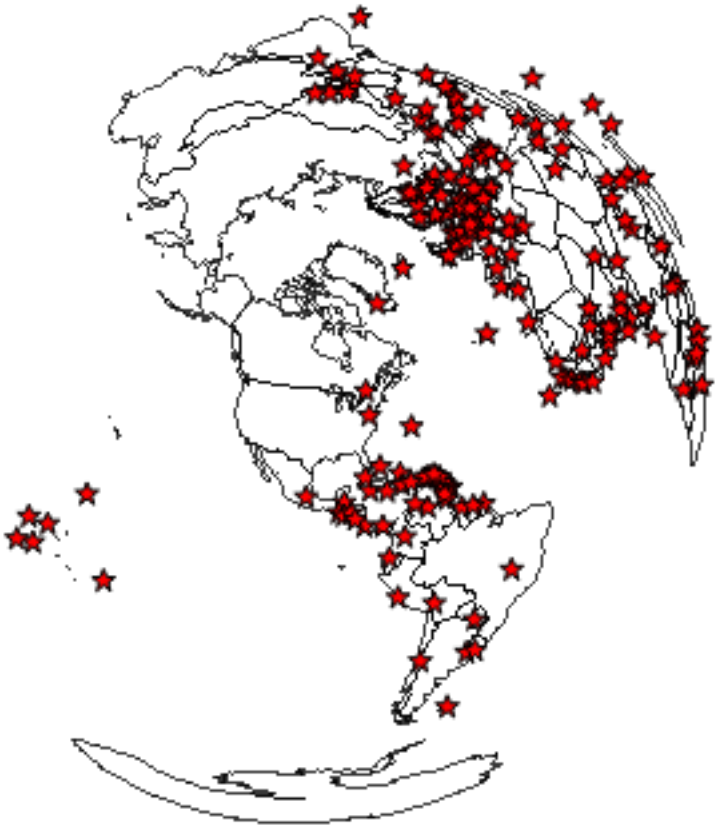
```
image.save('../path/to/outdir/map.png')
```

```
>>> True
```

Мы указали только размеры изображения. Карта вывелась в Системе Координат (СК), наиболее подходящей для отображения слоев в ней. В нашем случае обе таблицы оказались в одной СК, которая и была выбрана для отображения.

Теперь отрисуем эту же карту Азимутальной СК:

```
from axipy.cs import CoordSystem

azimuth = CoordSystem.from_epsg(2163)
image = map.to_image(600, 300, coordsystem=azimuth)
```



Границы карты по умолчанию определились равными границам СК. Снова нарисуем нашу карту уже в Широте/Долготе в границах, примерно включающих Италию. Ограничивающий прямоугольник указываем в градусах:

```
longlat = CoordSystem.from_epsg(4326)
image = map.to_image(600, 300, coordsystem=longlat, bbox=(5, 35, 15, 15))
```

Теперь попробуем для слоя capital задать выражение для отображаемых меток axipy.render.VectorLayer.label, и для обоих слоев переопределить стиль оформления, сделав его однообразным axipy.render.VectorLayer.overrideStyle. Заметим, что перечень доступных слоев карты доступен через свойство axipy.render.Map.layers. Т.е. помимо передачи перечня слоев в конструктор карты, также возможно управление этим списком позже.

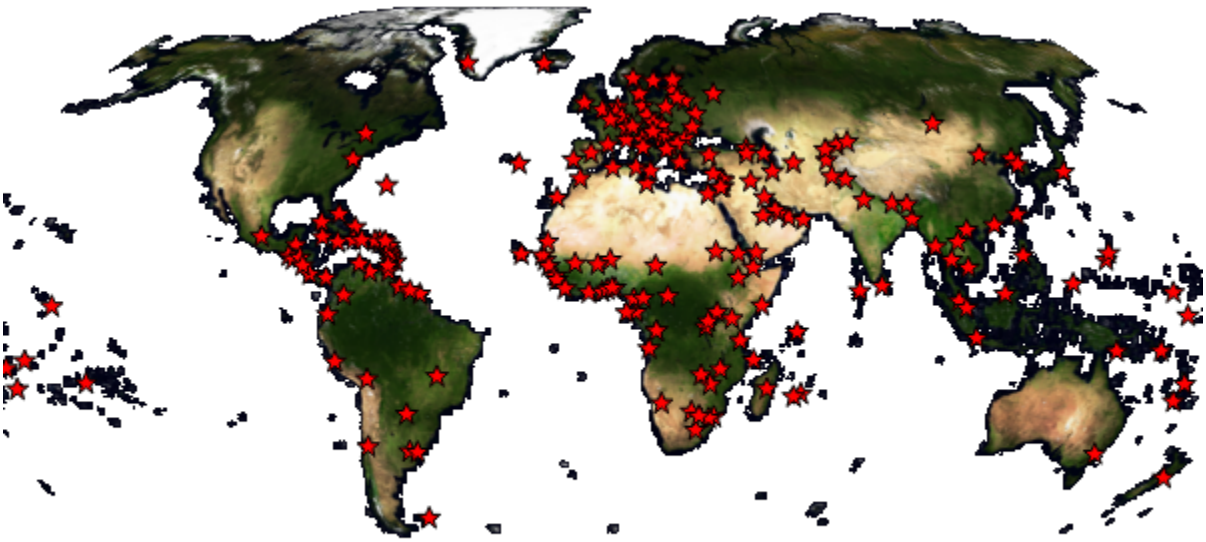
```
from axipy.da import *

lay_capital = map.layers[0]
lay_capital.label.text = 'Столица'
lay_capital.label.placementPolicy = Label.DisallowOverlap
lay_capital.overrideStyle = Style.from_mapinfo('Symbol (34,255,6)')
lay_world = map.layers['world']
lay_world.overrideStyle = Style.from_mapinfo('Pen (1, 2, 0) Brush (8, 255)')
image = map.to_image(600, 300, coordsystem=longlat, bbox=(5, 35, 15, 15))
```

В рамках примера по управлению слоями в конце удалим слой со столицами (самый верхний):

```
map.layers.remove(0)
```

Создадим карту на базе растра как подложки и установим прозрачным цвет фона.

```
from axipy import *
from PySide2.QtGui import QColor

raster = provider_manager.openfile('../path/to/datadir/TrueMarble.tab')
rasterLayer = Layer.create(raster)
rasterLayer.transparentColor = QColor('#000014')

mapRaster = Map([capital, rasterLayer])
image = mapRaster.to_image(600, 320)
```



9.3 Тематические слои

Тематическая карта отображает ваши данные в виде условных знаков, выделяя их оттенками, цветами, штриховками, а также представляя их в виде столбчатых и круговых диаграмм.

Для векторных слоев `axipy.render.VectorLayer` есть возможность формирования и отрисовки тематических слоев. Т.е. применить оформление на базе атрибутивной информации.

Тематические слои добавляются как дочерние к их базовому слою.

```
from axipy import *

world = map.layers[0]
thematic = RangeThematicLayer('Население')
world.thematic.add(thematic)
```

Поддерживаются следующие виды тематических слоев:

- RangeThematicLayer - Интервалы
- PieThematicLayer - Круговые диаграммы
- BarThematicLayer - Столбчатые диаграммы
- SymbolThematicLayer - Знаки
- IndividualThematicLayer - Индивидуальные значения
- DensityThematicLayer - Плотность точек

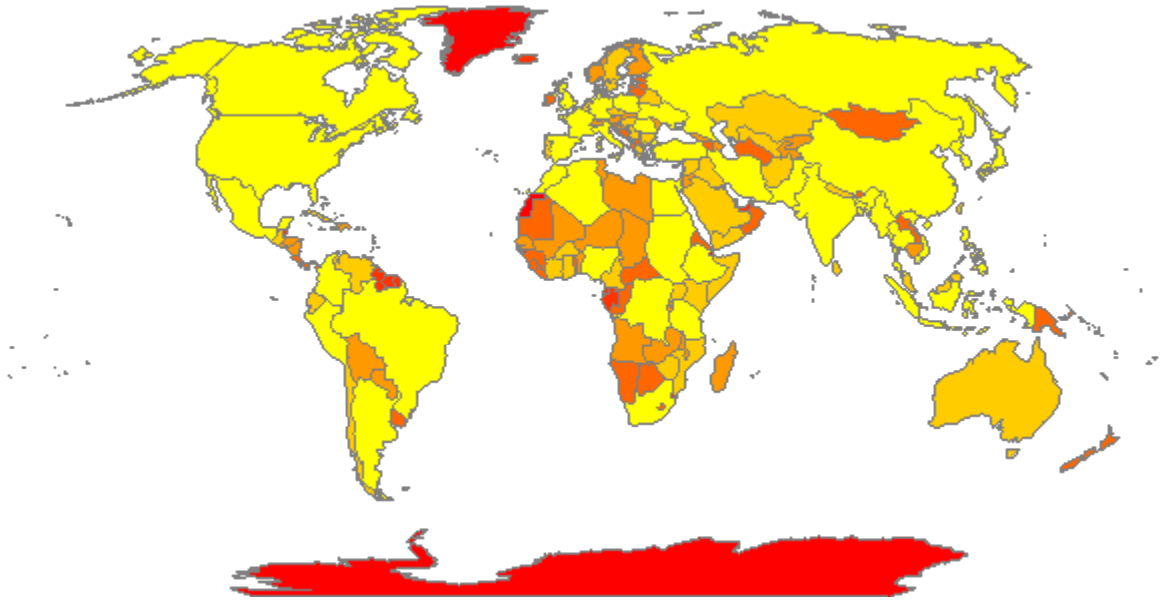
Для более удобного распределения по цветам используются различного рода алгоритмы. Выбор того или иного алгоритма обусловлен исходными требованиями к составлению тематики. Эти алгоритмы сгруппированы в базовом интерфейсе

axipy.render.ReallocateThematicColor и могут быть использованы в наследниках. Поддерживаются следующие виды распределения:

- По двум заданным крайним цветам axipy.render.ReallocateThematicColor.assign_two_colors().
- По двум заданным крайним цветам и цвету разрыва axipy.render.ReallocateThematicColor.assign_two_colors().
- По спектру axipy.render.ReallocateThematicColor.assign_rainbow().
- Градация серого axipy.render.ReallocateThematicColor.assign_gray().
- Монотонная заливка разной яркости axipy.render.ReallocateThematicColor.assign_monotone().

Рассмотрим на примере тематики по интервалам. Построим тематику по атрибутивному полю “Население” на 6 интервалов с равномерным распределением по количеству записей. Цвета распределим градиентом от желтого до красного.

```
table_world = provider_manager.openfile('world.tab')
world = Layer.create(table_world)
range1 = RangeThematicLayer("Население")
range1.ranges = 6
range1.splitType = RangeThematicLayer.EQUAL_COUNT
range1.assign_two_colors(Qt.yellow, Qt.red)
world.thematic.add(range1)
```



Поменяем стиль оформления для первого интервала:

```
range1.set_style(0, PolygonStyle(45, Qt.blue))
```

Так же есть возможность ручного переопределения значений разбивки по интервалам. Т.е. задание минимального и максимального значений требуемого интервала. Переопределим верхнее значение для первого интервала:

```
iv = range1.get_interval_value(0)
print('Old values:', iv)
```

(continues on next page)

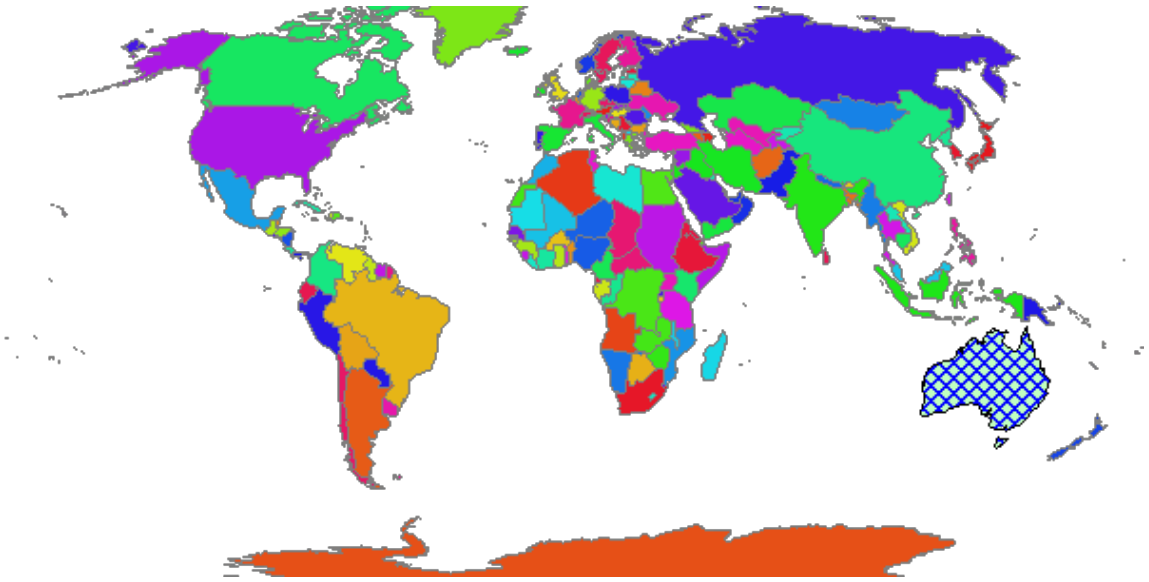
(продолжение с предыдущей страницы)

```
range1.set_interval_value(0, (iv[0], 100000.0))
print('New values:', range1.get_interval_value(0))
```

```
>>> Old values: (0.0, 66687.0)
>>> New values: (0.0, 100000.0)
```

Создадим тематику с отдельными значениями по полю „Страна“. Распределение по цветам случайное:

```
individual = IndividualThematicLayer('Страна')
individual.assign_rainbow()
world.thematic.add(individual)
individual.set_style(0, PolygonStyle(45, Qt.blue))
```



Изменим цвет интервала по индексу 1 на желтый.

```
s = individual.get_style(1)
s.polygon.fill.color = Qt.yellow
individual.set_style(1, s)
```

Необходимую тематику слоя можно получить по ее индексу `axipy.render.Layer.thematic()`:

```
range1 = world.thematic[0]
```

Если необходимо просмотреть все тематики слоя:

```
for t in world.thematic:
    print('thematic:', t.title)
```

```
>>> thematic: Интервалы
>>> thematic: Значения
```

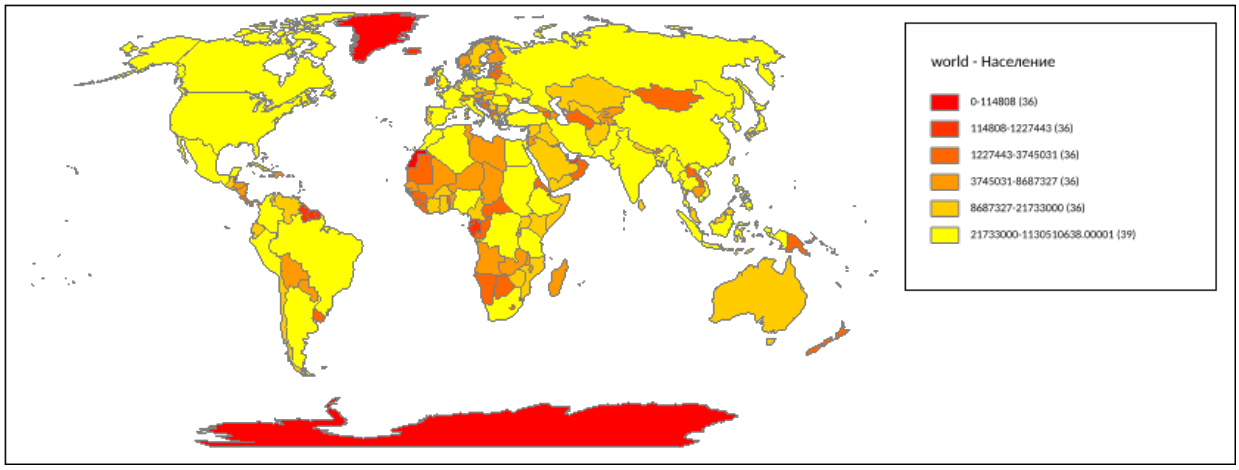
9.4 Легенда

Для вывода условных обозначений используется легенда. Легенда – таблица, содержащая образцы условных обозначений и письменных пояснений к ним. В ГИС «Аксиома» легенды карт представляются в отдельных окнах. Попробуем отрисовать в растре ранее созданные слой и тематику по интервалам. Заметим, что легенду также можно отрисовать на одном растре вместе с картой.

```
from axipy import *
from PySide2.QtGui import QImage, QPainter

legend_world = Legend(lay_world)
legend_world.position = (10, 10)

legend_thematic = Legend(thematic)
legend_thematic.position = (200, 10)

image = QImage(500, 200, QImage.Format_ARGB32_Premultiplied)
image.fill(Qt.white)
painter_legend = QPainter(image)
context_legend = Context(painter_legend)

legend_world.draw(context_legend)
legend_thematic.draw(context_legend)
```

Создадим легенду, на этот раз сразу переведя в `PySide2.QtGui.QImage`, и скомпонуем с тематическим слоем, полученным ранее:

```
from axipy.render import Legend

legend_thematic = Legend(thematic)
legend_thematic.position = (10, 5)
legend_image = legend_thematic.to_image(200, 170)
```



Доступ к элементам легенды производится через свойство `axipy.render.Legend.items`.

```
for it in legend_thematic.items:
    print(it.title, it.visible, it.style.to_mapinfo())
```

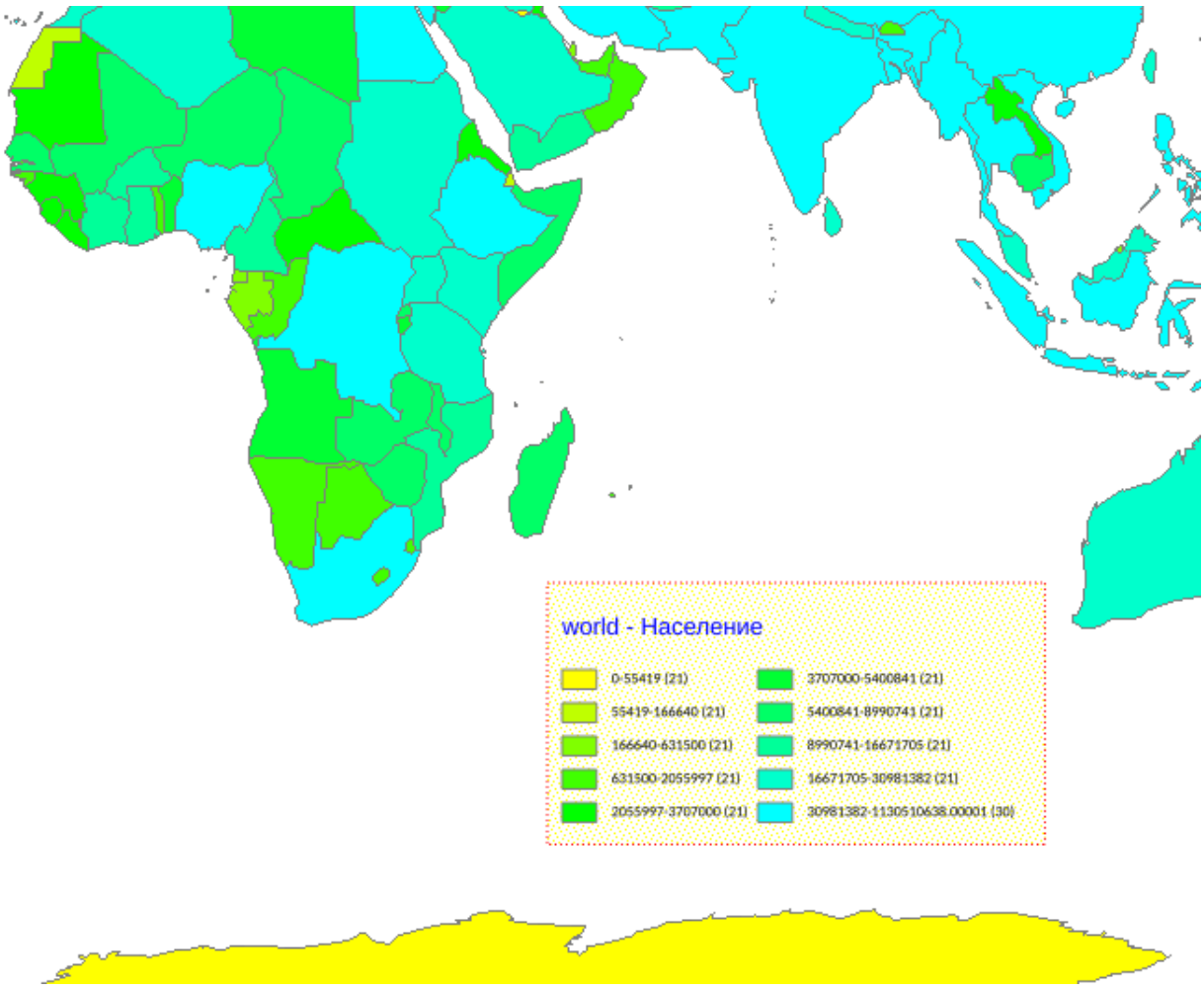
```
>>> 0-55419 True Pen (1, 2, 0) Brush (45, 255)
>>> 166640-631500 True Pen (1, 2, 8421504) Brush (2, 8453888)
>>> 631500-2055997 True Pen (1, 2, 8421504) Brush (2, 4259584)
```

Если требуется изменить какой-то элемент. к примеру сделать его скрытым или изменить описание, то в данном случае необходимо сначала запросить требуемый элемент, изменить его характеристики и обновить на модифицированный. Прямое изменение не поддерживается.

```
item = legend.items[0]
item.title = 'Описание'
legend.items[0] = item
```

Если легенду необходимо как-то выделить на общем фоне или она с ним сливается, то можно указать стиль рамки и заливки заднего фона:

```
legend.border_style = LineStyle(3, Qt.red)
legend.fill_style = PolygonStyle(49, Qt.yellow)
```



9.5 Отчет

Для вывода информации на печать предусмотрено создание отчетов. Отчет формируется на базе стандартного подхода работы с принтером в Qt `PySide2.QtPrintSupport.QPrinter`. Создадим макет отчета, в который поместим геометрический объект и карту. Вывод сделаем в файл формата PDF. Для этого предварительно создадим объект принтера и установим необходимые свойства

```
from PySide2.QtPrintSupport import QPrinter

printer = QPrinter()
printer.setOutputFormat(QPrinter.PdfFormat)
printer.setOutputFileName('../path/to/outdir/report.pdf')
```

Далее, создадим сам отчет и в конструктор передадим созданный ранее принтер.

```
from axipy import *

report = Report(printer)
```

Создадим геометрический элемент и добавим его в отчет. Координаты в единицах измерения листа принтера.

```
geometryReportItem = GeometryReportItem()
geometryReportItem.geometry = Polygon((10, 10), (10, 100), (100, 100), (10, 10))
geometryReportItem.style = PolygonStyle(45, Qt.red)
report.items.add(geometryReportItem)
```

Аналогично добавим карту.

```
table = provider_manager.openfile('world.tab')
world = Layer.create(table)
map = Map([ world ])
mapReportItem = MapReportItem(Rect(10, 110, 200, 210), map)
mapReportItem.scale = 200000000
report.items.add(mapReportItem)
```

Контекст для печати по подобию рассмотренному контексту для карты.

```
from PySide2.QtGui import QPainter

painterReport = QPainter(printer)
context = Context(painterReport)
```

Производим печать.

```
report.draw(context)
```

В результате в файловой системе мы получим файл report.pdf, который содержит геометрический элемент и карту.

- Слой
- Свойства подписей
- Тематические слои
- Легенда
- Отчет

10 Интерфейс

10.1 Создание кнопок

Расположение кнопки в интерфейсе ГИС «Аксиома» определяется Вкладкой и Группой. Например, вкладка “Основные” группа “Команды”. В модуле `axipy.menubar` есть необходимые функции для создания кнопок.

```
button = ActionButton('Простое действие', on_click=lambda: print('triggered'))
position = Position('Основные', 'Команды')
position.add(button)
```

Детально разберем, что делает этот пример.

Создается кнопка с текстом “Простое действие”, и ,используя параметр `on_click`, привязывается нажатие на кнопку к анонимной лямбде, которая печатает в консоль текст «triggered». Это обработчик нажатия кнопки. Обработчиком может служить любой callable-объект(функтор) без параметров, т.е. функции, лямбды, объекты с методом `__call__`. Также можно задать иконку кнопки параметром `icon=`. Иконкой может быть строка-ссылка на ресурс или объект типа `PySide2.QtGui.QIcon`.

Далее ищется расположение в интерфейсе. Если вкладка или группа с такими именами отсутствуют, то они будут созданы при добавлении кнопки.

В последней строке кнопка добавляется в заданное расположение.

10.2 Передача параметров в инструменты

Дополнительные параметры могут быть переданы прямо в конструктор инструмента axipy.gui.MapTool при создании кнопки axipy.menubar.ToolButton. Для этого необходимо «обернуть» вызов конструктора в функцию, захватив (capture) все необходимые параметры. Например, сравните:

Список 1: Инструмент без параметров

```
button = ToolButton('Мой инструмент', MyTool)
```

Список 2: Инструмент с захватом параметров

```
button = ToolButton('Мой инструмент', lambda: MyTool(config, params))
```

10.3 Наблюдатели значений

В приложении ГИС «Аксиома» многие кнопки и инструменты меняют свойство доступности в зависимости от текущего состояния. Например, если кнопка производит действие над выбранным объектом, то логично сделать эту кнопку доступной только при наличии выбранных объектов. Чтобы проще задавать подобное поведение для своих кнопок, предлагается использовать axipy.da.ValueObserver и место их регистрации axipy.da.state_manager.

Так, чтобы сделать кнопку активной только при наличии выборки, ее можно создать следующим образом, передав параметр enable_on функции axipy.menubar.create_button():

```
button = menubar.create_button('Имя кнопки', on_click=action_func,
        enable_on=state_manager.Selection)
```

Идентификаторы наблюдателей по умолчанию перечислены в axipy.da.DefaultKeys; они также доступны в axipy.da.state_manager.

Для инструментов идентификатор рекомендуется задавать в самом классе инструмента, переопределив параметр axipy.gui.MapTool.enable_on.

Возможно добавление своих наблюдателей, в том числе их комбинация с уже существующими axipy.da.StateManager.create().

Примечание:

- Наблюдатели значений могут использоваться и для других целей.
- Не все наблюдатели имеет смысл комбинировать друг с другом; некоторые значения являются взаимоисключающими.
- Не все наблюдатели напрямую подходят для задания доступности кнопкам, а скорее лишь те, которые имеют тип bool. В общем случае наблюдение может быть за любыми базовыми значениями: int, str, list и прочее.

Чтобы просто получить текущее значение наблюдателя, используется метод axipy.da.ValueObserver.value().

10.4 Панель активного инструмента

Если при работе с инструментом пользователю регулярно нужно взаимодействовать с различными графическими элементами / настройками то для этого можно воспользоваться готовым решением из `ActiveToolPanel`. Это обертка вокруг стандартной панели `QDockWidget` предоставляющая дополнительные возможности:

1. Можно задать наблюдателя который будет автоматически следить за доступностью панели.
2. Предопределенное место для панели активного инструмента.
3. Готовые компоненты с кнопками управления для упрощения добавления пользовательского графического элемента.

При написании плагинов текущий экземпляр `ActiveToolPanel` можно получить из метода `axipy.AxiomaInterface.active_tool_panel()`. Взаимодействие с панелью активного инструмента идет через обработчик который можно создать через один из методов `make_acceptable()` или `make_custom()`. При работе с панелью активного инструмента через обработчик `AxipyAcceptableActiveToolHandler` созданный через `make_acceptable()` то на панели активного инструмента по умолчанию расположены кнопки «Применить» и «Отмена». При нажатии на них будут посланы соответствующие сигналы `accepted` и `rejected`. Если нужно настроить кнопки управления по своему усмотрению то можно воспользоваться обработчиком `AxipyCustomActiveToolPanelHandler`, который создаётся с помощью `make_custom()`.

Список 3: Создание обработчика для взаимодействия с панелью активного инструмента.

```
service = ActiveToolPanel()
# Любой пользовательский графический элемент
widget = QWidget()

# Создаём обработчик для панели активного инструмента через который
# будем управлять панелью.
tool_panel = service.make_acceptable(
    title="Мой инструмент",
    observer_id=DefaultKeys.SelectionEditable,
    widget=widget)

# Подписываемся на сигнал отправляемый после нажатия на кнопку "Применить" в панели
tool_panel.accepted.connect(lambda: print("Применяем изменения"))
```

Чтобы показать панель активного инструмента с переданным ранее графическим элементом нужно вызвать `activate()`, а для закрытия `deactivate()`.

Список 4: Включение/Выключение панели активного инструмента.

```
class PyTool(MapTool):

    def mousePressEvent(self, event: QMouseEvent) -> Optional[bool]:
        if event.button() == Qt.LeftButton:
            self.tool_panel.activate()
        return self.PassEvent

    def keyPressEvent(self, event: QKeyEvent) -> Optional[bool]:
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        if event.key() == Qt.Key_Escape:
            self.tool_panel.deactivate()
            return self.BlockEvent
        return super().keyPressEvent(event
```

См.также:

Создание и добавление кнопок Создание кнопок

В коде выше в методе `mousePressEvent()` при нажатии левой кнопкой мыши мы показываем панель с активным инструментом, а в методе `keyPressEvent()` при нажатии на клавишу Esc панель закрываем.

10.5 Окно редактирования карты

Окно карты `axipy.render.Map`, созданное программно, можно экспортировать в виде растра или же поместить в отчет. Если же стоит задача проведения некоторых манипуляций с картой, таких как редактирование геометрии, то для проведения подобных манипуляций ее необходимо поместить в окно редактирования `axipy.gui.MapView`. Для создания этого окна необходимо использовать метод `axipy.gui.ViewManager.create_mapview()`. Данный метод доступен через сервис view_manager. Рассмотрим на примере:

Список 5: Пример использования.

```
# Откроем таблицу, создадим на ее базе слой и добавим в карту
table_world = provider_manager.openfile(filepath)
world = Layer.create(table_world)
map = Map([ world ])
# Для полученной карты создадим окно просмотра
mapview = view_manager.create_mapview(map)
```

При желании непосредственно в окно карты можно поместить элементы управления. Рассмотрим пример добавления ползунка в левый верхний угол окна карты, который изменяет прозрачность верхнего слоя карты:

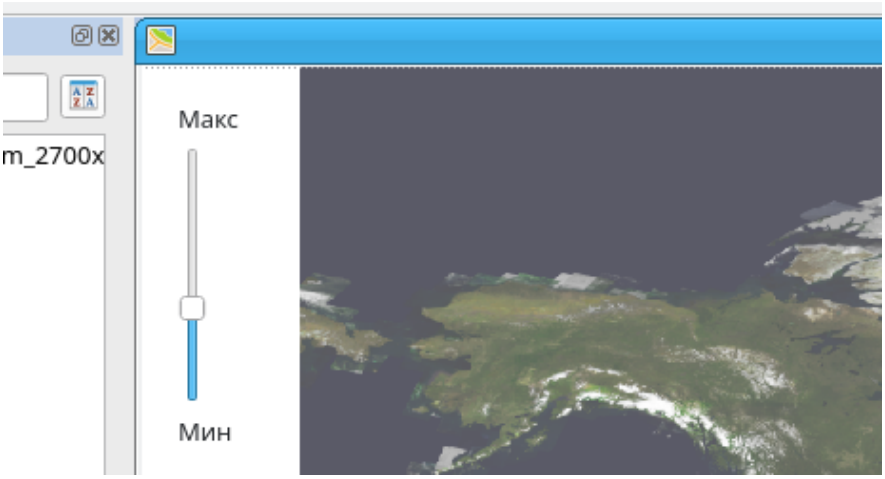
```
from axipy import view_manager
from PySide2.QtWidgets import QSlider, QFrame, QLabel, QVBoxLayout
from PySide2.QtCore import Qt
from PySide2.QtGui import QPalette

def change_opacity(v):
    mv = view_manager.active
    if mv and len(mv.map.layers):
        mv.map.layers[0].opacity = 100 - v

mv = view_manager.active
if mv is not None:
    frame = QFrame(mv.widget)
    layout = QVBoxLayout()
    slider = QSlider(Qt.Vertical, frame)
    slider.setRange(0, 100)
    slider.valueChanged.connect(change_opacity)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    layout.addWidget(QLabel("Макс", frame))
    layout.addWidget(slider)
    layout.addWidget(QLabel("Мин", frame))
    frame.setGeometry(10, 10, 50, 300)
    frame.setLayout(layout)
    frame.show()
```

Результат будет выглядеть примерно так:



10.6 Окно легенды для карты

Для просмотра и редактирования легенды `axipy.render.Legend` для карты необходимо использовать метод `axipy.gui.ViewManager.create_legendview()`, передав в него как параметр окно ранее созданной карты:

Список 6: Пример использования.

```
map = Map([ world ])
mapview = view_manager.create_mapview(map)
legendView = view_manager.create_legendview(mapview)
```

При этом будут помещены все доступные для просмотра легенды слоев карты map_view.

Легенды окна можно посмотреть следующим образом:

Список 7: Пример использования.

```
for legend in legendView.legends:
   print(legend.caption)
'''
>>> World
'''
```

10.7 Окно редактирования отчета

Если необходимости в визуальном редактировании отчета axipy.render.Report нет, а требуется только программно создать макет отчета и распечатать его на принтере или сохранить как PDF, то достаточно создать axipy.render.Report и работать с ним. В противном случае отчет, по аналогии с картой для визуального редактирования его так же помещают в окно редактирования axipy.gui.ReportView. В качестве примера создадим окно отчета и поместим в него геометрию и карту. Стоит заметить, что карту так-же можно использовать у уже открытого окна карты axipy.gui.MapView.map(). Координаты элементов задаются в единицах измерения отчета axipy.render.Report.unit. В нашем случае это миллиметры.

Список 8: Пример использования.

```
from PySide2.QtCore import Qt
rv = view_manager.create_reportview()

# Добавим полигон
geomItem = GeometryReportItem()
geomItem.geometry = Polygon((10,10), (10, 100), (100, 100), (10, 10))
geomItem.style = PolygonStyle(45, Qt.red)
rv.report.items.add(geomItem)

# Добавим направляющую
rv.x_guidelines.append(20)

# Текущий масштаб просмотра
rv.view_scale = 33

# Включение режима привязки координат
rv.snap_mode = True

# Размер ячейки сетки
rv.mesh_size = 20

# Откроем и добавим карту
table = provider_manager.openfile(filepath)
world = Layer.create(table)
map = Map([ world ])
mapItem = MapReportItem(Rect(10, 100, 200, 200), map)
mapItem.scale = 200000000
rv.report.items.add(mapItem)

# Поменяем стиль у первого объекта
rv.report.items[0].style = PolygonStyle(45, Qt.blue)
```

11

Пользовательский интерфейс

- Программное наполнение диалога
- Использование файла ресурсов *.ui

Для построения пользовательского интерфейса библиотека Qt, поставляемая в рамках ГИС Аксиома, предоставляет большой набор инструментария.

Для примера рассмотрим построение простейшего диалога. Целью данного примера является краткое описание того инструментария, который можно использовать для создания пользовательских форм и диалогов.

Для того, чтобы наш диалог обладал некоторыми специфичными свойствами, унаследуем его от стандартного базового класса PySide2.QtWidgets.QDialog и переопределим его поведение в соответствие с нашими потребностями. Первоначальный код будет выглядеть примерна так:

```
from axipy import *
from PySide2.QtWidgets import QDialog

class MyDialog(QDialog):
    pass

dialog = MyDialog(view_manager.global_parent)
dialog.resize(500,300)
dialog.exec()
```

Здесь мы создали пользовательский класс диалога и активировали его. Отметим, что свойство view_manager.global_parent, переданное в конструктор рассматривается как объект-владелец данного диалога. Если данное свойство не проставить, то диалог в панели задач будет показан как отдельное приложение.

Тестирование можно производить в рамках самой ГИС Аксиомы. Для этого необходимо открыть редактор. Кнопка запуска находится на панели «Консоль Python». Если она отсутствует в интерфейсе, то для включения необходимо в выпадающем меню кнопки «Панели» включить пункт «Консоль Python». Сохраним данный файл в файловой системе под именем mydialog.py.

Далее, чтобы разместить на данном диалоге элементы управления можно пойти двумя путями:

- Создать эти элементы программно

- Использовать для этого файл ресурсов *.ui

Второй вариант более понятен и удобен, но мы вкратце рассмотрим оба подхода.

11.1 Программное наполнение диалога

Создадим простой диалог, на который программно добавим кнопку и поле ввода. По нажатию на кнопку в консоль будет выведен текст элемента ввода.

```
from axipy import *
from PySide2.QtWidgets import QDialog, QPushButton, QLineEdit

class MyDialog(QDialog):
    def __init__(self, parent):
        super().__init__(parent)
        self.button = QPushButton(self)
        self.button.setText("Кнопка")
        self.button.move(150,50)
        # Реакция на нажатие кнопки
        self.button.clicked.connect(self.button_clicked)
        self.edit = QLineEdit(self)
        self.edit.setText("Текст")
        self.edit.move(10,50)

    def button_clicked(self) :
        print(self.edit.text())

dialog = MyDialog(view_manager.global_parent)
dialog.resize(500,300)

dialog.exec()
```

В данном примере элементы располагаются с явным указанием их места. Это не совсем удобно и в зависимости от размеров элементов управления на конкретной системе могут накладываться друг на друга. Или при изменении размеров самой формы могут вообще исчезать. Для решения этой проблемы обычно используются классы динамического размещения элементов на форме. В нашем случае это наследники класса `PySide2.QtWidgets.QLayout`. Модифицируем пример с использованием класса `PySide2.QtWidgets.QGridLayout`.

```
from axipy import *
from PySide2.QtWidgets import QDialog, QPushButton, QLineEdit, QGridLayout

class MyDialog(QDialog):
    def __init__(self, parent):
        super().__init__(parent)
        # создаем layout
        layout = QGridLayout()
        self.button = QPushButton()
        self.button.setText("Кнопка")
        self.button.clicked.connect(self.button_clicked)
        # Добавляем кнопку
        layout.addWidget(self.button, 0, 0, Qt.AlignTop)
        self.edit = QLineEdit()
        self.edit.setText("Текст")
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        # Добавляем элемент ввода
        layout.addWidget(self.edit, 0, 1, Qt.AlignTop)
        # Назначаем layout для диалога
        self.setLayout(layout)

    def button_clicked(self) :
        print(self.edit.text())

dialog = MyDialog(view_manager.global_parent)
dialog.resize(500,300)

dialog.exec()
```

Теперь наши элементы при изменении размеров диалога будут пропорционально менять свои размеры, прижимаясь к верхнему краю.

Если элементов на форме много, то такой способ размещения элементов достаточно трудоемок. Рассмотрим альтернативный способ размещения с использованием дизайнера Qt Designer.

11.2 Использование файла ресурсов *.ui

Для формирования UI представления формы нам понадобится инструмент Qt Designer. Он поставляется совместно со средствами разработки Qt, но его также можно установить отдельно как пакет python qt5_applications. Подробнее о инсталляции пакетов см в разделе Без интернета. Ручная установка пакетов.. Т.е. в конечном итоге в зависимости от окружения нам нужно выполнить команду:

```
python3 -m pip install --user qt5_applications
```

После успешной инсталляции в каталог site-packages из файлового менеджера переходим в каталог site-packages/qt5_applications/Qt/bin и запускаем на выполнение файл designer.

Выбираем тип диалога или формы. В нашем случае это Dialog without buttons. помещаем на него посредством перетаскивания кнопку `PySide2.QtWidgets.QPushButton` и элемент ввода `PySide2.QtWidgets.QLineEdit`. Меняем у кнопки ее текст на «Кнопка» (двойным щелчком на элементах) и для поля ввода заносим текст «Текст». По правой кнопке мыши можно поменять названия элементов на button и edit (чтобы они совпадали с предыдущими примерами). Далее, можно применить динамическое размещение элементов, выбрав на панели его тип. Для этого выберем мышью фарсу и нажмем кнопку на панели `Lay Out in a Grid`. Чтобы элементы прижать кверху, перетянем по аналогии с кнопкой и полем ввода элемент `Vertical Spacer` и отпустим его над нижней частью окна. После этого наши элементы должны с середины переместиться наверх. Сохраняем файл под именем mydialog.ui и кладем его рядом с файлом mydialog.py.

Далее, нам необходимо этот файл загрузить в диалог. Измененный пример с аналогичным функционалом будет выглядеть следующим образом:

```
from axipy import *
from PySide2.QtWidgets import QDialog, QPushButton, QLineEdit, QVBoxLayout
from PySide2.QtUiTools import QUiLoader
import os
```

(continues on next page)

(продолжение с предыдущей страницы)

```
class MyDialog(QDialog):
    def __init__(self , parent):
        super().__init__(parent)
        # Путь файла в файловой системе
        uiFile = os.path.join(os.path.dirname(__file__), "mydialog.ui")
        # Загружаем файл ui
        self.ui = QUiLoader().load(uiFile, parent)
        self.ui.button.clicked.connect(self.button_clicked)

    def button_clicked(self) :
        print(self.ui.edit.text())

    # Перенаправим метод show на self.ui
    def show(self):
        return self.ui.show()

# Показ диалога как немодальное окно. Для модального используем метод exec
dialog = MyDialog(view_manager.global_parent)
dialog.resize(500,300)
dialog.show()
```

Стоит заметить, что к элементам теперь необходимо обращаться не через self, а через self.ui. В примерах поставки ГИС аксиомы есть подобное решение с реализацией в виде модуля ru_axioma_gis_axipy_example_dialog.

12

Задачи и отображение прогресса

12.1 Задачи

Скрипты на python выполняются в основном потоке приложения или по другому в потоке интерфейса. Если операция будет выполняться слишком долго, то интерфейс «зависнет» и будет невозможно со стороны пользователя понять это ошибка или программа все-таки продолжает выполнять свою работу. Чтобы этого избежать длительные вычисления следует выносить в фоновые потоки. В потоке интерфейса во время выполнения фоновой задачи есть возможность показывать прогресс операции.

С помощью класса `AxipyAnyCallableTask` можно превратить любую пользовательскую функцию в задачу, либо, что еще проще, воспользоваться методом `axipy.concurrent.TaskManager.run_and_get()`:

Список 1: Пример использования.

```
def user_heavy_function(arg1: int, arg2: str):
    print(f"Переданные аргументы: {arg1}, {arg2} \n")
    return 1

spec = ProgressSpecification(
    description="Длительная операция",
    flags=ProgressGuiFlags.CANCELABLE,
    with_handler=False)
result = task_manager.run_and_get(spec, user_heavy_function, "Hi, ", "world!")
assert result == 1
```

Если объем кода в одной функции будет сильно увеличиваться или понадобится запускать новые задачи внутри одной базовой, тогда лучше воспользоваться наследованием от базового класса `AxipyTask` и переопределить метод `run`.

12.2 Представление прогресса операции

Для обмена информацией между выполняемой задачей и элементом, отображающим прогресс, используется класс `AxipyProgressHandler`. Типичный вариант использования выглядит следующим образом:

```
def user_heavy_function(ph: AxipyProgressHandler, count: int):
    # Вначале задаём верхнюю планку изменения прогресса
    ph.set_max_progress(count)
    for i in range(0, count):
        if ph.is_canceled():
            break
        # Тут делаем длительные вычисления
        ph.add_progress(1)
    return ph.progress()

spec = ProgressSpecification(
    description="Длительная операция",
    flags=ProgressGuiFlags.CANCELABLE)
times = 6
real_times = task_manager.run_and_get(spec, user_heavy_function, times)
# выводим колличество раз которое отработал цикл внутри
print(real_times)
```

`ProgressSpecification` - это вспомогательная структура данных, которая используется для первичной инициализации элемента, отображающего прогресс. Вместо `is_canceled` можно использовать `raise_if_canceled` если нужно выйти из нескольких вложенных вызовов функций или циклов.

12.3 Выполнение задач и многопоточность

Важно понимать, что задачи будут выполняться не в потоке интерфейса, поэтому внутри этих задач нельзя отображать никакие графические элементы (`QWidget`). Так же нужно внимательно следить за тем какие ресурсы могут использоваться несколькими потоками и при необходимости использовать различные механизмы синхронизации (мьютексы, локи и т.д.). Общее правило при работе с несколькими потоками следующее: старайтесь чтобы каждая задача содержала в себе все необходимые данные для выполнения. Синхронизацию, если она необходима, следует использовать только в момент получения результата. Это упростит код и сведет к минимум количество ошибок.

Переданные на выполнения задачи ставятся в общую очередь, которая распределяется между физическими ядрами процессора в порядке добавления. Поэтому нет гарантии, что переданная задача выполнится мгновенно, а так же то что группа задач будет выполняться именно в порядке добавления. Если задача предполагает долгое ожидание без интенсивных нагрузок на процессор, то лучше воспользоваться стандартными python потоками. Типичные примеры таких задач это загрузка ресурсов с диска или скачивание файлов по сети.

13

Модули (Плагины)

Модуль - это компонент, добавляющий определенный функционал в программу. Это позволяет программе быть расширяемой.

Данный раздел описывает требования и рекомендации по созданию таких компонентов для ГИС «Аксиома».

Наименования «Модуль» и «Плагин» обозначают одну сущность, воспринимаемую разными субъектами: модуль с точки зрения пользователя и плагин с точки зрения разработчика. Оба эти понятия могут встречаться в документации и их можно считать взаимозаменяемыми.

Чтобы создать модуль, руководствуйтесь следующим:

1. Придумать идею - функционал, решающий какую-то проблему.
2. Создать структуру плагина - файлы и папки.
3. Написать код.
4. Протестировать.
5. Опубликовать.

Совет: Изучайте исходный код готовых модулей.

13.1 Структура модуля

Модуль для ГИС «Аксиома» - это специально оформленный модуль Python с дополнительными файлами.

Рассмотрим структуру минимального модуля с обязательными параметрами и более расширенного:

Минимальный:

```
ru_mycompany_minimal_module  # папка с модулем
├── __init__.py  # точка входа
└── manifest.ini  # информация о модуле
```

Расширенный:

```
ru_mycompany_extended_module
├── documentation
│   └── index.html
├── business_logic.py
├── i18n
│   ├── translation_en.qm
│   └── translation_en.ts
├── __init__.py
├── manifest.ini
└── ui
    ├── form.ui
    ├── image.png
    └── logo.png
```

13.1.1 Идентификатор модуля

`ru_mycompany_minimal_module` - папка с модулем, она же - уникальный идентификатор модуля. Так же, как и при создании обычных модулей Python, избегайте конфликтов имен. Делайте имя модуля уникальным. Для этого следуйте простому соглашению именования: - используйте имя вашего веб-сайта или электронной почты, разделив на слова в обратном порядке; - используйте только маленькие латинские буквы и символ нижнего подчеркивания "_", т.е. `[a-z0-9_]`.

Так для модуля mymodule рекомендуемым идентификатором будет: - для веб-сайта axioma-gis.ru - `ru_axioma_gis_mymodule` - для почты andrey@yandex.ru - `ru_yandex_at_andrey_mymodule`

13.1.2 Точка входа

`__init__.py` - точка входа в модуль, так же как и для любого другого модуля Python - является обязательным для системы импорта. Содержит основной код модуля или импортирует другие локальные файлы.

См.также:

Точка входа может содержать специальный Класс Plugin.

13.1.3 Информация о модуле

`manifest.ini` - содержит основную информацию, версию, название и прочее. Является ini-файлом с простыми парами ключ=значение.

Примечание: Файл `manifest.ini` должен иметь кодировку UTF-8.

Минимальный пример содержимого:

```
name=Пример модуля
description=Короткий текст с описанием модуля.
```

См.также:

Для более подробного описания формата ini, поддерживаемого ГИС «Аксиома», смотрите документацию configparser.

Таблица 1: Таблица значений

Параметр	Обязательно	Описание
name	True	Короткая строка с именем модуля.
description	True	Короткий текст с описанием.
version	False	Строка с версией модуля.
homepage	False	Ссылка на домашнюю страницу модуля.
target	False	Версия Аксиомы, с которой работает модуль; цифры, разделенные точкой.
platforms	False	Список поддерживаемых платформ; через запятую из возможных Windows Linux и Darwin.
icon	False	Имя файла или относительный путь (относительно корневой папки модуля) в формате PNG.

Также могут содержаться другие необязательные параметры:

```
; может содержать коментарии
name=Пример модуля
description=Короткий текст с описанием модуля.
    Может быть многострочным.
; конец обязательных параметров

; необязательные параметры
version=1.0
homepage=https://dev.axioma-gis.ru
platforms=Windows,Darwin
target=3.1.0
author=Developer Name
email=dev@axioma-gis.ru
repository=https://github.com/developer/mymodule
license=New BSD
```

(continues on next page)

(продолжение с предыдущей страницы)

```
; секция локализации
[i18n]
name_en=Plugin example
description_en=Plugin example description.
```

13.2 Документация

Документация может быть написана в HTML файлах. ГИС «Аксиома» откроет документацию в системном веб-браузере. Аксиома ищет документацию в папке с модулем documentation - файлы `index[locale].html`. Пользователь откроет документацию с суффиксом локали, совпадающим с языком системы. При отсутствии совпадения будет открываться файл без суффикса - `index.html`.

13.3 Переводы

Можно предусмотреть загрузку модуля на разных языках.

Название и описание самого модуля может быть переведено на другие языки в манифесте в секции `i18n`:

```
; секция локализации
[i18n]
name_en=Plugin example
description_en=Plugin example description.
name_fr=Exemple de plugin
description_fr=Description de l'exemple de plugin.
```

У пользователя отобразится название и описание в случае, если язык системы будет совпадать с суффиксом локали. Иначе отобразится название и описание из основной секции.

Наиболее простой способ создания и сопровождения переводов строк из исходного кода - использование «Qt Linguist».

Примечание: Подробнее о переводе в документации Qt Linguist.

Основные этапы:

1. Отмечаются строки, предназначенные для перевода.
2. Для экспорта строк используется утилита `lupdate`. Она проходит по файлам с исходным кодом и забирает все встречаемые строки, отмеченные для перевода. Результатом является файл с расширением `.ts` - простой структурированный xml файл со строками.
3. `.ts` - файл открывается в «Qt Linguist» и переводится на один или более языков.

4. После завершения перевода отдельных строк файл `.ts` “компилируется” в бинарный файл с расширением `.qm`, который будет загружен Аксиомой.ГИС. Для компиляции используется утилита `lrelease`.

```
lrelease your_plugin.ts
```

5. `.qm` - файлы размещаются в подпапке `i18n` внутри модуля. Они будут загружены вместе с модулем.

13.4 Класс Plugin

Модули рекомендуется писать в объектном стиле. Для этого точка входа `__init__.py` должна содержать класс `Plugin`. Тогда ГИС «Аксиома» при загрузке модуля создаст его экземпляр, а при выгрузке - уничтожит его.

Вспомогательный класс `axipy.AxiomaPlugin` и его базовый класс `axipy.AxiomaInterface` содержат свойства и функции, необходимые почти для любого плагина. Например, загрузка/сохранение настроек `settings`, получение файлов внутри папки с модулем `local_file()`, перевод строк `tr()`, добавление кнопок в интерфейс `create_action` и другое.

Пример модуля `__init__.py`

```
"""Пример добавления кнопки и подключение действия по нажатию на нее
(показ сообщения). При выгрузке кнопка удаляется из интерфейса.
"""
from PySide2.QtWidgets import QMessageBox
from axipy import AxiomaPlugin


class Plugin(AxiomaPlugin):
    def load(self):
        self.__action = self.create_action('Пример действия',
                icon=':/icons/share/32px/run.png', on_click=self.show_message)
        position = self.get_position('Основные', 'Команды')
        position.add(self.__action)

    def unload(self):
        self.__action.remove()

    def show_message(self):
        QMessageBox.information(None, 'Сообщение',
                'Пример выполнения действия по нажатию кнопки')
```

- При загрузке Аксиома создаст экземпляр модуля и вызовет метод `load()`.
- `unload()` - вызывается, когда модуль выгружается.

Важно: При наследовании от `axipy.AxiomaPlugin` не используйте методы базового класса в конструкторе `__init__`. Помещайте логику инициализации в метод `load()`. Это необходимо для правильной работы базового класса.

Внимание: Не рекомендуется, начиная с версии 3.5.

Инициализация модуля через «внедрение зависимостей» продолжает работать, но **не рекомендуется** в пользу наследования (описан выше). При внедрении зависимостей класс `Plugin` не наследуется ни от чего, а в конструктор принимается параметр `iface` - экземпляр класса `axipy.AxiomaInterface`.

```
class Plugin:
    def __init__(self, iface):
        self.__action = iface.menubar.create_button('Пример действия',
                icon=':://icons/share/32px/run.png', on_click=self.show_message)
        ...
```

13.5 Архив

Физически модуль представлен в виде папки с уникальным именем, внутри которой расположены файлы и папки с бизнес-логикой, конфигурациями, документацией, зависимостями, графическими формами и прочим. Для гарантии целостности и удобства распространения готовые модули помещаются в архив.

Архив использует формат ZIP и имеет следующую структуру:

```
my_plugin_archive_v1.axp
└── ru_axioma_gis_axipy_example_plugin_from_package
    ├── __init__.py
    └── manifest.ini
```

Таким образом архив просто содержит папку с модулем. Имя архива может быть любым и должно заканчиваться на `.axp`, в то время как имя папки с модулем должно быть уникальным.

Для создания архива достаточно запаковать модуль в ZIP любым поддерживаемым архиватором и указать расширение выходного файла как `.axp` вместо стандартного `.zip`.

С помощью архиватора можно проверить целостность архива, например:

```
zip -Tv my_plugin_archive_v1.axp
```

Результат проверки:

```
Archive:  my_plugin_archive_v1.axp
    testing: ru_axioma_gis_axipy_example_plugin_from_package/   OK
    testing: ru_axioma_gis_axipy_example_plugin_from_package/__init__.py   OK
    testing: ru_axioma_gis_axipy_example_plugin_from_package/manifest.ini   OK
No errors detected in compressed data of my_plugin_archive_v1.axp
test of my_plugin_archive_v1.axp OK
```

Архив с модулем распаковывается в пользовательскую директорию при установке. Архив может быть установлен пользователем через интерфейс программы ГИС «Аксиома» в диалоге «Модули». Все модули, установленные пользователем, могут быть удалены из того же диалога.

Физически модули устанавливаются в installed_modules в пользовательскую папку. Расположение пользовательской папки зависит от операционной системы:

- для Windows – «%APPDATA%ESTIAxioma.GIS»
- для Linux – «$HOME/.local/share/ESTI/Axioma.GIS/»
- для macOS – «$HOME/Library/Application Support/ESTI/Axioma.GIS/»

13.6 Зависимости

Модули Аксиома могут использовать сторонние библиотеки Python. Такой модуль во время установки обратится к каталогу пакетов PyPI и установит необходимые зависимости.

В коде модуля пакет импортируется обычным способом:

Список 1: Файл __init__.py

```
import numpy
...
```

Зависимости перечисляются в специальном файле `requirements.txt`. Так модуль с зависимостями может иметь следующую структуру:

```
ru_axioma_gis_axipy_example_plugin_from_package
├── __init__.py
├── manifest.ini
└── requirements.txt
```

Файл является простым текстовым файлом в кодировке `UTF-8`, в котором построчно перечислены необходимые пакеты. Например:

Список 2: Файл requirements.txt

```
numpy
requests
idna
```

Примечание: В настоящий момент нет возможности указать версию пакета.

13.6.1 Библиотека matplotlib

Matplotlib - одна из популярных библиотек для визуализации данных на языке Python. Она может быть использована с PySide2, axipy и ГИС Аксиома.

Однако, из-за особенностей поиска доступных реализаций, matplotlib пытается использовать PyQt5. PyQt5 присутствует в ГИС Аксиома потому, что старое API axioma построено на нем.

Чтобы заставить matplotlib использовать PySide2, можно временно спрятать модуль PyQt5. Тогда импорт matplotlib пройдет без ошибок.

Список 3: Импорт matplotlib с использованием PySide2

```
hidepyqt = sys.modules.pop('PyQt5.QtCore', None)
import matplotlib.backends.backend_qt5agg
if hidepyqt:
    sys.modules['PyQt5.QtCore'] = hidepyqt
```

13.6.2 Без интернета. Ручная установка пакетов.

Может возникнуть ситуация, когда устанавливаемый модуль имеет внешние зависимости, а доступа к каталогу пакетов PyPI нет. Модуль не сможет успешно установиться.

В этом случае можно скачать все необходимые зависимые пакеты на компьютере, который имеет доступ к интернету и конкретно к каталогу пакетов PyPI, а затем перенести их на целевой компьютер.

Для этого рекомендуется:

1. Установить ГИС Аксиома на компьютер с доступом к сети Интернет (той же версии и платформы).
2. Извлечь из архива с модулем `.axp` файл зависимостей `requirements.txt`.
3. Используя командную строку и интерпретатор Python внутри установленной Аксиомы, выполнить

Список 4: Загрузка необходимых пакетов в папку

```
python -m pip download -r requirements.txt --dest ./module_deps/
```

, где `module_deps` - папка, в которую будут загружены зависимые пакеты.

4. Перенести зависимые пакеты на компьютер без интернета.
5. Аналогично, используя командную строку и интерпретатор Python внутри установленной Аксиомы, выполнить

Список 5: Установка необходимых пакетов из папки

```
python -m pip install -r requirements.txt --user --no-index --find-links ./module_
↪deps/
```

6. Установить сам модуль `*.axp`.

Определить каталог, куда будут устанавливаться зависимые python пакеты можно следующей командой:

```
python -m site --user-site
```

Результат:

```
C:\Users\user\AppData\Roaming\Python\Python37\site-packages
```

Для ориентировки, примерное расположение в зависимости от системы будет следующее:

13.6.2.1 Расположение каталога site-packages

- для Windows – «%APPDATA%pythonpython<VERSION>site-packages»
- для Linux – «$HOME/.local/lib/python<VERSION>/site-packages»
- для macOS – «$HOME/Library/Python/<VERSION>/lib/python/site-packages»

Можно использовать для этих целей командный файл.

Пример скрипта для linux (install_requirement), где в качестве параметра можно передать имя файла с набором пакетов. Если параметр опущен, берется файл из текущего каталога requirements.txt:

```
#!/bin/bash

AXIOMA_BASE=/opt/Axioma.GIS

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:${AXIOMA_BASE}/bin

if [ -z "$1" ]; then
 filename="requirements.txt"
else
 filename="$1"
fi

${AXIOMA_BASE}/python/bin/python3 -m pip install --user -r  ${filename}
```

Пример команды для Windows (выполняется в консоли cmd.exe):

```
"C:/Program Files/Axioma v4/bin/python/python.exe" -m pip install --user -r␣
↪requirements.txt
```

Если есть необходимость установить отдельный пакет по имени, то команда будет выглядеть следующим образом:

```
"C:/Program Files/Axioma v4/bin/python/python.exe" -m pip install --user qt5_
↪applications
```

где qt5_applications наименование пакета, который необходимо установить.

14 Среда разработки

Для разработки с использованием **axipy** необязательно использовать никаких дополнительных инструментов. Достаточно любого простого текстового редактора типа Блокнот для написания кода. Также в саму ГИС Аксиома встроен редактор кода с базовым функционалом.

Однако иногда полезно использовать профессиональные интегрированные среды разработки (Integrated development environment - IDE), например, если вы разрабатываете большой продукт/инструмент на основе ГИС Аксиома. Интегрированные среды разработки существенно облегчают разработку, предоставляя полезные возможности, такие как: подсветка кода, автодополнение, отладка, система контроля версий и другие.

Примечание: Необязательно использовать IDE. Достаточно любого текстового редактора.

14.1 Настройка

Существует множество сред разработки для Python. При разработке с использованием API для ГИС Аксиома можно пользоваться любыми понравившимися инструментами. Одной из популярных сред разработки является Visual Studio Code. На ее примере разберем настройку IDE для разработки модулей. Настройка любой другой IDE будет схожей.

Примечание: Используйте любые понравившиеся инструменты.

Visual Studio Code - кроссплатформенная среда разработки с бесплатной лицензией. Ее можно скачать с официального сайта: Visual Studio Code.

Перед началом в установленной **Visual Studio Code** необходимо установить расширение Python. Это можно сделать во вкладке Extensions (Ctrl+Shift+X).

Создадим папку с проектом `axioma_hello_world`.

Хорошей практикой при разработке на языке Python является создание виртуального окружения virtualenv. В командной строке (Ctrl+`) Visual Studio Code выполним:

```
"C:\Program Files\Axioma v4\bin\python\python.exe" -m venv .venv --system-site-
↪packages
```

Примечание: Расположение установленной ГИС Аксиома может отличаться.

Важно: Обязательно использовать Python, поставляемый вместе с ГИС Аксиома.

Создастся виртуальное окружение в папке `.venv`. Visual Studio Code сразу найдет эту папку и предложит использовать ее окружение для текущего проекта.

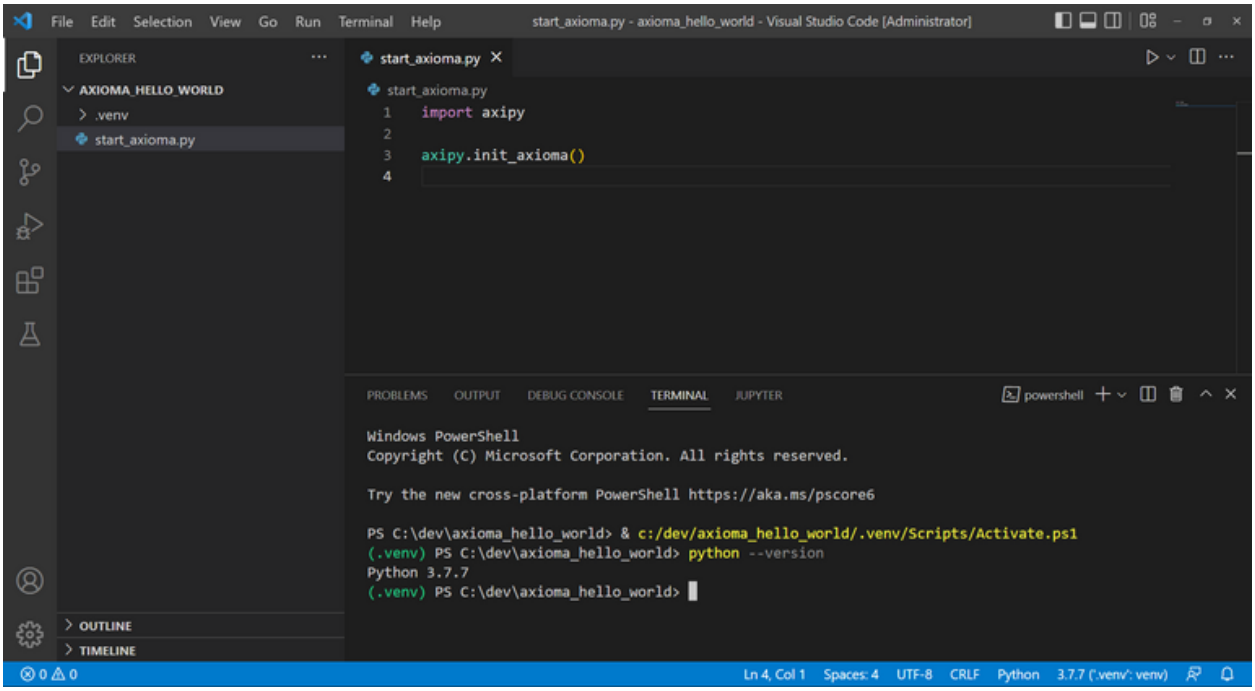
Чтобы убедиться, что окружение работает корректно, выведем версию интерпретатора Python; создадим простой стартовый скрипт и увидим, что ГИС Аксиома запускается из среды разработки:

Список 1: Содержимое файла start_axioma.py

```
import axipy
from axipy.app import MainWindow

app = axipy.init_axioma()
MainWindow.show()
app.exec_()
```

Список 2: Версия интерпретатора Python.

```
python --version
```



Создадим простой модуль:

```
axioma_hello_world  # папка с проектом
├── ru_axioma_gis_axipy_example_plugin_minimal  # модуль
│   ├── __init__.py
│   └── manifest.ini
└── start_axioma.py  # стартовый скрипт
```

См.также:

Раздел Модули.

Совет: Готовый скрипт `start_axioma.py` и пример плагина `ru_axioma_gis_axipy_example_plugin_minimal` можно найти в папке с установленной ГИС Аксиома.

Список 3: Содержимое файла manifest.ini

```
name=Пример модуля axipy - Минимальный
description=Модуль для демонстрации возможностей разработки на Python
```

Список 4: Содержимое файла __init__.py

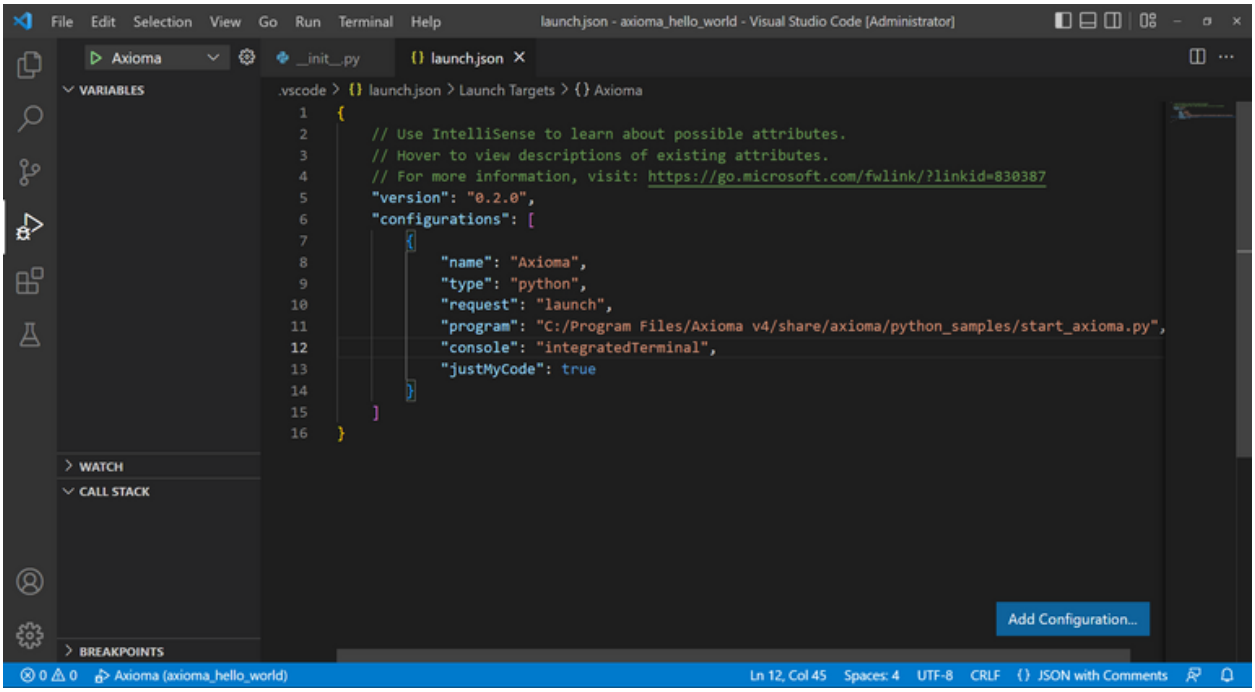
```
from PySide2.QtWidgets import QMessageBox
from axipy import AxiomaPlugin, Position


class Plugin(AxiomaPlugin):
    def load(self):
        self.__action = self.create_action('Пример действия',
                icon=':/icons/share/32px/run3.png', on_click=self.show_message)
        position = Position('Основные', 'Команды')
        position.add(self.__action)
        self.__action.action.setToolTip('Всплывающая подсказка')

    def unload(self):
        self.__action.remove()

    def show_message(self):
        QMessageBox.information(None, 'Сообщение',
                'Пример выполнения действия по нажатию кнопки')
```

Запустим ГИС Аксиома из скрипта `start_axioma.py` и в диалоге Модули -> Настройки добавим путь к нашему проекту с модулем. Найдем его в списке найденных модулей и загрузим (можно сравнить действительное расположение модуля). Так образом можно продолжать разрабатывать этот модуль, выгружая и загружая его вновь после внесения изменений.

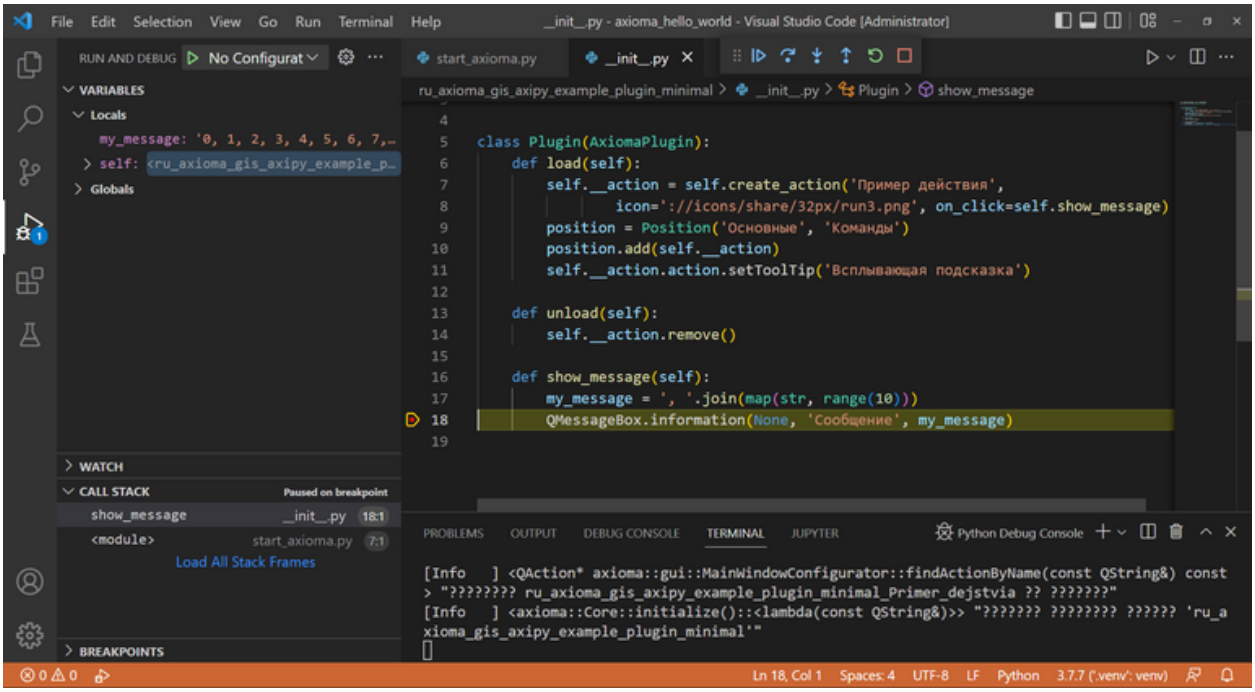
Важно: Чтобы изменения вступили в силу, необходимо выгрузить и повторно загрузить модуль.

14.2 Отладка

Проект готов для того, чтобы отлаживать модуль, ставя точки останова в коде. Для удобства можно создать конфигурацию для отладки, которая будет сама запускать стартовый скрипт. Создайте конфигурацию для отладки - Visual Studio Code создаст файл `launch.json`. Стартовый скрипт есть в папке с установленной ГИС Аксиома. Можно указать путь сразу к нему, и убрать из проекта с модулем.



Теперь при запуске этой конфигурации можно отлаживать модуль, ставя точки останова, анализируя стек вызовов и содержимое переменных.


File Edit Selection View Go Run Terminal Help
__init__.py - axioma_hello_world - Visual Studio Code [Administrator]
RUN AND DEBUG
No Configurat
start_axioma.py
__init__.py
ru_axioma_gis_axipy_example_plugin_minimal > __init__.py > Plugin > show_message
VARIABLES
Locals
my_message: '0, 1, 2, 3, 4, 5, 6, 7,...
self: <ru_axioma_gis_axipy_example_p...
Globals
WATCH
CALL STACK
Paused on breakpoint
show_message __init__.py 18:1
<module> start_axioma.py 7:1
Load All Stack Frames
BREAKPOINTS
class Plugin(AxiomaPlugin):
def load(self):
self.__action = self.create_action('Пример действия',
icon=':/icons/share/32px/run3.png', on_click=self.show_message)
position = Position('Основные', 'Команды')
position.add(self.__action)
self.__action.action.setToolTip('Всплывающая подсказка')
def unload(self):
self.__action.remove()
def show_message(self):
my_message = ', '.join(map(str, range(10)))
QMessageBox.information(None, 'Сообщение', my_message)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
Python Debug Console
[Info] <QAction* axioma::gui::MainWindowConfigurator::findActionByName(const QString&) const > "???????? ru_axioma_gis_axipy_example_plugin_minimal_Primer_dejstvia ?? ????????"
[Info] <axioma::Core::initialize()::<lambda(const QString&)>> "??????? ???????? ?????? 'ru_axioma_gis_axipy_example_plugin_minimal'"
Ln 18, Col 1 Spaces: 4 UTF-8 LF Python 3.7.7 ('.venv': venv)

15

История изменений

15.1 4.4 Изменения

Февраль 2023

15.1.1 Исправления

- Методы `axipy.da.CollectionStyle.point()`, `axipy.da.CollectionStyle.line()`, `axipy.da.CollectionStyle.polygon()`, `axipy.da.CollectionStyle.text()` реализованы как свойства.

15.2 4.3 Изменения

Декабрь 2022

15.2.1 Новое

- Модули устанавливаются в папку `installed_modules`.
- Явное указание секции `[general]` в файле метаданных модуля `manifest.ini` необязательно.
- Модули с зависимостями.
- Обход ошибки при импорте библиотеки Matplotlib.
- Раздел Среда разработки.
- Методы `select_by_mouse()` и `select_by_rect()` для выделения геометрий как в инструменте Выбор.
- Доступ к объектам данных в `axipy.da.DataManager` по имени как в словаре `dict`.
- Редактируемый атрибут `axipy.da.Table.schema`.
- Свойства подписей.

- Список загруженных провайдеров axipy.da.ProviderManager.providers().

15.2.2 Исправления

- При выполнении конвертации axipy.da.MifMidDataProvider.convert_to_tab() терялись пространственные данные.
- Из-за похожести с axipy.gui.MapView.device_rect свойство axipy.gui.View.rect переименовано в axipy.gui.View.position.
- axipy.gui.Map.to_image() не учитывал ограничивающий прямоугольник.
- Метод axipy.da.Geometry.from_json() переименован в axipy.da.Geometry.from_geojson().
- Класс axipy.utl.Printer переименован в axipy.utl.FloatFormatter.
- Для axipy.da.DataManager.sql_dialect сменен тип данных с str на enum.
- Свойство axipy.render.Label.placementPolicy вынесено как enum axipy.render.LabelOverlap.

15.3 4.0 Изменения

Июнь 2022

15.3.1 Новое

- Метод создания и показа главного окна axipy.app.MainWindow.show().

15.3.2 Исправления

- Свойство axipy.gui.ReportView.scale() переименовано в axipy.gui.ReportView.view_scale.

15.4 3.7.0 Изменения

Март 2022

15.4.1 Новое

- Методы load()/unload() класса axipy.gui.MapTool; метод axipy.gui.MapTool.deactivate() отмечен как устаревший.
- Метод axipy.gui.MapTool.canDeactivate() переименован в axipy.gui.MapTool.canUnload().
- Функция поиска перевода axipy.tr().

15.4.2 Исправления

- Изменены пределы для свойств axipy.render.RasterLayer.brightness и axipy.render.RasterLayer.contrast на диапазон (-100...100).

15.5 3.5.0 Изменения

Август 2021

15.5.1 Новое

- Новые вспомогательные методы в axipy.gui.MapTool.
- Объектно-ориентированный стиль создания кнопок axipy.menubar.Button.
- Механизм слежения за значениями axipy.da.state_manager.
- Распространение модулей в архивах.
- Объявление модулей с наследованием от axipy.AxiomaPlugin.
- Каталог данных содержит таблицу выборки axipy.da.DataCatalog.selection.
- Менеджер для запуска и управления пользовательскими задачами axipy.concurrent.TaskManager.
- Добавлена панель активного инструмента axipy.gui.ActiveToolPanel в которую можно поместить графический элемент упрощающий работу с пользовательским инструментом.

15.5.2 Исправления

- Класс axipy.da.Collection переименован в axipy.da.GeometryCollection.
- Методы axipy.da.DataCatalog.tables(), axipy.da.DataCatalog.objects(), axipy.da.DataCatalog.count() реализованы как свойства. Метод axipy.da.Schema.attribute_names() так-же переделан как свойство.
- Убраны класс axipy.cs.UnitService и его экземпляр axipy.cs.unit. Их функционал перенесен в базовый класс axipy.cs.EarthUnit, который переименован в axipy.cs.Unit. Переименованы методы axipy.cs.LinearUnit.list_all(), axipy.cs.AreaUnit.list_all().
- Переименован класс axipy.da.DataCatalog в axipy.da.DataManager

- Переименован класс axipy.gui.ViewService в axipy.gui.ViewManager
- Переименован класс axipy.gui.SelectionService в axipy.gui.SelectionManager
- Переименован класс axipy.da.DataProviders в axipy.da.ProviderManager
- Экземпляр класса axipy.render.Map axipy.render.Map.unit перенесен в класс axipy.gui.MapView.

15.6 3.0.0 Изменения

Апрель 2021

15.6.1 Новое

- Руководство разработчика объединено со справочником функций.
- Свойство временной таблицы axipy.da.Table.is_temporary.
- Менеджер контекста with для axipy.da.DataObject.
- Транзакционная модель редактирования таблиц: axipy.da.Table.restore(), axipy.da.Table.commit(), axipy.da.Table.is_modified, axipy.da.Table.insert(), axipy.da.Table.update(), axipy.da.Table.delete().
- Каталог объектов данных axipy.app.MainWindow.catalog по умолчанию. Открываемые объекты данных автоматически попадают в каталог главного окна. Запросы axipy.da.DataCatalog.query() производятся к этому каталогу без явного указания конкретных таблиц.
- Создаваемые окна axipy.gui.ViewService.create_view() автоматически добавляются в главное окно программы.
- Настройки ГИС «Аксиома» axipy.Settings.
- Провайдеры данных axipy.da.DataProviders со специализированными параметрами для открытия/создания и импорта/экспорта: tab, shp и другие.
- Раздельные типы стилей: axipy.da.PointStyle, axipy.da.PolygonStyle и другие.
- Раздельные типы геометрий: axipy.da.Point, axipy.da.Polygon и другие.
- Загрузка/сохранение рабочих наборов axipy.app.MainWindow.load_workspace(), axipy.app.MainWindow.save_workspace().
- Удаление кнопок axipy.menubar.remove() приводит к удалению групп и вкладок axipy.menubar.Position, если они стали пустыми.

15.6.2 Исправления

- Ошибка при попытке закрытия временной таблицы с изменениями.
- Ошибка при задании разделителя в формате CSV `axipy.da.CsvDataProvider`.

15.7 2.9.0 Изменения

Декабрь 2020

15.7.1 Новое

- Первоначальный релиз.

16

Справочник функций

Справочник описывает модули и функции основного модуля **axipy** - нового API для ГИС «Аксиома» на языке Python.

Примечание: Не путать с модулем **axioma**; документация к нему расположена в другом месте.

16.1 Модули ГИС «Аксиома»

16.1.1 axipy

Основной пакет API для взаимодействия с ГИС Аксиома.

Предоставляет доступ к Аксиоме.ГИС через набор модулей, подмодулей, классов и функций.

`axipy.`**`io`**

Объект открытия/создания объектов данных.

> **Внимание:** Устарел в версии 3.0.0. Используйте готовый экземпляр `axipy.da.provider_manager`.

Type axipy.da.ProviderManager

16.1.1.1 AxiomaInterface - Интерфейс модуля

class axipy.**AxiomaInterface**

Интерфейс для модуля.

Вспомогательный класс для создания модулей.

См.также:

Подробнее в главе Модули (Плагины).

Methods:

active_tool_panel()	Возвращает экземпляр панели активного инструмента.
local_file(*paths)	Возвращает путь к файлу/папке относительно модуля.
tr(text)	Ищет перевод строки.
window()	Возвращает главное окно ГИС Аксиома.

Attributes:

catalog	Хранилище объектов данных.
io	Класс открытия/создания объектов данных.
language	Значение языка, с которым запущено приложение.
menubar	Объект с функциями меню главного окна ГИС Аксиома.
notifications	Отправление уведомлений в виде всплывающего окна.
settings	Настройки модуля.

active_tool_panel()

Возвращает экземпляр панели активного инструмента.

Тип результата ActiveToolPanel

Результат Менеджер для управления панелью активного инструмента.

property catalog

Хранилище объектов данных.

Тип результата DataManager

property io

Класс открытия/создания объектов данных.

Тип результата ProviderManager

property language

Значение языка, с которым запущено приложение.

Тип результата str

local_file(*paths)

Возвращает путь к файлу/папке относительно модуля.

Параметры ***path** – Составные относительного пути.

Тип результата str

Результат Абсолютный путь.

Пример:

```
plugin_path = iface.local_file()
icon_path = iface.local_file('images', '32px', 'logo.png')
```

property menubar

Объект с функциями меню главного окна ГИС Аксиома.

См.также:

axipy.menubar

property notifications

Отправление уведомлений в виде всплывающего окна.

Type Notifications

property settings

Настройки модуля.

Позволяет сохранять и загружать параметры.

См.также:

Подробнее в документации на класс PySide2.QtCore.QSettings.

Тип результата QSettings

tr(text)

Ищет перевод строки.

Производит поиск строки в загруженных файлах перевода.

Параметры **text** (str) – Строка для перевода.

Тип результата str

Результат Перевод стоки, если строка найдена. Иначе - сама переданная строка.

Пример:

```
button_name = iface.tr('My button')
```

window()

Возвращает главное окно ГИС Аксиома.

Тип результата QMainWindow

16.1.1.2 AxiomaPlugin - Модуль ГИС «Аксиома»

class axipy.**AxiomaPlugin**

Базовые классы: `axipy.interface.AxiomaInterface`

Модуль для ГИС Аксиома.

Содержит вспомогательные функции и свойства, которые могут быть использованы при реализации пользовательского модуля.

Примечание: Не переопределяйте конструктор. Переопределяйте метод `load()`.

См.также:

Подробнее в главе Модули (Плагины).

Methods:

`active_tool_panel`()	Возвращает экземпляр панели активного инструмента.
`create_action`(title, on_click[, icon, ...])	Создает кнопку с действием.
`create_separator`()	Создает разделитель.
`create_tool`(title, on_click[, icon, ...])	Создает кнопку с инструментом.
`get_position`(tab, group)	Возвращает положение в меню.
`load`()	Загружает модуль.
`local_file`(*paths)	Возвращает путь к файлу/папке относительно модуля.
`tr`(text)	Ищет перевод строки.
`unload`()	Выгружает модуль.
`user_plugin_data_dir`([file_name])	Возвращает каталог, в котором находится изменяемые данные модуля.
`user_plugin_dir`([file_name])	Возвращает каталог данного модуля.
`window`()	Возвращает главное окно ГИС Аксиома.

Attributes:

`catalog`	Хранилище объектов данных.
`io`	Класс открытия/создания объектов данных.
`language`	Значение языка, с которым запущено приложение.
`menubar`	Объект с функциями меню главного окна ГИС Аксиома.
`notifications`	Отправление уведомлений в виде всплывающего окна.
`settings`	Настройки модуля.

active_tool_panel()

Возвращает экземпляр панели активного инструмента.

Тип результата `ActiveToolPanel`

Результат Менеджер для управления панелью активного инструмента.

property catalog
Хранилище объектов данных.

Тип результата DataManager

create_action(title, on_click, icon='', enable_on=None, tooltip=None, doc_file=None)
Создает кнопку с действием.

Параметры

- **title** (str) – Текст.
- **on_click** (Callable[[], Any]) – Действие на нажатие.
- **icon** (Union[str, QIcon]) – Иконка. Может быть путем к файлу или адресом ресурса.
- **enable_on** (Union[str, DefaultKeys, None]) – Идентификатор наблюдателя для определения доступности кнопки.
- **tooltip** (Optional[str]) – Строка с дополнительной короткой информацией по данному действию.
- **doc_file** (Optional[str]) – Относительная ссылка на файл документации. Расположение рассматривается по отношению к каталогу documentation.

Тип результата ActionButton

Результат Кнопка с действием.

См.также:

axipy.da.StateManager.

Примечание: То же, что и axipy.menubar.ActionButton, но дополнительно делает идентификатор кнопки уникальным для данного модуля.

---

create_separator()
Создает разделитель.

Тип результата Separator

create_tool(title, on_click, icon='', enable_on=None, tooltip=None, doc_file=None)
Создает кнопку с инструментом.

Параметры

- **title** (str) – Текст.
- **on_click** (Callable[[], MapTool]) – Класс инструмента.
- **icon** (Union[str, QIcon]) – Иконка. Может быть путем к файлу или адресом ресурса.
- **enable_on** (Union[str, DefaultKeys, None]) – Идентификатор наблюдателя для определения доступности кнопки.
- **tooltip** (Optional[str]) – Строка с дополнительной короткой информацией по данному действию.

- **doc_file** (Optional[str]) – Относительная ссылка на файл документации. Расположение рассматривается по отношению к каталогу documentation.

Тип результата ToolButton

Результат Кнопка с инструментом.

См.также:

class:axipy.da.StateManager.

Примечание: То же, что и axipy.menubar.ToolButton, но дополнительно делает идентификатор кнопки уникальным для данного модуля.

---

get_position(tab, group)

Возвращает положение в меню. Может заранее не существовать.

Параметры

- **tab** (str) – Название вкладки.
- **group** (str) – Название группы.

Тип результата Position

Результат Положение для кнопки.

Примечание: Дублирует axipy.menubar.Position.

---

property io

Класс открытия/создания объектов данных.

Тип результата ProviderManager

property language

Значение языка, с которым запущено приложение.

Тип результата str

load()

Загружает модуль.

Переопределяйте этот метод для задания логики модуля.

local_file(*paths)

Возвращает путь к файлу/папке относительно модуля.

Параметры ***path** – Составные относительного пути.

Тип результата str

Результат Абсолютный путь.

Пример:

```
plugin_path = iface.local_file()
icon_path = iface.local_file('images', '32px', 'logo.png')
```

property menubar

Объект с функциями меню главного окна ГИС Аксиома.

См.также:

axipy.menubar

property notifications
Отправление уведомлений в виде всплывающего окна.

Type Notifications

property settings
Настройки модуля.

Позволяет сохранять и загружать параметры.

См.также:

Подробнее в документации на класс PySide2.QtCore.QSettings.

Тип результата QSettings

tr(text)
Ищет перевод строки.

Производит поиск строки в загруженных файлах перевода.

Параметры text (str) – Строка для перевода.

Тип результата str

Результат Перевод стоки, если строка найдена. Иначе - сама переданная строка.

Пример:

```
button_name = iface.tr('My button')
```

unload()
Выгружает модуль.

Переопределяйте этот метод для очистки ресурсов.

user_plugin_data_dir(file_name='')
Возвращает каталог, в котором находится изменяемые данные модуля. Расположение определяется в подкаталоге installed_plugins_data, расположенном на один уровень вверх по отношению к каталогу самого модуля `plugin_dir()`.

Параметры file_name (str) – Если параметр указан, возвращается полный путь файла в данном каталоге.

Тип результата str

user_plugin_dir(file_name='')
Возвращает каталог данного модуля.

Параметры file_name (str) – Если параметр указан, возвращается полный путь файла в каталоге модуля.

Тип результата str

window()
Возвращает главное окно ГИС Аксиома.

Тип результата QMainWindow

16.1.1.3 Settings - Настройки ГИС «Аксиома»

class axipy.**Settings**

Настройки ГИС Аксиома.

Класс не требует создания объекта. Используйте методы класса.

Список 1: Пример использования

```
# Читает значение
val = Settings.value(Settings.RulerColorLine)
# Записывает значение
new_value = QColor(0, 255, 0)
Settings.setValue(Settings.RulerColorLine, new_value)
# Сбрасывает на значение по умолчанию
Settings.reset(Settings.RulerColorLine)
```

Таблица 5: Атрибуты

Значение	Тип	Наименование
SilentCloseWidget	bool	Подтверждать закрытие несохраненных данных
SnapSensitiveRasius	int	Привязка узлов - размер
SnapColor	QColor	Привязка узлов - цвет
SnapThickness	int	Привязка узлов - толщина линии
EditNodeColor	QColor	Узлы при редактировании - цвет
EditNodeSize	int	Узлы при редактировании - размер
NearlyGeometriesTopology	bool	Перемещать узлы соседних объектов при редактировании
NodesUpdateMode	bool	Использовать перезапись истории изменений при редакти
ShowDrawingToolTip	bool	Показывать данные при рисовании
CreateTabAfterOpen	bool	Создавать TAB при открытии
RenameDataObjectFromTab	bool	Переименовывать открытый объект по имени TAB файла
LastSavePath	str	Последний пусть сохранения
UseLastSelectedFilter	bool	Запоминать последний фильтр в диалоге открытия файлов
SelectByInformationTool	bool	Инструмент «Информация» выбирает объект
SaveAsToOriginalFileFolder	bool	Сохранять копию в каталог с исходным файлом
LastNameFilter	str	Последний использованный фильтр файлов
SensitiveMouse	bool	Чувствительность мыши
ShowSplashScreen	bool	Отображать экран загрузки
RulerModeSpherical	bool	Линейка - измерение на сфере
RulerColorLine	bool	Линейка - цвет линии
UseAntialiasing	bool	Использовать сглаживание при отрисовке
ShowDegreeTypeNumeric	bool	Отображать градусы в формате Десятичное значение
DrawCoordSysBounds	bool	Отображать границы мира
PreserveScaleMap	bool	Сохранять масштаб при изменении размеров окна
ShowMapScaleBar	bool	Показывать масштабную линейку
ShowScrollOnMapView	bool	Показывать полосы прокрутки
LoadLastWorkspace	bool	Загружать при старте последнее рабочее пространство
ShowMeshLayout	bool	Отображать сетку привязки
MeshSizeLayout	float	Размер ячейки
SnapToMeshLayout	bool	Привязывать элементы отчета к сетке
ShowMeshLegend	bool	Отображать сетку привязки
MeshSizeLegend	float	Размер ячейки
SnapToMeshLegend	bool	Привязывать к сетке

continue

Таблица 5 – продолжение с предыдущей страницы

Значение	Тип	Наименование
LastOpenPath	str	Последний каталог откуда открывались данные
LastPathWorkspace	str	Последний каталог к рабочему набору
DefaultPathCache	str	Каталог с кэшированными данными
UserDataPaths	list[str]	Список пользовательских каталогов с названиями
EnableSmartTabs	bool	Умное переключение вкладок
DistancePrecision	int	Точность по умолчанию для расстояний и площадей

Methods:

reset(key)	Устанавливает значение по умолчанию.
setValue(key, value)	Устанавливает значение параметра.
value(key)	Читает значение параметра.

classmethod reset(key)
Устанавливает значение по умолчанию.

Параметры key – Параметр.

classmethod setValue(key, value)
Устанавливает значение параметра.

Параметры

- **key** – Параметр.
- **value** – Значение.

Исключение TypeError – Если значение неправильного типа.

Например:

```
Settings.setValue(Settings.LastOpenPath, 'C:/mydir')
```

classmethod value(key)
Читает значение параметра.

Параметры key – Параметр.

Тип результата Any

Результат Значение.

Например:

```
val = Settings.value(Settings.LastOpenPath)
```

Функции

axipy.**init_axioma**()
Инициализирует ядро ГИС Аксиома.

Тип результата QApplication

Результат Приложение Qt5 с очередью событий (event-loop).

Пример:

```
app = init_axioma()
app.exec_()  # запускает обработку очереди событий
```

axipy.**tr**(text)
Ищет перевод строки строки.

Производит поиск строки в загруженных файлах перевода.

Параметры **text** (str) – Строка для перевода.

Результат Перевод стоки, если строка найдена. Иначе - сама переданная строка.

Пример:

```
button_name = tr('My button')
```

16.1.2 axipy.app

Модуль приложения.

Данный модуль является основным модулем приложения.

axipy.app.**mainwindow**
Готовый экземпляр главного окна ГИС Аксиома.

Type MainWindow

16.1.2.1 MainWindow - Главное окно

class axipy.app.**MainWindow**
Главное окно ГИС Аксиома.

Примечание: Используйте готовый объект axipy.app.mainwindow.

Methods:

add(view)	Добавляет окно просмотра данных.
add_dock_widget(dock_widget, area[, icon])	Добавляет панель в главное окно приложения.
add_layer_current_map(layer)	Добавляет слой в текущей карте.

continues on next page

Таблица 7 – продолжение с предыдущей страницы

add_layer_interactive(layer)	Добавляет слой с запросом на помещение на текущую карту или в новую.
add_layer_new_map(layer)	Открывает слой в новой карте.
load_workspace(fileName)	Читает рабочее пространство из файла.
qt_object()	Возвращает Qt5 объект окна.
remove_dock_widget(dock)	Удаляет существующую панель у главного окна приложения.
save_workspace(fileName)	Сохраняет рабочее пространство в файл.
show()	Создает и показывает главное окно программы.
show_html_url(url, caption)	Показывает окно для локального файла html или если это web страница, запускает браузер по ассоциации

Attributes:

catalog	Хранилище объектов приложения.
is_valid	Корректность состояния главного окна.

add(view)
Добавляет окно просмотра данных.

Параметры **view** (View) – окно просмотра данных.

Примечание: При создании окон просмотра данных axipy.gui.ViewManager.create_mapview() или axipy.gui.ViewManager.create_tableview() они автоматически добавляются в главное окно программы.

Тип результата QMdiSubWindow

add_dock_widget(dock_widget, area, icon=None)
Добавляет панель в главное окно приложения. При успешном добавлении возвращает True. Если же данная панель уже присутствует, то команда игнорируется и возвращается False. Элементы управления, которые требуется разместить на панели, создаются в дополнительном окне, а уже это окно, в свою очередь, устанавливается для панели (см. пример ниже).

Параметры

- **dock_widget** (QDockWidget) – Пользовательская созданная панель.
- **area** (DockWidgetArea) – Расположение.
- **icon** (Optional[QIcon]) – Иконка для отображения в списке всех доступных панелей.

Пример:

```
from PySide2.QtWidgets import QDockWidget, QWidget, QPushButton
from PySide2.QtCore import Qt
```

(continues on next page)

(продолжение с предыдущей страницы)

```
dock = QDockWidget('Заголовок')
widget = QWidget()
layout = QVBoxLayout()
button = QPushButton("Кнопка")
button.clicked.connect(lambda: print('Реакция на кнопку'))
layout.addWidget(button)
layout.addStretch()
widget.setLayout(layout)
dock.setWidget(widget)
app.mainwindow.add_dock_widget(dock, Qt.RightDockWidgetArea, QIcon('filename.
↪png'))
```

Тип результата bool

add_layer_current_map(layer)
Добавляет слой в текущей карте.

Тип результата MapView

add_layer_interactive(layer)
Добавляет слой с запросом на помещение на текущую карту или в новую.

Тип результата MapView

add_layer_new_map(layer)
Открывает слой в новой карте.

Тип результата MapView

property catalog
Хранилище объектов приложения.

Это то же хранилище, которое отображается в панели «Открытые данные».

Примечание: При открытии объектов данных axipy.da.ProviderManager.openfile() они автоматически попадают в каталог.

Тип результата DataManager

property is_valid
Корректность состояния главного окна.

Тип результата bool

load_workspace(fileName)
Читает рабочее пространство из файла.

Параметры **fileName** (str) – Наименование входного файла.

qt_object()
Возвращает Qt5 объект окна.

Тип результата QMainWindow

remove_dock_widget(dock)
Удаляет существующую панель у главного окна приложения.

save_workspace(fileName)
Сохраняет рабочее пространство в файл.

Параметры **fileName** (str) – Наименование выходного файла.

static show()
Создает и показывает главное окно программы.

Тип результата MainWindow

show_html_url(url, caption)
Показывает окно для локального файла html или если это web страница, запускает браузер по ассоциации

Параметры

- **url** (QUrl) – Ссылка на файл html или адрес страницы.
- **caption** (Optional[str]) – Заголовок окна

16.1.2.2 Notifications - Отправление уведомлений

class axipy.app.**Notifications**
Отправление уведомлений в виде всплывающего окна с его последующей регистрацией в окне уведомлений.

Methods:

push(title, text[, type_message])	Отправляет уведомление.

static push(title, text, type_message=0)
Отправляет уведомление.

Параметры

- **title** (str) – Заголовок
- **text** (str) – Текст сообщения.
- **type_message** (int) – Тип сообщения. В зависимости от типа сообщения в окне уведомлений оно помечается соответствующим цветом.

Таблица 10: Допустимые значения для типа сообщения:

Атрибут	Наименование
Information	Информационное сообщение. Устанавливается по умолчанию
Warning	Предупреждение
Critical	Критическая ошибка
Success	Успешное выполнение процесса

Пример:

```
from axipy.app import Notifications
Notifications.push('Предупреждение', 'Сообщение', Notifications.Warning)
```

16.1.2.3 Version - Информация о версии

class axipy.app.**Version**

Информация о версии ГИС Аксиома.

Пример использования:

```
from axipy.app import Version
print('Версия:', Version.string(), Version.compare(4,3))
```

Methods:

compare(major, minor, patch)	Сравнивает переданное значение с текущей версией Аксиомы.
number()	Возвращает версию в виде одного числа, в котором каждый сегмент располагается в отдельном байте.
qtFormat()	Возвращает версию в Qt формате.
segments()	Возвращает кортеж чисел - сегменты версии: (major, minor, patch).
string()	Возвращает версию в виде строки.

static compare(major, minor, patch)

Сравнивает переданное значение с текущей версией Аксиомы.

Параметры

- **major** (int) – Основная версия
- **minor** (Optional[int]) – Минорная версия
- **patch** (Optional[int]) – Исправления

Возвращает -1, если переданная меньше, 1 если больше и 0 если равны

Тип результата int

static number()

Возвращает версию в виде одного числа, в котором каждый сегмент располагается в отдельном байте.

Тип результата int

static qtFormat()

Возвращает версию в Qt формате.

Тип результата QVersionNumber

static segments()

Возвращает кортеж чисел - сегменты версии: (major, minor, patch).

Тип результата tuple

static string()

Возвращает версию в виде строки.

Тип результата str

16.1.3 axipy.cs

Модуль систем координат.

В данном модуле содержатся классы и методы, предназначенные для удобной работы с координатными системами.

16.1.3.1 CoordSystem - Система Координат (СК)

class axipy.cs.**CoordSystem**

Система координат (СК). СК описывает каким образом реальные объекты на земной поверхности могут быть представлены в виде двумерной проекции. Выбор СК для представления данных зависит от конкретных исходных условий по представлению исходных данных.

Примечание: Проверка на идентичность параметров двух СК производится простым сравнением.

Примечание: Для получения текстового представления можно воспользоваться функцией `str`.

Поддерживается создание СК посредством следующих вариантов:

- Из строки MapInfo PRJ `from_prj()`
- Из строки PROJ `from_proj()`
- Из строки WKT `from_wkt()`
- Из значения EPSG `from_epsg()`
- План/Схему с указанием единиц измерения и охвата `from_units()`

Список 2: Пример создания СК разного типа.

```
cs_epsg = CoordSystem.from_epsg(4326)
cs_prj = CoordSystem.from_prj('1, 104')
cs_proj = CoordSystem.from_proj('+proj=longlat +ellps=WGS84 +no_defs')
cs_wkt = CoordSystem.from_wkt('GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SPHEROID[
↪"WGS_1984",6378137,298.257223563]],PRIMEM["Greenwich",0],UNIT["Degree",0.
↪0174532925199433295]]')
# Создание из строки с указанием вида формата
crs1 = CoordSystem.from_string('epsg:4326')
crs2 = CoordSystem.from_string('prj:1,104')
```

Список 3: Проверка на идентичность координатных систем производится простым сравнением.

```
cs1 = CoordSystem.from_prj("1, 104")
cs2 = CoordSystem.from_prj("1, 104")
if cs1 == cs2:
    print("Координатные системы эквивалентны.")
'''
```

(continues on next page)

(продолжение с предыдущей страницы)

```
>>> Координатные системы эквивалентны.
'''
```

convert_from_degree(value)
Переводит из градусов в единицы измерения системы координат.

Тип результата Union[Pnt, List[Pnt], Rect]

convert_to_degree(value)
Переводит из единиц измерения системы координат в градусы.

Список 4: Пример.

```
csMercator = CoordSystem.from_prj("10, 104, 7, 0")
p_out = csMercator.convert_to_degree((1000000, 1000000))
print(p_out)
'''
>>> (8.983152841195214 9.005882635078796)
'''
```

Тип результата Union[Pnt, List[Pnt], Rect]

classmethod current()
Текущая установленная система координат (СК). Данная СК используется как значение по умолчанию, когда она не определена. Например, в диалоге создания новой таблицы.

Тип результата CoordSystem

property description
Краткое текстовое описание.

Тип результата str

property epsg
Значение EPSG если существует для данной системы координат, иначе None.

Тип результата Optional[int]

classmethod from_epsg(code)
Создает координатную систему по коду EPSG.

См.также:

Подробнее см. EPSG

Параметры code (int) – Стандартное значение EPSG.

Тип результата CoordSystem

classmethod from_prj(prj)
Создает координатную систему из строки MapBasic.

См.также:

Подробнее см. PRJ

Параметры prj (str) – Строка MapBasic. Допустима короткая нотация.

Список 5: Пример.

```
csMercator = CoordSystem.from_prj('10, 104, 7, 0')
csLatLon = CoordSystem.from_prj('Earth Projection 1, 104')
csMercator = CoordSystem.from_prj('NonEarth 0, \'m\'')
```

Тип результата CoordSystem

classmethod from_proj(proj)
Создает координатную систему из строки proj.

См.также:

Подробнее см. PROJ

Параметры **proj** (str) – Строка proj.

Тип результата CoordSystem

classmethod from_string(string)
Создает систему координат из строки.

Строка состоит из двух частей: префикса и строки представления СК. Возможные значения префиксов: «proj», «wkt», «epsg», «prj».

Параметры **string** (str) – Строка.

Тип результата CoordSystem

classmethod from_units(unit, rect=<axipy.utl.Rect object>)
Создает декартову систему координат.

Параметры

- **unit** (LinearUnit) – Единицы измерения системы координат.
- **rect** (Union[Rect, QRectF, None]) – Охват системы координат.

Список 6: Пример.

```
ne = CoordSystem.from_units(Unit.km, Rect(-100, -100, 100, 100))
```

Тип результата CoordSystem

classmethod from_wkt(wkt)
Создает координатную систему из строки WKT.

См.также:

Подробнее см. WKT

Параметры **wkt** (str) – Строка WKT.

Тип результата CoordSystem

property inv_flattening
Полярное сжатие.

Тип результата float

property lat_lon
Является ли данная СК широтой/долготой.

Тип результата bool

property name
Наименование системы координат.

Тип результата str

property non_earth
Является ли данная СК декартовой.

Тип результата bool

property prj
Строка prj формата MapBasic или пустая строка, если аналога не найдено.

Тип результата str

property proj
Строка PROJ или пустая строка, если аналога не найдено.

Тип результата str

property rect
Максимально допустимый охват.

Тип результата Rect

property semi_major
Большая полуось.

Тип результата float

property semi_minor
Малая полуось.

Тип результата float

classmethod set_current(coordsystem)
Устанавливает новую текущую систему координат

Параметры **cs** – Новое значение системы координат.

Пример установки нового значения:

```
CoordSystem.set_current(CoordSystem.from_prj("10, 104, 7"))
```

to_string()
Текстовое представление в виде <тип>:<строка>

property unit
Единицы измерения.

Тип результата LinearUnit

property wkt
Строка WKT или пустая строка, если аналога не найдено.

Тип результата str

16.1.3.2 CoordTransformer - Трансформация координат

class axipy.cs.**CoordTransformer**(cs_from, cs_to)

Класс для преобразования координат из одной СК в другую. При создании объекта трансформации в него передается исходная и целевая СК. После этого данный объект может использоваться для преобразования данных между этими СК.

Параметры

- **cs_from** (Union[CoordSystem, str]) – Исходная СК.
- **cs_to** (Union[CoordSystem, str]) – Целевая СК.

Список 7: Пример преобразования точки

```
from axipy import CoordSystem, CoordTransformer

csLL = CoordSystem.from_prj("1, 104")
csMercator = CoordSystem.from_prj("10, 104, 7, 0")
inPoint = Pnt(10, 10)
transformer = CoordTransformer(csLL, csMercator)
outPoint = transformer.transform(inPoint)
print('Result point:', outPoint)
'''
>>> Result point: (1113194.9079327357 1111475.1028522244)
'''
outRect = transformer.transform(Rect(0,0,10,10))
print('Result rect:', outRect)
'''
>>> Result rect: (0.0 0.0) (1113194.9079327357 1111475.1028522244)
'''
```

classmethod proj_transform_definition(cs_from, cs_to)

Возвращает строку трансформации (pipeline) для преобразования между двумя СК, заданными в формате proj.

Параметры

- **cs_from** (str) – Строка с определением исходной СК в формате proj
- **cs_to** (str) – Строка с определением исходной СК в формате proj

Тип результата str

Результат Строка с определением трансформации между двумя этими СК в формате proj

Список 8: Пример получения строки трансформации

```
str_from = '+proj=longlat +ellps=WGS84 +no_defs'
str_to = '+proj=merc +ellps=GRS80 +no_defs'
print(CoordTransformer.proj_transform_definition(str_from, str_to))
'''
>>> proj=pipeline step proj=unitconvert xy_in=deg xy_out=rad step proj=merc␣
↪lon_0=0 k=1 x_0=0 y_0=0 ellps=GRS80
'''
```

transform(value)

Преобразовывает точки из исходной СК в целевую СК.

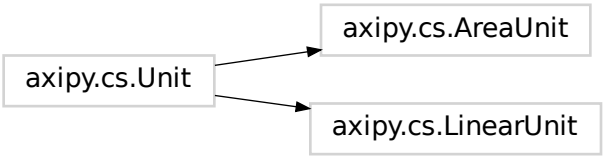
Параметры value (Union[Pnt, List[Pnt], QPointF, QRectF, Rect, List[QPointF]]) – Входное значение. Может быть точкой, массивом точек axipy.utl.Pnt или axipy.utl.Rect.

Тип результата Union[Pnt, Rect, List[Pnt]]

Результат Выходное значение. Тип зависит от входного и аналогичен ему.

Исключение RuntimeError – Ошибка выполнения преобразования.

16.1.3.3 Unit - Единицы измерения



class axipy.cs.**Unit**

Класс единиц измерения.

Получение экземпляра единиц измерения осуществляется по атрибуту.

Список 9: Пример создания

```
meters = Unit.m  # LinearUnit
kilometers = Unit.sq_km  # AreaUnit
```

Таблица 12: Доступные единицы расстояний

Атрибут	Тип	Наименование
km	LinearUnit	Километры
m	LinearUnit	Метры
mm	LinearUnit	Миллиметры
cm	LinearUnit	Сантиметры
mi	LinearUnit	Мили
nmi	LinearUnit	Морские мили.
inch	LinearUnit	Дюймы
ft	LinearUnit	Футы
yd	LinearUnit	Ярды
survey_ft	LinearUnit	Топографические футы.
li	LinearUnit	Линки
ch	LinearUnit	Чейны
rd	LinearUnit	Роды

Таблица 13: Доступные единицы площадей

Атрибут	Тип	Наименование
sq_km	AreaUnit	Квадратные километры
sq_m	AreaUnit	Квадратные метры
sq_mm	AreaUnit	Квадратные миллиметры
sq_cm	AreaUnit	Квадратные сантиметры
sq_mi	AreaUnit	Квадратные мили
sq_nmi	AreaUnit	Квадратные морские мили
sq_inch	AreaUnit	Квадратные дюймы
sq_ft	AreaUnit	Квадратные футы
sq_yd	AreaUnit	Квадратные ярды
sq_survey_ft	AreaUnit	Квадратные топографические футы
acre	AreaUnit	Акры
hectare	AreaUnit	Гектары
sq_li	AreaUnit	Квадратные линки
sq_ch	AreaUnit	Квадратные чейны
sq_rd	AreaUnit	Квадратные роды
perch	AreaUnit	Перчи
rood	AreaUnit	Руды

Так же доступно задание единиц в градусах через свойство `Unit.degree`

Attributes:

conversion	Коэффициент преобразования в метры.
description	Краткое описание.
localized_name	Локализованное краткое наименование единиц измерения.
name	Краткое наименование единиц измерения.

Methods:

to_unit(unit[, value])	Перевод значения в другие единицы измерения.

property conversion
Коэффициент преобразования в метры.

Тип результата float

property description
Краткое описание.

Тип результата str

property localized_name
Локализованное краткое наименование единиц измерения.

Тип результата str

property name
Краткое наименование единиц измерения.

Тип результата str

to_unit(unit, value=1)
Перевод значения в другие единицы измерения.

Параметры

- **unit** (Union[LinearUnit, AreaUnit]) – Единицы измерения, в которые необходимо перевести значение.
- **value** (float) – Значение для перевода.

Пример:

```
from axipy import *

print("Linear:", unit.km.to_unit(unit.m, 2))
print("Area:", unit.sq_km.to_unit(unit.sq_m, 2))

>>> Linear: 2000.0
>>> Area: 2000000.0
```

Тип результата float

16.1.3.4 LinearUnit - Единицы измерения расстояний

class axipy.cs.**LinearUnit**

Базовые классы: axipy.cs.Unit

Линейные единицы измерения.

Используются для работы с координатами объектов или расстояний.

Примечание: Получить экземпляр можно через базовый класс axipy.cs.Unit по соответствующему атрибуту.

Список 10: Пример создания

```
meters = Unit.m  # LinearUnit
kilometers = Unit.sq_km  # AreaUnit
```

Methods:

by_name(name)	Возвращает единицу измерения по ее наименованию.
list_all()	Возвращает перечень всех линейных единиц измерения.
to_unit(unit[, value])	Перевод значения в другие единицы измерения.

Attributes:

conversion	Коэффициент преобразования в метры.
description	Краткое описание.

continues on next page

Таблица 17 – продолжение с предыдущей страницы

localized_name	Локализованное краткое наименование единиц измерения.
name	Краткое наименование единиц измерения.

classmethod by_name(name)
Возвращает единицу измерения по ее наименованию.

Attrs: name: Наименование name, localized_name или description

Тип результата Optional[LinearUnit]

property conversion
Коэффициент преобразования в метры.

Тип результата float

property description
Краткое описание.

Тип результата str

classmethod list_all()
Возвращает перечень всех линейных единиц измерения.

Тип результата List[LinearUnit]

property localized_name
Локализованное краткое наименование единиц измерения.

Тип результата str

property name
Краткое наименование единиц измерения.

Тип результата str

to_unit(unit, value=1)
Перевод значения в другие единицы измерения.

Параметры

- **unit** (Union[LinearUnit, AreaUnit]) – Единицы измерения, в которые необходимо перевести значение.
- **value** (float) – Значение для перевода.

Пример:

```
from axipy import *

print("Linear:", unit.km.to_unit(unit.m, 2))
print("Area:", unit.sq_km.to_unit(unit.sq_m, 2))

>>> Linear: 2000.0
>>> Area: 2000000.0
```

Тип результата float

16.1.3.5 AreaUnit - Единицы измерения площадей

class axipy.cs.**AreaUnit**

Базовые классы: axipy.cs.Unit

Единицы измерения площадей.

Примечание: Получить экземпляр можно через базовый класс axipy.cs.Unit по соответствующему атрибуту.

Список 11: Пример создания

```
meters = Unit.m  # LinearUnit
kilometers = Unit.sq_km  # AreaUnit
```

Methods:

by_name(name)	Возвращает единицу измерения по ее наименованию.
list_all()	Возвращает перечень всех площадных единиц измерения.
to_unit(unit[, value])	Перевод значения в другие единицы измерения.

Attributes:

conversion	Коэффициент преобразования в метры.
description	Краткое описание.
localized_name	Локализованное краткое наименование единиц измерения.
name	Краткое наименование единиц измерения.

classmethod by_name(name)
Возвращает единицу измерения по ее наименованию.

Attrs: name: Наименование name, localized_name или description

Тип результата Optional[AreaUnit]

property conversion
Коэффициент преобразования в метры.

Тип результата float

property description
Краткое описание.

Тип результата str

classmethod list_all()
Возвращает перечень всех площадных единиц измерения.

Тип результата List[AreaUnit]

property localized_name
Локализованное краткое наименование единиц измерения.

Тип результата str

property name
Краткое наименование единиц измерения.

Тип результата str

to_unit(unit, value=1)
Перевод значения в другие единицы измерения.

Параметры

- **unit** (Union[LinearUnit, AreaUnit]) – Единицы измерения, в которые необходимо перевести значение.
- **value** (float) – Значение для перевода.

Пример:

```
from axipy import *

print("Linear:", unit.km.to_unit(unit.m, 2))
print("Area:", unit.sq_km.to_unit(unit.sq_m, 2))

>>> Linear: 2000.0
>>> Area: 2000000.0
```

Тип результата float

16.1.3.6 UnitValue - Значение вместе с единицей измерения

class axipy.cs.**UnitValue**(value=1, unit=None)

Базовые классы: object

Контейнер, который хранит значение вместе с его единицей измерения.

Параметры

- **value** (float) – Значение.
- **unit** (Optional[Unit]) – Единица измерения, в которой содержится значение. Если значение не указано, принимаются метры LinearUnit.m.

Пример:

```
unit =  UnitValue(2, LinearUnit.km)
print(unit)
>>> 2 km
```

Attributes:

unit	Единица измерения.
value	Значение.

property unit

Единица измерения.

Тип результата Unit

property value
Значение.

Тип результата float

16.1.4 axipy.concurrent

Модуль для работы с длительными пользовательскими задачами, выполняемыми в фоновом потоке.

axipy.concurrent.**task_manager**
Экземпляр менеджера запускающего пользовательские задачи.

Type TaskManager

16.1.4.1 AxipyTask - Пользовательская задача

class axipy.concurrent.**AxipyTask**
Базовый класс пользовательской задачи. От него можно наследоваться, создавая собственный задачи. Для этого нужно переопределить метод run().

Methods:

on_finished(result)	Функция вызывается при завершении пользовательской задачи в потоке интерфейса.
progress_handler()	Возвращает обработчик для текущей задачи.
run()	Функция, выполняющая код пользовательской задачи.
set_progress_handler(ph)	Устанавливает обработчик для текущей задачи.

on_finished(result)
Функция вызывается при завершении пользовательской задачи в потоке интерфейса.

progress_handler()
Возвращает обработчик для текущей задачи.

Тип результата AxipyProgressHandler

abstract run()
Функция, выполняющая код пользовательской задачи. Переопределяется в дочерних классах.

set_progress_handler(ph)
Устанавливает обработчик для текущей задачи. Поддерживаются любые наследники AxipyProgressHandler.

16.1.4.2 AxipyAnyCallableTask - Обертка над пользовательской функцией для создания задачи

class axipy.concurrent.**AxipyAnyCallableTask**(fn, *args, **kwargs)

Объекты этого класса оборачивают пользовательские функции, превращая их в задачу, которая будет выполнена в фоновом потоке.

Параметры

- **fn** – Пользовательская функция, которая будет выполнятся. В нее будут переданы сохраненные параметры: список args и словарь kwargs.
- **args** – Список аргументов, передаваемый в функцию при запуске.
- **kwargs** – Словарь, передаваемый в функцию при запуске.

Список 12: Пример использования.

```
def user_heavy_function(arg1: int, arg2: str):
    print(f"Переданные аргументы: {arg1}, {arg2} \n")

task = AxipyAnyCallableTask(user_heavy_function, arg1=1, arg2="Тест")
task.with_handler(False)
task_manager.start_task(task)
```

Methods:

run()	Метод запускает выполнение задачи.
with_handler(value)	По умолчанию в пользовательскую функцию первым аргументом передаётся обработчик для управления задачей, установки прогресса и обработки отмены.

run()

Метод запускает выполнение задачи.

> **Предупреждение:** Вызывается автоматически при выполнении задачи. Вручную вызывать не следует.

with_handler(value)

По умолчанию в пользовательскую функцию первым аргументом передаётся обработчик для управления задачей, установки прогресса и обработки отмены. Однако он не будет передаваться если вызвать эту функцию с False.

16.1.4.3 AxipyProgressHandler - Объект для связи с задачей и её управлением

class axipy.concurrent.**AxipyProgressHandler**

Класс, объекты которого выполняют функцию канала передачи данных между выполняемой задачей и элементом отображающим прогресс.

error

Сигнал об ошибке, содержащий информацию о исключении.

Type Singal[tuple]

add_progress(value)

Добавляет к текущему прогрессу переданное значение.

Параметры **value** (float) – Значение, которое будет добавлено к прогрессу.

cancel()

Отменяет задачу, связанную с обработчиком.

Примечание: Эта функция посылает только запрос на отмену операции. Реальное прерывание операции возможно только если есть поддержка в пользовательском коде. Например, с помощью функций is_canceled или raise_if_canceled

---

property canceled

Signal[] Уведомляет, что задача была отменена. Сигнал испускается когда была вызвана функция cancel.

Предупреждение: Получение этого сигнала не означает, что задача была завершена. Если в пользовательском коде не обрабатывается отмена, то задача будет продолжать выполняться до логического завершения.

Тип результата Signal

property description_changed

Signal[str] Уведомляет о изменении описания задачи.

Тип результата Signal

property finished

Signal[] Уведомляет о завершении выполняемой задачи.

Тип результата Signal

is_canceled()

Проверяем не была ли задача отменена.

Тип результата bool

is_finished()

Проверяем не завершилась ли задача.

Тип результата bool

is_running()

Возвращает True если задача сейчас выполняется.

Тип результата bool

prepare_to_write_changes(description='')
Делает индикатор выполнения бесконечным, убирает кнопку отмены и добавляет переданное описание. По умолчанию описание содержит запись о том, что производится запись изменений.

Параметры description (str) – Сообщение которое будет отображаться.

progress()
Возвращает текущий прогресс выполнения.

Тип результата float

property progress_changed
Signal[float] Уведомляет о изменении значения прогресса.

Тип результата Signal

raise_if_canceled()
Если задача была отменена выбрасывает исключение. Удобно при работе с большим количеством вложенных циклов или вызовов функции.

property result
Содержит результат выполнения задачи, связанной с обработчиком. Возвращается None если произошла ошибка, либо задача не предполагает возвращение результата.

Результат Результат выполнения задачи или None.

set_description(description)
Устанавливаем описание для задачи. Эта информация может быть использована элементами отображающими прогресс выполнения.

Параметры description (str) – Новое описание задачи.

set_max_progress(value)
Устанавливает максимальное значение прогресса. Минимальное значение берется за ноль.

Параметры value (float) – Верхний порог для прогресса операции.

set_progress(value)
Устанавливает текущий прогресс задачи.

Параметры value (float) – Новое значение прогресса.

set_window_title(title)
Устанавливает заголовок диалога с прогрессом.

Параметры title (str) – Новый заголовок.

property started
Signal[] Уведомляет о старте выполнения задачи.

Тип результата Signal

property window_title_changed
Signal[str] Уведомляет о изменении заголовка диалога отображающего прогресс.

Тип результата Signal

16.1.4.4 TaskManager - Сервис для запуска и конфигурирования пользовательских задач

class axipy.concurrent.**TaskManager**(progress_element_factory=<axipy.concurrent.TaskManager.Prog object>)

Сервис, содержащий вспомогательные функции для запуска и конфигурирования пользовательских задач.

generate_dialog_for_task(task, spec)

Возвращает диалог, следящий за выполнением задачи.

Параметры

- **task** (AxipyTask) – Пользовательская задача.
- **spec** (ProgressSpecification) – Описание задачи.

Тип результата QDialog

Результат Диалог, который будет отображать прогресс выполнения задачи.

Список 13: Пример использования.

```
def user_heavy_func(ph: AxipyProgressHandler, arg1: str, arg2: str):
    print(arg1 + arg2)

# Создаём задачу из пользовательской функции и передаём необходимые
# аргументы
task = AxipyAnyCallableTask(user_heavy_func, "Длительная ", "задача")
spec = ProgressSpecification(description="Длительная задача")
dialog = task_manager.generate_dialog_for_task(task, spec)
# Ставим задачу в очередь на выполнение
task_manager.start_task(task)
# Показываем диалог с прогрессом пока не завершилась задача
dialog.exec_()
```

run_and_get(spec, func, *args, **kwargs)

Превращает пользовательскую функцию в задачу и запускает её с отображением прогресса выполнения.

Параметры

- **spec** (ProgressSpecification) – Описание задачи.
- **func** (Callable) – Пользовательская функция, которая будет выполняться. В нее передается список args и словарь kwargs.
- **args** – Список аргументов, передаваемый в функцию при запуске.
- **kwargs** – Словарь, передаваемый в функцию при запуске.

Тип результата Any

Результат Результат выполнения пользовательской функции или None при ошибке.

Список 14: Пример использования.

```
def user_heavy_function(ph: AxipyProgressHandler, count: int):
    # Вначале задаём верхнюю планку изменения прогресса
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    ph.set_max_progress(count)
    for i in range(0, count):
        if ph.is_canceled():
            break
        # Тут делаем длительные вычисления
        ph.add_progress(1)
    return ph.progress()

spec = ProgressSpecification(
    description="Длительная операция",
    flags=ProgressGuiFlags.CANCELABLE)
times = 6
real_times = task_manager.run_and_get(spec, user_heavy_function, times)
# выводим колличество раз которое отработал цикл внутри
print(real_times)
```

run_in_gui(func, *args, **kwargs)
Выполняет переданную задачу в потоке интерфейса.

Это может быть удобно когда в процессе выполнения длительной фоновой задачи нужно спросить о чем нибудь пользователя отобразив диалог. Так же создавать/взаимодействовать с некоторыми объектами можно только из потока интерфейса.

Тип результата Any

start_task(task)
Добавляет переданную задачу в очередь на выполнение.

Параметры **task** – Пользовательская задача.

Список 15: Пример использования.

```
def user_function(message: str):
    print(message)
task = AxipyAnyCallableTask(user_function, "Hi, world!")
# Т.к. мы не управляем прогрессом, то можно отключить передачу
# обработчика в функцию
task.with_handler(False)
task_manager.start_task(task)
```

16.1.4.5 ProgressGuiFlags - Флаги для настройки диалога, отображающего прогресс

class axipy.concurrent.**ProgressGuiFlags**(value)
Флаги для настройки диалога, отображающего прогресс

Наименование	Значение	Описание
IDLE	1	Просто диалог с прогрессом.
CANCELABLE	2	У диалога с прогрессом будет кнопка отмены.
NO_DELAY	4	Диалог с прогрессом появляется сразу без задержки. По умолчанию это 2 - 3 секунды.

16.1.4.6 ProgressSpecification - Параметры для настройки диалога, отображающего прогресс

class axipy.concurrent.**ProgressSpecification**(description='', window_title='', flags=<ProgressGuiFlags.IDLE: 0>, with_handler=True)

Содержит параметры для настройки отображения диалога с прогрессом.

Список 16: Пример использования.

```
flags = ProgressGuiFlags.CANCELABLE | ProgressGuiFlags.NO_DELAY
spec = ProgressSpecification(
    description="Тестовое описание",
    window_title="Заголовок окна",
    flags=flags)
```

window_title
Задаёт заголовок окна для диалога, отображающего прогресс.

Type str

flags
С помощью флагов можно выбрать тип диалога который будет отображаться пользователю. Флаги можно комбинировать с помощью побитовых операций.

Type ProgressGuiFlags

description
Описание выполняемой задачи.

Type str

with_handler
По умолчанию в пользовательскую функцию будет передаваться обработчик прогресса выполнения задачи AxipyProgressHandler. Но иногда это не нужно, тогда можно этот параметр установить в False.

Type bool

16.1.5 axipy.utl

Сервисные классы.

16.1.5.1 Pnt - Точка

class axipy.utl.**Pnt**(x, y)

Точка без геопривязки. Может быть использована в качестве параметра геометрии (точки полигона) или при получении параметров, где результат представлен в виде точки (центр карты или элемента отчета).

Параметры

- **x** (float) – X координата.
- **y** (float) – Y координата.

Methods:

from_qt(p)	Преобразует из формата Qt.
to_qt()	Преобразование в формат Qt.

Attributes:

x	Координата X.
y	Координата Y.

classmethod from_qt(p)
Преобразует из формата Qt. Если класс не соответствует, возвращает None

Параметры p (Union[QPointF, QPoint]) – Преобразуемая точка.

Тип результата Optional[Pnt]

to_qt()
Преобразование в формат Qt.

Тип результата QPointF

property x
Координата X.

Тип результата float

property y
Координата Y.

Тип результата float

16.1.5.2 Rect - Прямоугольник

class axipy.utl.**Rect**(xmin, ymin, xmax, ymax)
Прямоугольник, который не обладает геопривязкой. Используется для различного вида запросов.

property center
Центр прямоугольника.

Тип результата Pnt

contains(other)
Содержит ли полностью в своих границах переданный объект.

Параметры other (Union[Pnt, QPointF, Rect, QRectF]) – Переданный объект - точка или прямоугольник.

Пример:

```
r = Rect(2,2,5,5)
print(r.contains((3,3)))
print(r.contains((3,10)))
print(r.contains(Rect(3,3,7,7)))
print(r.contains(Rect(3,3,4,4)))

>>> True
>>> False
```

(continues on next page)

(продолжение с предыдущей страницы)

```
>>> False
>>> True
```

Тип результата bool

expanded(dx, dy)

Возвращает прямоугольник, увеличенный на заданные величины. Увеличение размеров производится по отношению к центру, который не меняется в результате операции.

Параметры

- **dx** (float) – расширение по X
- **dy** (float) – расширение по Y

Пример:

```
r = Rect(2,2,5,5)
print(r.expanded(2, 4))
>>> (1.0 0.0) (6.0 7.0)
```

Тип результата Rect

classmethod from_qt(r)

Преобразует из формата Qt. Если класс не соответствует, возвращает None

Параметры **r** (Union[QRectF, QRect]) – Преобразуемый прямоугольник.

Тип результата Optional[Rect]

property height

Высота прямоугольника.

Тип результата float

intersected(other)

Возвращает общий для обоих прямоугольник.

Параметры **other** (Rect) – Прямоугольник, с которым производится операция.

Пример:

```
r1 = Rect(2,2,5,5)
r2 = Rect(3,3,7,7)
print(r1.intersected(r2))
>>> (3.0 3.0) (5.0 5.0)
```

Тип результата Rect

property is_empty

Если один или оба размера равны нулю.

Тип результата bool

property is_valid

Является ли прямоугольник правильным.

Тип результата bool

merge(other)
Возвращает прямоугольник, занимаемый обоими прямоугольниками.

Параметры **other** (Rect) – Прямоугольник, с которым производится операция.

Пример:

```
r1 = Rect(2,2,5,5)
r2 = Rect(3,3,7,7)
print(r1.merge(r2))
>>> (2.0 2.0) (7.0 7.0)
```

Тип результата Rect

normalize()
Исправляет прямоугольник, если его ширина или высота отрицательны.

Тип результата Rect

to_qt()
Преобразование в формат Qt.

Тип результата QRectF

translated(dx, dy)
Возвращает прямоугольник, смещенный на заданную величину.

Параметры

- **dx** (float) – смещение по X
- **dy** (float) – смещение по Y

Пример:

```
r = Rect(2,2,5,5)
print(r.translated(10, -10))
>>> (12.0 -8.0) (15.0 -5.0)
```

Тип результата Rect

property width
Ширина прямоугольника.

Тип результата float

property xmax
Максимальное значение X.

Тип результата float

property xmin
Минимальное значение X.

Тип результата float

property ymax
Максимальное значение Y.

Тип результата float

property ymin
Минимальное значение Y.

Тип результата float

16.1.5.3 CoordFormatter - Преобразование значений координат

class CoordFormatter
Класс производит преобразование значений из строки в тип float и обратно. Под значениями в первую очередь следует понимать координаты объектов.

Пример использования:

```
from axipy.utl import CoordFormatter

reader = CoordFormatter()
# Строковое представление в число
res, succ = reader.as_float('55-55-4.78')
print('>>>>', res, succ)
# Число в строку
print(reader.as_string(33.3744777))
print( reader.as_string_with_delimeter(33.3744777, '-'))
print( reader.as_string_with_delimeter(33.3744777, '/'))
# румбы
res, succ = reader.as_float('ЮВ 46.6')
print('>>>', res, succ)
print( reader.double_to_rumb(133.3744777))
```

```
# Вывод
# >>>  55.91799444444444, True
# >>> 33°22'28,11972
# >>> 33-22-28,11972
# >>> 33/22/28,11972
# >>> 133.4 True
# >>> ЮВ 46.6255223
```

as_float(in)
Преобразует строку в числовое значение.

Допустимый формат входной строки: <dd°mm'ss,zz»>, <dd mm ss,zz>, <dd/mm/ss,zz>, <dd-mm-ss,zz>, <dd,mm,ss.zz>, <dd.zz>, <dd,zz> или в румбах ЮВ dd.zz

Параметры **in** (str) – Значение в виде строки

Результат Пара значение/успешность операции

Тип результата tuple(float, bool)

as_string(in)
Преобразует число в строку в формате <dd°mm'ss,zz»>.

Параметры **in** (float) – Входное значение

as_string_with_delimeter(in, delimeter)
Преобразует число в строку по формату с заданным разделителем.

Параметры

- **in** (float) – Входное значение

- **delimeter** (str) – Разделитель. Например, „-„ или „/“

double_to_rumb(in)
Преобразует число в строку в формате румбов.

Параметры **in** (float) – Значение в градусах

Результат Строка в формате румбов.

Тип результата str

16.1.5.4 FloatFormatter - Преобразование float в str

class axipy.utl.**FloatFormatter**
Класс для преобразования чисел с плавающей точкой в текст

Methods:

float_round(value, precision)	Округляет число до заданной точности
float_round_signific(value[, digits])	Округляет число с указанием количества значащих цифр.
float_to_str(value[, use_delimeter, precision])	Представляет значение типа float в виде строки с заданными параметрами.
to_localized_string(value[, locale])	Возвращает число в виде строки форматированный с учетом переданной локали.
to_localized_string_round(value, precision)	Возвращает число в виде строки.

static float_round(value, precision)
Округляет число до заданной точности

Параметры

- **value** (float) – Входное значение
- **precision** (int) – Количество знаков после запятой

Список 17: Пример.

```
v = 333.99343111113
print(FloatFormatter.float_round(v, 2))
'''
>>> 333.99
'''
```

Тип результата float

static float_round_signific(value, digits=15)
Округляет число с указанием количества значащих цифр.

Параметры

- **value** (float) – Входное значение
- **digits** (int) – Количество значащих цифр

Список 18: Пример.

```
v = 333.99343111113
print(FloatFormatter.float_round_signific(v, 6))
'''
>>> 333.993
'''
```

Тип результата float

static float_to_str(value, use_delimeter=False, precision=15)

Представляет значение типа float в виде строки с заданными параметрами.

Параметры

- **value** (float) – Входное значение
- **use_delimeter** (bool) – Если True, то использовать разделитель разрядов
- **precision** (int) – Количество знаков после запятой, если необходимо округление

Список 19: Пример.

```
v = 3333333333.99343111113
print(FloatFormatter.float_to_str(v))
print(FloatFormatter.float_to_str(v, True))
print(FloatFormatter.float_to_str(v, True, 4))
'''
3333333333.993431091308594
3 333 333 333,993431091308594
3 333 333 333,9934
'''
```

Тип результата str

static to_localized_string(value, locale=<PySide2.QtCore.QLocale(C, Default, Default)>)

Возвращает число в виде строки форматированный с учетом переданной локали.

Параметры

- **value** (float) – Значение которое нужно представить в виде стоки.
- **precision** – Необходимое число знаков после запятой.
- **locale** (QLocale) – Локаль в которой нужно вывести значение. По умолчанию текущая локаль.

Тип результата str

static to_localized_string_round(value, precision, locale=<PySide2.QtCore.QLocale(C, Default, Default)>)

Возвращает число в виде строки. Количество знаков после запятой задается параметром precision.

Параметры

- **value** (float) – Значение которое нужно представить в виде стоки.
- **precision** (int) – Необходимое число знаков после запятой.
- **locale** (QLocale) – Локаль в которой нужно вывести значение. По умолчанию текущая локаль.

Тип результата str

16.1.6 axipy.da

Модуль источников данных.

В данном модуле содержатся классы и методы для работы с источниками данных.

axipy.da.**provider_manager**
Готовый экземпляр открытия/создания объектов данных.

Type axipy.da.ProviderManager

axipy.da.**state_manager**
Готовый экземпляр наблюдателей за состоянием.

Type axipy.da.StateManager

axipy.da.**data_manager**
Хранилище объектов приложения.

Это то же хранилище, которое отображается в панели «Открытые данные».

Примечание: При открытии объектов данных axipy.da.ProviderManager.openfile() они автоматически попадают в каталог.

Type axipy.da.DataManager

class axipy.da.**DefaultKeys**
Идентификаторы наблюдателей по умолчанию.

Таблица 26: Атрибуты

Значение	Тип	Наименование
Selection	bool	Есть выборка
Editable	bool	Активная карта имеет редактируемый слой
SelectionEditable	bool	Карта имеет редактируемый слой и есть выделенные объекты на одном из слоев карты
SelectionEditableIsSame	bool	Карта имеет редактируемый слой и выборку на этом слое
Widget	bool	Есть активное окно
MapView	bool	Есть активное окно карты
TableView	bool	Есть активное окно таблицы
HasTables	bool	Открыта хотя бы одна таблица

class axipy.da.**ValueObserver**
Наблюдатель за одним значением.

value()
Возвращает значение.

```
# Эквивалентно
v = obs.value()
v = obs()
```

Тип результата typing.Any

setValue(value)
Устанавливает значение.

При изменении значения испускается сигнал changed.

Параметры **value** (typing.Any) – Новое значение.

property changed(value)
Сигнал об изменении значения.

Параметры **value** (typing.Any) – Новое значение.

Тип результата PySide2.QtCore.Signal

```
def print_func(value):
    print(value)

observer.changed.connect(print_func)
```

16.1.6.1 StateManager - Менеджер состояний

class axipy.da.**StateManager**
Наблюдатели за состоянием.

Примечание: Используйте готовый экземпляр этого класса axipy.da.state_manager.

Methods:

create(id[, init_value])	Создает наблюдатель за значением.
find(id)	Ищет наблюдатель с указанным идентификатором.
get(id)	Возвращает наблюдатель с указанным идентификатором.

create(id, init_value=None)
Создает наблюдатель за значением.

Параметры

- **id** (str) – Идентификатор.
- **init_value** (Optional[Any]) – Первоначальное значение.

Исключение **RuntimeError** – Если наблюдатель с указанным идентификатором уже существует.

Список 20: Пример использования

```
# Создает наблюдателя, зависящего от другого наблюдателя.
selection = state_manager.get(state_manager.Selection)
no_selection = state_manager.create('NoSelection', not selection.value())
selection.changed.connect(lambda: no_selection.setValue(not selection.
↪value()))
```

Список 21: Пример наблюдателя за количеством открытых растров

```
# Функция проверки состояния наблюдаталя.
# Проверяет присутствует ли открытый растр при изменении каталога.
def check_observer():
    def israster(obj):
        return isinstance(obj, Raster)
    rasters = list(filter( israster, data_manager.objects))
    self.__my_observer.setValue(len(rasters))
# Проверка существования наблюдателя.
# Если он не существует, то создается
if state_manager.find('MyStateManager') == None:
    self.__my_observer = state_manager.create('MyStateManager', False)
else:
    self.__my_observer = state_manager.find('MyStateManager')
# Привязка наболюдателя к событию на изменение каталога.
data_manager.updated.connect(check_observer)
# Привязка наблюдателя к кнопке с целью отслеживания состояния.
# Доступна, если открыт хотя бы один растр.
self.__action = self.create_action('Пример действия',
        icon=':/icons/share/32px/run3.png', on_click=self.show_message,
        enable_on = 'MyStateManager')
```

Тип результата ValueObserver

find(id)

Ищет наблюдатель с указанным идентификатором. Возвращает искомый наблюдатель или None, если не найдено.

Параметры **id** (Union[str, DefaultKeys]) – Идентификатор.

Тип результата Optional[ValueObserver]

get(id)

Возвращает наблюдатель с указанным идентификатором.

Параметры **id** (Union[str, DefaultKeys]) – Идентификатор.

Исключение **RuntimeError** – Если наблюдатель с указанным идентификатором не найден.

Список 22: Пример использования

```
obs = state_manager.get(state_manager.Selection)
obs2 = state_manager.get('Editable')
```

Тип результата ValueObserver

16.1.6.2 DataManager - Каталог данных

class axipy.da.**DataManager**

Хранилище объектов данных. При открытии таблицы или растра эти объекты автоматически попадают в данный каталог. Для отслеживания изменений в каталоге используются события added и removed.

Если же разработка ведется не в рамках приложения и ядро инициализировано явно посредством axipy.init_axioma(), то объект в каталог надо добавлять явно DataManager.add()

Примечание: Используйте готовый экземпляр этого класса axipy.da.data_manager.

Список 23: Пример использования.

```
# Отслеживание добавления или удаления в каталоге.
data_manager.added[str].connect(lambda n: print(f'Таблица "{n}" добавлена в␣
↪каталог'))
data_manager.removed[str].connect(lambda n: print(f'Таблица "{n}" удалена из␣
↪каталога'))
# Открываем таблицу
table = provider_manager.openfile(filepath)
# Список объектов каталога
for t in data_manager:
    print(t.name)
# Доступ по имени
'world' in data_manager
try:
    found_table = data_manager['world']
except KeyError:
    pass
# Закрываем таблицу
table.close()
# Убираем отслеживание
data_manager.added[str].disconnect()
data_manager.removed[str].disconnect()
'''
Таблица "world" добавлена в каталог
world
Таблица "world" удалена из каталога
'''
```

Methods:

add(data_object)	Добавляет объект данных в хранилище.
exists(obj)	Проверяет присутствует ли объект в каталоге.
find(name)	Производит поиск объект данных по имени.
query(query_text[, dialect])	Выполняет SQL-запрос к перечисленным таблицам.
query_hidden(query_text[, dialect])	Выполняет SQL-запрос к таблицам.
remove(data_object)	Удаляет объект данных.

continues on next page

Таблица 28 – продолжение с предыдущей страницы

remove_all()	Удаляет все объекты данных.

Attributes:

added	Signal[str] Сигнал о добавлении объекта.
all_objects	Список всех объектов, включая скрытые.
count	Количество объектов данных.
objects	Список объектов.
removed	Signal[str] Сигнал об удалении объекта.
selection	Таблица выборки, если она существует.
sql_dialect	Тип используемого диалекта по умолчанию для выполнения SQL-предложений.
tables	Список таблиц.
updated	Signal[] Сигнал об изменении количества объектов.

add(data_object)
Добавляет объект данных в хранилище.

Параметры data_object (DataObject) – Объект данных для добавления.

property added
Signal[str] Сигнал о добавлении объекта. В качестве параметра передается наименование таблицы.

Тип результата Signal

property all_objects
Список всех объектов, включая скрытые.

Тип результата List[DataObject]

property count
Количество объектов данных.

Тип результата int

exists(obj)
Проверяет присутствует ли объект в каталоге. Проверяет так-же и скрытые объекты, которые отсутствуют в общем списке.

Параметры obj (DataObject) – проверяемый объект данных.

Тип результата bool

find(name)
Производит поиск объект данных по имени.

Параметры name (str) – Имя объекта данных.

Тип результата Optional[DataObject]

Результат Искомый объект данных или None.

property objects
Список объектов.

Тип результата List[DataObject]

query(query_text, dialect=None)
Выполняет SQL-запрос к перечисленным таблицам.

Параметры

- **query_text** (str) – Текст запроса.
- **dialect** (Union[TypeSqlDialect, str, None]) – Диалект, который используется при выполнении запроса. Значение по умолчанию установлено как значение свойства sql_dialect.

Тип результата Optional[Table]

Результат Таблица, если результатом запроса является таблица.

Исключение **RuntimeError** – При возникновении ошибки.

Список 24: Пример использования, если работа ведется в рамках Аксиома .

```
filepath = 'path/to/world.tab'
# Открываем таблицу
table = provider_manager.openfile(filepath)
# Выполняем запрос
qry = data_manager.query('select * from world1 where Страна like "A%"')
# Выполняем тот-же запрос, но с явным указанием диалекта
qry = data_manager.query('select * from world1 where Страна like "A%"',␣
↪TypeSqlDialect.axioma)
```

Список 25: Пример использования, если выполняется как отдельное приложение с инициализацией axipy.init_axioma().

```
filepath = 'path/to/world.tab'
# Открываем таблицу
table = provider_manager.openfile(filepath)
# Добавляем таблицу в каталог
data_manager.add(table)
# Выполняем запрос
qry = data_manager.query('select * from world1 where Страна like "A%"')
# Выполняем тот-же запрос, но с явным указанием диалекта
qry = data_manager.query('select * from world1 where Страна like "A%"',␣
↪TypeSqlDialect.axioma)
# Если запрос qry будет использоваться при создании нового запроса, то его по␣
↪аналогии тоже
# нужно добавить в каталог
data_manager.add(qry)
# Удаление из каталога
data_manager.remove(qry)
data_manager.remove(table)
```

query_hidden(query_text, dialect=None)
Выполняет SQL-запрос к таблицам. В отличие от query() результирующий объект Table добавляется в каталог как скрытый объект. Он не учитывается в общем списке и от него из этого каталога не приходят события.

Параметры

- **query_text** (str) – Текст запроса.
- **dialect** (Union[TypeSqlDialect, str, None]) – Диалект, который используется при выполнении запроса. Значение по умолчанию установлено как sql_dialect.

Тип результата Optional[Table]

Результат Таблица, если результатом запроса является таблица.

remove(data_object)
Удаляет объект данных.

Объект данных при этом закрывается.

Параметры **data_object** (DataObject) – Объект данных для удаления.

remove_all()
Удаляет все объекты данных.

property removed
Signal[str] Сигнал об удалении объекта.

Тип результата Signal

property selection
Таблица выборки, если она существует.

См.также:

axipy.gui.selection_manager

Тип результата Optional[SelectionTable]

property sql_dialect
Тип используемого диалекта по умолчанию для выполнения SQL-предложений. Если необходимо переопределить, то для конкретного sql предложения необходимо указывать диалект явно query()

Тип результата TypeSqlDialect

Результат Тип диалекта. Возможные значения TypeSqlDialect.axioma или TypeSqlDialect.sqlite.

property tables
Список таблиц.

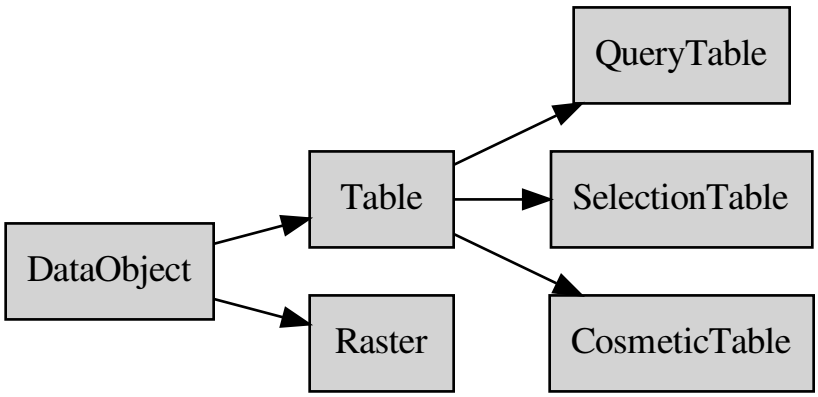
Тип результата List[Table]

property updated
Signal[] Сигнал об изменении количества объектов.

Тип результата Signal

16.1.6.3 DataObject - Объект данных

Иерархия классов:



class axipy.da.**DataObject**

Объект данных.

Открываемые объекты из источников данных представляются объектами этого типа. Возможные реализации: таблица, растр, грид, чертеж, панорама, и так далее.

Пример:

```
table = provider_manager.openfile('path/to/file.tab')
...
table.close()  # Закрывает таблицу
```

Для закрытия объекта данных можно использовать менеджер контекста - выражение `with`. В таком случае таблица будет закрыта при выходе из блока. См. `close()`.

Пример:

```
with provider_manager.openfile('path/to/file.tab') as raster:
    ...
    # При выходе из блока растр будет закрыт
```

Methods:

close()	Пытается закрыть таблицу.

Attributes:

destroyed	Сигнал оповещения об удалении объекта.

continues on next page

Таблица 31 – продолжение с предыдущей страницы

is_spatial	Признак того, что объект данных является пространственным.
name	Название объекта данных.
properties	Дополнительные свойства объекта данных.
provider	Провайдер изначального источника данных.

close()
Пытается закрыть таблицу.

Исключение RuntimeError – Ошибка закрытия таблицы.

Примечание: Объект данных не всегда может быть сразу закрыт. Например, для таблиц используется транзакционная модель редактирования и перед закрытием необходимо сохранить или отменить изменения, если они есть. См. Table.is_modified.

property destroyed
Сигнал оповещения об удалении объекта.

Тип результата Signal

property is_spatial
Признак того, что объект данных является пространственным.

Тип результата bool

property name
Название объекта данных.

Тип результата str

property properties
Дополнительные свойства объекта данных.

Тип результата dict

property provider
Провайдер изначального источника данных.

Тип результата str

16.1.6.4 Raster - Растр

class axipy.da.**Raster**

Базовые классы: axipy.da.DataObject

Растровый объект.

Methods:

close()	Пытается закрыть таблицу.
get_gcps()	Возвращает точки привязки.

Attributes:

coordsystem	Система координат растра.
destroyed	Сигнал оповещения об удалении объекта.
device_to_scene_transform	Матрица преобразования из точек на изображении (пиксели) в точки на карте.
is_spatial	Признак того, что объект данных является пространственным.
name	Название объекта данных.
properties	Дополнительные свойства объекта данных.
provider	Провайдер изначального источника данных.
scene_to_device_transform	Матрица преобразования из точек на карте в точки на изображении (пиксели).
size	Размер растра в пикселях.

close()
Пытается закрыть таблицу.

Исключение RuntimeError – Ошибка закрытия таблицы.

Примечание: Объект данных не всегда может быть сразу закрыт. Например, для таблиц используется транзакционная модель редактирования и перед закрытием необходимо сохранить или отменить изменения, если они есть. См. Table.is_modified.

property coordsystem
Система координат растра.

Тип результата Optional[CoordSystem]

property destroyed
Сигнал оповещения об удалении объекта.

Тип результата Signal

property device_to_scene_transform
Матрица преобразования из точек на изображении (пиксели) в точки на карте.

Тип результата QTransform

get_gcps()
Возвращает точки привязки.

Тип результата List[GCP]

property is_spatial
Признак того, что объект данных является пространственным.

Тип результата bool

property name
Название объекта данных.

Тип результата str

property properties
Дополнительные свойства объекта данных.

Тип результата dict

property provider
Провайдер изначального источника данных.

Тип результата str

property scene_to_device_transform
Матрица преобразования из точек на карте в точки на изображении (пиксели).

Тип результата QTransform

property size
Размер растра в пикселях.

Тип результата QSize

16.1.6.5 Table - Таблица

class axipy.da.**Table**

Базовые классы: axipy.da.DataObject

Таблица.

Менеджер контекста сохраняет изменения и закрывает таблицу.

Пример:

```
with provider_manager.openfile('path/to/file.tab') as table:
    ...
    # При выходе из блока таблица будет сохранена и закрыта
```

См.также:

commit(), DataObject.close().

Attributes:

can_redo	Возможен ли откат на один шаг вперед.
can_undo	Возможен ли откат на один шаг назад.
coordsystem	Система координат таблицы.
data_changed	Сигнал об изменении данных таблицы.
destroyed	Сигнал оповещения об удалении объекта.
is_editable	Признак того, что таблица является редактируемой.
is_modified	Таблица содержит несохраненные изменения.
is_spatial	Признак того, что объект данных является пространственным.
is_temporary	Признак того, что таблица является временной.
name	Название объекта данных.

continues on next page

Таблица 34 – продолжение с предыдущей страницы

properties	Дополнительные свойства объекта данных.
provider	Провайдер изначального источника данных.
schema	Схема таблицы.
schema_changed	Сигнал об изменении схемы таблицы.
supported_operations	Доступные операции.

Methods:

close()	Пытается закрыть таблицу.
commit()	Сохраняет изменения в таблице.
count([bbox])	Возвращает количество записей, удовлетворяющих параметрам.
insert(features)	Вставляет записи в таблицу.
items([bbox, ids])	Запрашивает записи, удовлетворяющие параметрам.
itemsByIds(ids)	Запрашивает записи по списку list с идентификаторами записей, либо перечень идентификаторов в виде списка.
itemsInObject(obj)	Запрашивает записи с фильтром по геометрическому объекту.
itemsInRect(bbox)	Запрашивает записи с фильтром по ограничивающему прямоугольнику.
redo()	Производит откат на один шаг вперед.
remove(ids)	Удаляет записи из таблицы.
restore()	Отменяет несохраненные изменения в таблице.
undo()	Производит откат на один шаг назад.
update(features)	Обновляет записи в таблице.

property can_redo
Возможен ли откат на один шаг вперед.

Тип результата bool

property can_undo
Возможен ли откат на один шаг назад.

Тип результата bool

close()
Пытается закрыть таблицу.

Исключение RuntimeError – Ошибка закрытия таблицы.

Примечание: Объект данных не всегда может быть сразу закрыт. Например, для таблиц используется транзакционная модель редактирования и перед закрытием необходимо сохранить или отменить изменения, если они есть. См. Table.is_modified.

---

commit()

Сохраняет изменения в таблице.

Если таблица не содержит несохраненные изменения, то команда игнорируется.

Исключение RuntimeError – Невозможно сохранить изменения.

property coordsystem
Система координат таблицы.

Тип результата Optional[CoordSystem]

count(bbox=None)
Возвращает количество записей, удовлетворяющих параметрам.

Данный метод является наиболее предпочтительным для оценки количества записей. При этом используется наиболее оптимальный вариант выполнения запроса для каждого конкретного провайдера данных.

Параметры bbox (Union[Rect, QRectF, tuple, None]) – Ограничивающий прямоугольник.

Тип результата int

Результат Количество записей.

property data_changed
Сигнал об изменении данных таблицы. Испускается когда были изменены данные таблицы.

Список 26: Пример подписки на изменение таблицы.

```
table = provider_manager.openfile(filepath)
table.data_changed.connect(lambda : print('Таблица была изменена.'))
```

Тип результата Signal

property destroyed
Сигнал оповещения об удалении объекта.

Тип результата Signal

insert(features)
Вставляет записи в таблицу.

Параметры features (Union[Iterator[Feature], Feature]) – Записи для вставки.

property is_editable
Признак того, что таблица является редактируемой.

Тип результата bool

property is_modified
Таблица содержит несохраненные изменения.

Тип результата bool

property is_spatial
Признак того, что объект данных является пространственным.

Тип результата bool

property is_temporary
Признак того, что таблица является временной.

Тип результата bool

items(bbox=None, ids=None)
Запрашивает записи, удовлетворяющие параметрам. В качестве фильтра может быть указан либо ограничивающий прямоугольник, либо перечень идентификаторов в виде списка.

Параметры

- **bbox** (Union[Rect, QRectF, tuple, None]) – Ограничивающий прямоугольник.
- **ids** (Optional[List[int]]) – Список идентификаторов.

Тип результата Iterator[Feature]

Результат Итератор по записям.

itemsByIds(ids)
Запрашивает записи по списку list с идентификаторами записей, либо перечень идентификаторов в виде списка. Идентификаторы несохраненных записей имеют отрицательные значения.

Параметры **ids** (List[int]) – Список идентификаторов.

Тип результата Iterator[Feature]

Результат Итератор по записям.

Список 27: Пример

```
table_world = provider_manager.openfile(filepath)
# Пример запроса по списку идентификаторов.
items = table_world.itemsByIds([11,27,41,163,203])
for f in items:
    print('Feature id={}. Страна={}'.format(f.id, f['Страна']))
# Просмотр идентификаторов всех записей, включая несохраненные
for f in  table_world.items():
    print( f.id, f['Страна'] )
# Получение несохраненных записей совместно с сохраненными
items_new = table_world.itemsByIds([-1,-12,27])
for f in items_new:
    print('Feature id={}. Страна={}'.format(f.id, f['Страна']))
```

itemsInObject(obj)
Запрашивает записи с фильтром по геометрическому объекту.

Параметры **obj** (Geometry) – Геометрия. Если для нее не задана СК, используется СК таблицы.

Тип результата Iterator[Feature]

Результат Итератор по записям.

Список 28: Пример запроса по полигону

```
table_world = provider_manager.openfile(filepath)
v = 2000000
polygon = Polygon((-v, -v), (-v, v), (v, v), (v, -v))
```

(continues on next page)

(продолжение с предыдущей страницы)

```
items = table_world.itemsInObject(polygon)
for f in items:
    print('Feature id={}. Страна={}'.format(f.id, f['Страна']))
```

itemsInRect(bbox)

Запрашивает записи с фильтром по ограничивающему прямоугольнику.

Параметры bbox (Union[Rect, QRectF, tuple]) – Ограничивающий прямоугольник.

Тип результата Iterator[Feature]

Результат Итератор по записям.

Список 29: Пример запроса (таблица в проекции Робинсона)

```
table_world = provider_manager.openfile(filepath)
v = 2000000
items = table_world.itemsInRect(Rect(-v, -v, v, v))
for f in items:
    print('Feature id={}. Страна={}'.format(f.id, f['Страна']))
```

property name

Название объекта данных.

Тип результата str

property properties

Дополнительные свойства объекта данных.

Тип результата dict

property provider

Провайдер изначального источника данных.

Тип результата str

redo()

Производит откат на один шаг вперед. При этом возвращается состояние до последней отмены.

remove(ids)

Удаляет записи из таблицы.

Параметры ids (Union[int, Iterator[int]]) – Идентификаторы записей для удаления.

restore()

Отменяет несохраненные изменения в таблице.

Если таблица не содержит несохраненные изменения, то команда игнорируется.

property schema

Схема таблицы.

Тип результата Schema

property schema_changed

Сигнал об изменении схемы таблицы. Испускается когда была изменена структура таблицы.

Тип результата Signal

property supported_operations
Доступные операции.

Список 30: Пример использования

```
flags = table.supported_operations
assert flags == SupportedOperations.ReadWrite
assert flags & SupportedOperations.Insert
assert SupportedOperations.Write in flags
```

Тип результата SupportedOperations

undo()
Производит откат на один шаг назад.

update(features)
Обновляет записи в таблице.

Параметры features (Union[Iterator[Feature], Feature]) – Записи для обновления.

При обновлении проверяется Feature.id. Если запись с таким идентификатором не найдена, то она пропускается.

Список 31: Пример использования

```
modified_feature = Feature({'attr_name': 'new_value'}, id=1)
table.update(modified_feature)
table.commit()
```

См.также:

Feature.id, commit(), is_modified.

16.1.6.6 QueryTable - SQL запрос.

class axipy.da.**QueryTable**

Базовые классы: axipy.da.Table

Таблица, построенная на основе SQL запроса.

Пример:

```
table = provider_manager.openfile('world.tab')
query = data_manager.query('select * from world')
```

Attributes:

can_redo	Возможен ли откат на один шаг вперед.
can_undo	Возможен ли откат на один шаг назад.
coordsystem	Система координат таблицы.
data_changed	Сигнал об изменении данных таблицы.
destroyed	Сигнал оповещения об удалении объекта.

continues on next page

Таблица 36 – продолжение с предыдущей страницы

is_editable	Признак того, что таблица является редактируемой.
is_modified	Таблица содержит несохраненные изменения.
is_spatial	Признак того, что объект данных является пространственным.
is_temporary	Признак того, что таблица является временной.
name	Название объекта данных.
properties	Дополнительные свойства объекта данных.
provider	Провайдер изначального источника данных.
schema	Схема таблицы.
schema_changed	Сигнал об изменении схемы таблицы.
sql_text	Текст SQL запроса.
supported_operations	Доступные операции.

Methods:

close()	Пытается закрыть таблицу.
commit()	Сохраняет изменения в таблице.
count([bbox])	Возвращает количество записей, удовлетворяющих параметрам.
insert(features)	Вставляет записи в таблицу.
items([bbox, ids])	Запрашивает записи, удовлетворяющие параметрам.
itemsByIds(ids)	Запрашивает записи по списку list с идентификаторами записей, либо перечень идентификаторов в виде списка.
itemsInObject(obj)	Запрашивает записи с фильтром по геометрическому объекту.
itemsInRect(bbox)	Запрашивает записи с фильтром по ограничивающему прямоугольнику.
redo()	Производит откат на один шаг вперед.
remove(ids)	Удаляет записи из таблицы.
restore()	Отменяет несохраненные изменения в таблице.
undo()	Производит откат на один шаг назад.
update(features)	Обновляет записи в таблице.

property can_redo
Возможен ли откат на один шаг вперед.

Тип результата bool

property can_undo
Возможен ли откат на один шаг назад.

Тип результата bool

close()

Пытается закрыть таблицу.

Исключение RuntimeError – Ошибка закрытия таблицы.

Примечание: Объект данных не всегда может быть сразу закрыт. Например, для таблиц используется транзакционная модель редактирования и перед закрытием необходимо сохранить или отменить изменения, если они есть. См. Table.is_modified.

---

commit()
Сохраняет изменения в таблице.

Если таблица не содержит несохраненные изменения, то команда игнорируется.

Исключение RuntimeError – Невозможно сохранить изменения.

property coordsystem
Система координат таблицы.

Тип результата Optional[CoordSystem]

count(bbox=None)
Возвращает количество записей, удовлетворяющих параметрам.

Данный метод является наиболее предпочтительным для оценки количества записей. При этом используется наиболее оптимальный вариант выполнения запроса для каждого конкретного провайдера данных.

Параметры bbox (Union[Rect, QRectF, tuple, None]) – Ограничивающий прямоугольник.

Тип результата int

Результат Количество записей.

property data_changed
Сигнал об изменении данных таблицы. Испускается когда были изменены данные таблицы.

Список 32: Пример подписки на изменение таблицы.

```
table = provider_manager.openfile(filepath)
table.data_changed.connect(lambda : print('Таблица была изменена.'))
```

Тип результата Signal

property destroyed
Сигнал оповещения об удалении объекта.

Тип результата Signal

insert(features)
Вставляет записи в таблицу.

Параметры features (Union[Iterator[Feature], Feature]) – Записи для вставки.

property is_editable
Признак того, что таблица является редактируемой.

Тип результата bool

property is_modified
Таблица содержит несохраненные изменения.

Тип результата bool

property is_spatial
Признак того, что объект данных является пространственным.

Тип результата bool

property is_temporary
Признак того, что таблица является временной.

Тип результата bool

items(bbox=None, ids=None)
Запрашивает записи, удовлетворяющие параметрам. В качестве фильтра может быть указан либо ограничивающий прямоугольник, либо перечень идентификаторов в виде списка.

Параметры

- **bbox** (Union[Rect, QRectF, tuple, None]) – Ограничивающий прямоугольник.
- **ids** (Optional[List[int]]) – Список идентификаторов.

Тип результата Iterator[Feature]

Результат Итератор по записям.

itemsByIds(ids)
Запрашивает записи по списку list с идентификаторами записей, либо перечень идентификаторов в виде списка. Идентификаторы несохраненных записей имеют отрицательные значения.

Параметры **ids** (List[int]) – Список идентификаторов.

Тип результата Iterator[Feature]

Результат Итератор по записям.

Список 33: Пример

```
table_world = provider_manager.openfile(filepath)
# Пример запроса по списку идентификаторов.
items = table_world.itemsByIds([11,27,41,163,203])
for f in items:
    print('Feature id={}. Страна={}'.format(f.id, f['Страна']))
# Просмотр идентификаторов всех записей, включая несохраненные
for f in  table_world.items():
    print( f.id, f['Страна'] )
# Получение несохраненных записей совместно с сохраненными
items_new = table_world.itemsByIds([-1,-12,27])
for f in items_new:
    print('Feature id={}. Страна={}'.format(f.id, f['Страна']))
```

itemsInObject(obj)
Запрашивает записи с фильтром по геометрическому объекту.

Параметры **obj** (Geometry) – Геометрия. Если для нее не задана СК, используется СК таблицы.

Тип результата Iterator[Feature]

Результат Итератор по записям.

Список 34: Пример запроса по полигону

```
table_world = provider_manager.openfile(filepath)
v = 2000000
polygon = Polygon((-v, -v), (-v, v), (v, v), (v, -v))
items = table_world.itemsInObject(polygon)
for f in items:
    print('Feature id={}. Страна={}'.format(f.id, f['Страна']))
```

itemsInRect(bbox)

Запрашивает записи с фильтром по ограничивающему прямоугольнику.

Параметры bbox (Union[Rect, QRectF, tuple]) – Ограничивающий прямоугольник.

Тип результата Iterator[Feature]

Результат Итератор по записям.

Список 35: Пример запроса (таблица в проекции Робинсона)

```
table_world = provider_manager.openfile(filepath)
v = 2000000
items = table_world.itemsInRect(Rect(-v, -v, v, v))
for f in items:
    print('Feature id={}. Страна={}'.format(f.id, f['Страна']))
```

property name

Название объекта данных.

Тип результата str

property properties

Дополнительные свойства объекта данных.

Тип результата dict

property provider

Провайдер изначального источника данных.

Тип результата str

redo()

Производит откат на один шаг вперед. При этом возвращается состояние до последней отмены.

remove(ids)

Удаляет записи из таблицы.

Параметры ids (Union[int, Iterator[int]]) – Идентификаторы записей для удаления.

restore()

Отменяет несохраненные изменения в таблице.

Если таблица не содержит несохраненные изменения, то команда игнорируется.

property schema
Схема таблицы.

Тип результата Schema

property schema_changed
Сигнал об изменении схемы таблицы. Испускается когда была изменена структура таблицы.

Тип результата Signal

property sql_text
Текст SQL запроса.

Тип результата str

property supported_operations
Доступные операции.

Список 36: Пример использования

```
flags = table.supported_operations
assert flags == SupportedOperations.ReadWrite
assert flags & SupportedOperations.Insert
assert SupportedOperations.Write in flags
```

Тип результата SupportedOperations

undo()
Производит откат на один шаг назад.

update(features)
Обновляет записи в таблице.

Параметры features (Union[Iterator[Feature], Feature]) – Записи для обновления.

При обновлении проверяется Feature.id. Если запись с таким идентификатором не найдена, то она пропускается.

Список 37: Пример использования

```
modified_feature = Feature({'attr_name': 'new_value'}, id=1)
table.update(modified_feature)
table.commit()
```

См.также:

Feature.id, commit(), is_modified.

16.1.6.7 SelectionTable - Таблица с текущей выборкой.

class axipy.da.**SelectionTable**

Базовые классы: axipy.da.Table

Таблица, построенная на основе текущей выборки. Носит временный характер. Создается, когда в выборку добавляются объекты и удаляется, когда выборка очищается.

Пример доступа к выборки с получением идентификаторов:

```
for f in data_manager.selection.items():
    print('>>>', f.id)
```

Attributes:

base_table	Таблица, на которой основана данная выборка.
can_redo	Возможен ли откат на один шаг вперед.
can_undo	Возможен ли откат на один шаг назад.
coordsystem	Система координат таблицы.
data_changed	Сигнал об изменении данных таблицы.
destroyed	Сигнал оповещения об удалении объекта.
is_editable	Признак того, что таблица является редактируемой.
is_modified	Таблица содержит несохраненные изменения.
is_spatial	Признак того, что объект данных является пространственным.
is_temporary	Признак того, что таблица является временной.
name	Название объекта данных.
properties	Дополнительные свойства объекта данных.
provider	Провайдер изначального источника данных.
schema	Схема таблицы.
schema_changed	Сигнал об изменении схемы таблицы.
supported_operations	Доступные операции.

Methods:

close()	Пытается закрыть таблицу.
commit()	Сохраняет изменения в таблице.
count([bbox])	Возвращает количество записей, удовлетворяющих параметрам.
insert(features)	Вставляет записи в таблицу.
items([bbox, ids])	Запрашивает записи, удовлетворяющие параметрам.
itemsByIds(ids)	Запрашивает записи по списку list с идентификаторами записей, либо перечень идентификаторов в виде списка.
itemsInObject(obj)	Запрашивает записи с фильтром по геометрическому объекту.
itemsInRect(bbox)	Запрашивает записи с фильтром по ограничивающему прямоугольнику.
redo()	Производит откат на один шаг вперед.
remove(ids)	Удаляет записи из таблицы.
restore()	Отменяет несохраненные изменения в таблице.
undo()	Производит откат на один шаг назад.
update(features)	Обновляет записи в таблице.

property base_table
Таблица, на которой основана данная выборка.

Тип результата Table

property can_redo
Возможен ли откат на один шаг вперед.

Тип результата bool

property can_undo
Возможен ли откат на один шаг назад.

Тип результата bool

close()
Пытается закрыть таблицу.

Исключение RuntimeError – Ошибка закрытия таблицы.

Примечание: Объект данных не всегда может быть сразу закрыт. Например, для таблиц используется транзакционная модель редактирования и перед закрытием необходимо сохранить или отменить изменения, если они есть. См. Table.is_modified.

---

commit()
Сохраняет изменения в таблице.

Если таблица не содержит несохраненные изменения, то команда игнорируется.

Исключение RuntimeError – Невозможно сохранить изменения.

property coordsystem
Система координат таблицы.

Тип результата Optional[CoordSystem]

count(bbox=None)
Возвращает количество записей, удовлетворяющих параметрам.

Данный метод является наиболее предпочтительным для оценки количества записей. При этом используется наиболее оптимальный вариант выполнения запроса для каждого конкретного провайдера данных.

Параметры **bbox** (Union[Rect, QRectF, tuple, None]) – Ограничивающий прямоугольник.

Тип результата int

Результат Количество записей.

property data_changed
Сигнал об изменении данных таблицы. Испускается когда были изменены данные таблицы.

Список 38: Пример подписки на изменение таблицы.

```
table = provider_manager.openfile(filepath)
table.data_changed.connect(lambda : print('Таблица была изменена.'))
```

Тип результата Signal

property destroyed
Сигнал оповещения об удалении объекта.

Тип результата Signal

insert(features)
Вставляет записи в таблицу.

Параметры **features** (Union[Iterator[Feature], Feature]) – Записи для вставки.

property is_editable
Признак того, что таблица является редактируемой.

Тип результата bool

property is_modified
Таблица содержит несохраненные изменения.

Тип результата bool

property is_spatial
Признак того, что объект данных является пространственным.

Тип результата bool

property is_temporary
Признак того, что таблица является временной.

Тип результата bool

items(bbox=None, ids=None)
Запрашивает записи, удовлетворяющие параметрам. В качестве фильтра

может быть указан либо ограничивающий прямоугольник, либо перечень идентификаторов в виде списка.

Параметры

- **bbox** (Union[Rect, QRectF, tuple, None]) – Ограничивающий прямоугольник.
- **ids** (Optional[List[int]]) – Список идентификаторов.

Тип результата Iterator[Feature]

Результат Итератор по записям.

itemsByIds(ids)

Запрашивает записи по списку list с идентификаторами записей, либо перечень идентификаторов в виде списка. Идентификаторы несохраненных записей имеют отрицательные значения.

Параметры **ids** (List[int]) – Список идентификаторов.

Тип результата Iterator[Feature]

Результат Итератор по записям.

Список 39: Пример

```
table_world = provider_manager.openfile(filepath)
# Пример запроса по списку идентификаторов.
items = table_world.itemsByIds([11,27,41,163,203])
for f in items:
    print('Feature id={}. Страна={}'.format(f.id, f['Страна']))
# Просмотр идентификаторов всех записей, включая несохраненные
for f in  table_world.items():
    print( f.id, f['Страна'] )
# Получение несохраненных записей совместно с сохраненными
items_new = table_world.itemsByIds([-1,-12,27])
for f in items_new:
    print('Feature id={}. Страна={}'.format(f.id, f['Страна']))
```

itemsInObject(obj)

Запрашивает записи с фильтром по геометрическому объекту.

Параметры **obj** (Geometry) – Геометрия. Если для нее не задана СК, используется СК таблицы.

Тип результата Iterator[Feature]

Результат Итератор по записям.

Список 40: Пример запроса по полигону

```
table_world = provider_manager.openfile(filepath)
v = 2000000
polygon = Polygon((-v, -v), (-v, v), (v, v), (v, -v))
items = table_world.itemsInObject(polygon)
for f in items:
    print('Feature id={}. Страна={}'.format(f.id, f['Страна']))
```

itemsInRect(bbox)

Запрашивает записи с фильтром по ограничивающему прямоугольнику.

Параметры bbox (Union[Rect, QRectF, tuple]) – Ограничивающий прямоугольник.

Тип результата Iterator[Feature]

Результат Итератор по записям.

Список 41: Пример запроса (таблица в проекции Робинсона)

```
table_world = provider_manager.openfile(filepath)
v = 2000000
items = table_world.itemsInRect(Rect(-v, -v, v, v))
for f in items:
    print('Feature id={}. Страна={}'.format(f.id, f['Страна']))
```

property name
Название объекта данных.

Тип результата str

property properties
Дополнительные свойства объекта данных.

Тип результата dict

property provider
Провайдер изначального источника данных.

Тип результата str

redo()
Производит откат на один шаг вперед. При этом возвращается состояние до последней отмены.

remove(ids)
Удаляет записи из таблицы.

Параметры ids (Union[int, Iterator[int]]) – Идентификаторы записей для удаления.

restore()
Отменяет несохраненные изменения в таблице.

Если таблица не содержит несохраненные изменения, то команда игнорируется.

property schema
Схема таблицы.

Тип результата Schema

property schema_changed
Сигнал об изменении схемы таблицы. Испускается когда была изменена структура таблицы.

Тип результата Signal

property supported_operations
Доступные операции.

Список 42: Пример использования

```
flags = table.supported_operations
assert flags == SupportedOperations.ReadWrite
assert flags & SupportedOperations.Insert
assert SupportedOperations.Write in flags
```

Тип результата SupportedOperations

undo()
Производит откат на один шаг назад.

update(features)
Обновляет записи в таблице.

Параметры features (Union[Iterator[Feature], Feature]) – Записи для обновления.

При обновлении проверяется Feature.id. Если запись с таким идентификатором не найдена, то она пропускается.

Список 43: Пример использования

```
modified_feature = Feature({'attr_name': 'new_value'}, id=1)
table.update(modified_feature)
table.commit()
```

См.также:

Feature.id, commit(), is_modified.

16.1.6.8 CosmeticTable - Таблица с данными косметического слоя.

class axipy.da.**CosmeticTable**

Базовые классы: axipy.da.Table

Объект данных косметического слоя axipy.render.CosmeticLayer

Attributes:

can_redo	Возможен ли откат на один шаг вперед.
can_undo	Возможен ли откат на один шаг назад.
coordsystem	Система координат таблицы.
data_changed	Сигнал об изменении данных таблицы.
destroyed	Сигнал оповещения об удалении объекта.
is_editable	Признак того, что таблица является редактируемой.
is_modified	Таблица содержит несохраненные изменения.
is_spatial	Признак того, что объект данных является пространственным.
is_temporary	Признак того, что таблица является временной.

continues on next page

Таблица 40 – продолжение с предыдущей страницы

name	Название объекта данных.
properties	Дополнительные свойства объекта данных.
provider	Провайдер изначального источника данных.
schema	Схема таблицы.
schema_changed	Сигнал об изменении схемы таблицы.
supported_operations	Доступные операции.

Methods:

close()	Пытается закрыть таблицу.
commit()	Сохраняет изменения в таблице.
count([bbox])	Возвращает количество записей, удовлетворяющих параметрам.
insert(features)	Вставляет записи в таблицу.
items([bbox, ids])	Запрашивает записи, удовлетворяющие параметрам.
itemsByIds(ids)	Запрашивает записи по списку list с идентификаторами записей, либо перечень идентификаторов в виде списка.
itemsInObject(obj)	Запрашивает записи с фильтром по геометрическому объекту.
itemsInRect(bbox)	Запрашивает записи с фильтром по ограничивающему прямоугольнику.
redo()	Производит откат на один шаг вперед.
remove(ids)	Удаляет записи из таблицы.
restore()	Отменяет несохраненные изменения в таблице.
undo()	Производит откат на один шаг назад.
update(features)	Обновляет записи в таблице.

property can_redo
Возможен ли откат на один шаг вперед.

Тип результата bool

property can_undo
Возможен ли откат на один шаг назад.

Тип результата bool

close()
Пытается закрыть таблицу.

Исключение RuntimeError – Ошибка закрытия таблицы.

Примечание: Объект данных не всегда может быть сразу закрыт. Например, для таблиц используется транзакционная модель редактирования и перед закрытием необходимо сохранить или отменить изменения, если они есть. См. Table.is_modified.

---

commit()
Сохраняет изменения в таблице.

Если таблица не содержит несохраненные изменения, то команда игнорируется.

Исключение RuntimeError – Невозможно сохранить изменения.

property coordsystem
Система координат таблицы.

Тип результата Optional[CoordSystem]

count(bbox=None)
Возвращает количество записей, удовлетворяющих параметрам.

Данный метод является наиболее предпочтительным для оценки количества записей. При этом используется наиболее оптимальный вариант выполнения запроса для каждого конкретного провайдера данных.

Параметры bbox (Union[Rect, QRectF, tuple, None]) – Ограничивающий прямоугольник.

Тип результата int

Результат Количество записей.

property data_changed
Сигнал об изменении данных таблицы. Испускается когда были изменены данные таблицы.

Список 44: Пример подписки на изменение таблицы.

```
table = provider_manager.openfile(filepath)
table.data_changed.connect(lambda : print('Таблица была изменена.'))
```

Тип результата Signal

property destroyed
Сигнал оповещения об удалении объекта.

Тип результата Signal

insert(features)
Вставляет записи в таблицу.

Параметры features (Union[Iterator[Feature], Feature]) – Записи для вставки.

property is_editable
Признак того, что таблица является редактируемой.

Тип результата bool

property is_modified
Таблица содержит несохраненные изменения.

Тип результата bool

property is_spatial
Признак того, что объект данных является пространственным.

Тип результата bool

property is_temporary
Признак того, что таблица является временной.

Тип результата bool

items(bbox=None, ids=None)
Запрашивает записи, удовлетворяющие параметрам. В качестве фильтра может быть указан либо ограничивающий прямоугольник, либо перечень идентификаторов в виде списка.

Параметры

- **bbox** (Union[Rect, QRectF, tuple, None]) – Ограничивающий прямоугольник.
- **ids** (Optional[List[int]]) – Список идентификаторов.

Тип результата Iterator[Feature]

Результат Итератор по записям.

itemsByIds(ids)
Запрашивает записи по списку list с идентификаторами записей, либо перечень идентификаторов в виде списка. Идентификаторы несохраненных записей имеют отрицательные значения.

Параметры **ids** (List[int]) – Список идентификаторов.

Тип результата Iterator[Feature]

Результат Итератор по записям.

Список 45: Пример

```
table_world = provider_manager.openfile(filepath)
# Пример запроса по списку идентификаторов.
items = table_world.itemsByIds([11,27,41,163,203])
for f in items:
    print('Feature id={}. Страна={}'.format(f.id, f['Страна']))
# Просмотр идентификаторов всех записей, включая несохраненные
for f in  table_world.items():
    print( f.id, f['Страна'] )
# Получение несохраненных записей совместно с сохраненными
items_new = table_world.itemsByIds([-1,-12,27])
for f in items_new:
    print('Feature id={}. Страна={}'.format(f.id, f['Страна']))
```

itemsInObject(obj)
Запрашивает записи с фильтром по геометрическому объекту.

Параметры **obj** (Geometry) – Геометрия. Если для нее не задана СК, используется СК таблицы.

Тип результата Iterator[Feature]

Результат Итератор по записям.

Список 46: Пример запроса по полигону

```
table_world = provider_manager.openfile(filepath)
v = 2000000
polygon = Polygon((-v, -v), (-v, v), (v, v), (v, -v))
```

(continues on next page)

(продолжение с предыдущей страницы)

```
items = table_world.itemsInObject(polygon)
for f in items:
    print('Feature id={}. Страна={}'.format(f.id, f['Страна']))
```

itemsInRect(bbox)

Запрашивает записи с фильтром по ограничивающему прямоугольнику.

Параметры bbox (Union[Rect, QRectF, tuple]) – Ограничивающий прямоугольник.

Тип результата Iterator[Feature]

Результат Итератор по записям.

Список 47: Пример запроса (таблица в проекции Робинсона)

```
table_world = provider_manager.openfile(filepath)
v = 2000000
items = table_world.itemsInRect(Rect(-v, -v, v, v))
for f in items:
    print('Feature id={}. Страна={}'.format(f.id, f['Страна']))
```

property name

Название объекта данных.

Тип результата str

property properties

Дополнительные свойства объекта данных.

Тип результата dict

property provider

Провайдер изначального источника данных.

Тип результата str

redo()

Производит откат на один шаг вперед. При этом возвращается состояние до последней отмены.

remove(ids)

Удаляет записи из таблицы.

Параметры ids (Union[int, Iterator[int]]) – Идентификаторы записей для удаления.

restore()

Отменяет несохраненные изменения в таблице.

Если таблица не содержит несохраненные изменения, то команда игнорируется.

property schema

Схема таблицы.

Тип результата Schema

property schema_changed

Сигнал об изменении схемы таблицы. Испускается когда была изменена структура таблицы.

Тип результата Signal

property supported_operations
Доступные операции.

Список 48: Пример использования

```
flags = table.supported_operations
assert flags == SupportedOperations.ReadWrite
assert flags & SupportedOperations.Insert
assert SupportedOperations.Write in flags
```

Тип результата SupportedOperations

undo()
Производит откат на один шаг назад.

update(features)
Обновляет записи в таблице.

Параметры features (Union[Iterator[Feature], Feature]) – Записи для обновления.

При обновлении проверяется Feature.id. Если запись с таким идентификатором не найдена, то она пропускается.

Список 49: Пример использования

```
modified_feature = Feature({'attr_name': 'new_value'}, id=1)
table.update(modified_feature)
table.commit()
```

См.также:

Feature.id, commit(), is_modified.

16.1.6.9 SupportedOperations - Доступные операции

class axipy.da.**SupportedOperations**(value)
Флаги доступных операций.

Реализованы как перечисление enum.IntFlag и поддерживают побитовые операции.

Атрибут	Значение
Empty	0
Insert	1
Update	2
Delete	4
Write	Insert | Update | Delete = 7
Read	8
ReadWrite	Read | Write = 15

Список 50: Пример использования

```
flags = table.supported_operations
assert flags == SupportedOperations.ReadWrite
assert flags & SupportedOperations.Insert
assert SupportedOperations.Write in flags
```

См.также:

enum.IntFlag, axipy.da.Table.supported_operations

16.1.6.10 Feature - Запись в таблице

class axipy.da.**Feature**(properties={}, geometry=None, style=None, id=None, **kwargs)

Запись в таблице.

Работа с записью похожа на работу со словарем dict. Но также допускает обращение по индексу.

Список 51: Примеры.

```
feature = Feature({'attr_name':'value'}, geometry=Point(10, 10), style=PointStyle.
↪create_mi_compat(35, 0))
# Количество атрибутов
count = len(feature)
# Запись значения по ключу
feature['attr_name'] = 'new_value'
# Запись значения по индексу
feature[0] = 'another_value'
# Чтение значения по ключу
value = feature['attr_name']
# Чтение значения по индексу
another_value = feature[0]
# Проверка наличия атрибута по ключу
'attr_name' in feature
# Проверка наличия атрибута по индексу
5 in feature
# Значения атрибутов можно задать словарем или именованными аргументами:
feature2 = Feature({'name1': 'value1', 'name2': 'value2'})
# Это эквивалентно
feature2 = Feature(name1='value1', name2='value2')
# Получение стиля оформления для геометрии
style = feature.style
# Установка нового стиля для геометрии
feature.style = style
# Получение геометрии
point = feature.geometry
# Установка нового значения для геометрии
feature.geometry = Point(20, 20)
# Просмотр всех наименований и значений атрибутов
for key, value in feature.items():
    print(('{} = {}').format(key, value))
'''
>>> attr_name = value
>>> +geometry = Point pos=(10.0 10.0)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
>>> +style = PointStyle Symbol (35, 0, 8)
'''
```

Параметры

- **properties** (dict) – Значения атрибутов.
- **geometry** (Optional[Geometry]) – Геометрия.
- **style** (Optional[Style]) – Стиль.
- **id** (Optional[int]) – Идентификатор.
- ****kwargs** – Значения атрибутов.

Примечание: Для доступа к геометрическому атрибуту и стилю по наименованию можно использовать предопределенные идентификаторы +geometry и +style соответственно:

- GEOMETRY_ATTR=+geometry
- STYLE_ATTR=+style

Attributes:

geometry	Геометрия записи.
id	Идентификатор записи в таблице.
style	Стиль записи.

Methods:

get(key[, default])	Возвращает значение заданного атрибута.
has_geometry()	Проверяет, имеет ли запись атрибут с геометрией.
has_style()	Проверяет, имеет ли запись атрибут со стилем.
items()	Возвращает список пар имя - значение.
keys()	Возвращает список имен атрибутов.
to_geojson()	Представляет запись в виде, похожем на „GeoJSON“.
values()	Возвращает список значений атрибутов.

property geometry

Геометрия записи.

См.также:

Feature.has_geometry(), GEOMETRY_ATTR

Тип результата Optional[Geometry]

Результат Значение геометрического атрибута; или None, если

значение пустое или отсутствует.

get(key, default=None)
Возвращает значение заданного атрибута.

Параметры

- **key** (str) – Имя атрибута.
- **default** (Optional[Any]) – Значение по умолчанию.

Результат Искомое значение, или значение по умолчанию, если заданный атрибут отсутствует.

has_geometry()
Проверяет, имеет ли запись атрибут с геометрией.

Тип результата bool

has_style()
Проверяет, имеет ли запись атрибут со стилем.

Тип результата bool

property id
Идентификатор записи в таблице.

Несохраненные записи в таблице будут иметь отрицательное значение.

См.также:

Table.is_modified, Table.commit()

Тип результата int

Результат 0 если идентификатор не задан.

items()
Возвращает список пар имя - значение.

Тип результата List[tuple]

keys()
Возвращает список имен атрибутов.

Тип результата List[str]

property style
Стиль записи.

См.также:

Feature.has_style(), STYLE_ATTR

Тип результата Optional[Style]

Результат Значение атрибута со стилем; или None, если значение пустое или отсутствует.

to_geojson()
Представляет запись в виде, похожем на „GeoJSON“.

Тип результата dict

values()
Возвращает список значений атрибутов.

Тип результата List

16.1.6.11 Schema - Схема таблицы

class axipy.da.**Schema**(*attributes, coordsystem=None)

Базовые классы: list

Схема таблицы. Представляет собой список атрибутов axipy.da.Attribute. Организован в виде list и свойством coordsystem. При задании coordsystem создается геометрический атрибут.

Параметры

- ***attributes** – Атрибуты.
- **coordsystem** (Union[str, CoordSystem, None]) – Система координат для геометрического атрибута. Может быть задана или в виде строки (подробнее axipy.cs.CoordSystem.from_string()) или как объект СК axipy.cs.CoordSystem.

Список 52: Пример создания

```
schema = Schema(
    Attribute.string('one', 25),
    Attribute.integer('two'),
    Attribute.decimal('three'),
    coordsystem='prj:Earth Projection 12, 62, "m", 0'
)
```

Список 53: Пример создания из списка

```
attrs = [Attribute.string('one', 25), Attribute.integer(
    'two'), Attribute.decimal('three')]
# распаковывается список атрибутов
schema = Schema(*attrs, coordsystem='prj:Earth Projection 12, 62, "m", 0')
```

Имеет стандартные функции работы со списком.

Список 54: Пример операций

```
count = len(schema)  # количество атрибутов
attribute = schema[2]  # получение атрибута по индексу
schema[2] = Attribute.string('attr', 30)  # изменяет
del schema[2]  # удаляет
schema.append(Attribute.string('new_attr'))  # добавляет в конец
schema.insert(2, Attribute.integer('num_attr'))  # добавляет по индексу
index = schema.index('new_attr')  # ищет по имени
assert 'new_attr' in schema  # проверяет существование
schema.remove('new_attr')  # удаляет по имени
a = schema.by_name('attr') # Получение атрибута по его имени
```

Attributes:

attribute_names	Возвращает список имен атрибутов.
coordsystem	Система координат.

Methods:

by_name(n)	Возвращает атрибут по его имени.
insert(index, attr)	Вставляет атрибут.

property attribute_names
Возвращает список имен атрибутов.

Тип результата List[str]

by_name(n)
Возвращает атрибут по его имени. Если такого имени не существует, возвращает None.

Параметры **n** (str) – Наименование атрибута.

Тип результата Attribute

property coordsystem
Система координат.

Тип результата Optional[CoordSystem]

Результат None, если СК отсутствует.

См.также:

axipy.da.Table.is_spatial

Список 55: Пример использования

```
if schema.coordsystem is not None:  # проверяет существование
    pass
crs_string = schema.coordsystem  # получает
schema.coordsystem = 'epsg:4326'  # изменяет
schema.coordsystem = None  # удаляет
```

insert(index, attr)
Вставляет атрибут.

Параметры

- **index** (int) – Индекс, по которому производится вставка.
- **attr** (Attribute) – Атрибут.

16.1.6.12 TabFile - Файл TAB

class axipy.da.**TabFile**
Класс поддержки файла TAB формата MapInfo.

generate_tab(data_object, out_file, override=True)
Генерирует файл TAB для переданного открытого объекта, если такую возможность поддерживает провайдер данных.

Параметры

- **data_object** (DataObject) – открытый объект данных, для которого необходимо создать файл TAB.
- **out_file** (str) – Имя файла с расширением tab. Как вариант, можно использовать результат suggest_tab_name().
- **override** (bool) – Перезаписывать файл. Если установлено False и файл существует, будет выброшено исключение FileExistsError

Тип результата bool

Результат Возвращает True, если успешно.

Создание TAB файла для открытой таблицы или растра:

```
from axipy import TabFile, provider_manager
filepath = 'world.tif'
out_file_name = 'world.tab'
table = provider_manager.openfile(filepath)
tab = TabFile()
tab.generate_tab(table, out_file_name)
```

Создание TAB файла для открытого источника тайлового сервиса:

```
prj_mercator = 'Earth Projection 10, 104, "m", 0 Bounds (-20037508.34, -
↪20037508.34) (20037508.34, 20037508.34)'
osm_raster = provider_manager.tms.open('http://maps.axioma-gis.ru/osm/{LEVEL}/
↪{ROW}/{COL}.png', prj=prj_mercator)
tab = TabFile()
out_file_name = tab.suggest_tab_name(osm_raster)
tab.generate_tab(osm_raster, out_file_name)
```

suggest_tab_name(data_object)
Сервисная функция. Предлагает наименование TAB файла для объекта данных. Результат можно использовать в методе generate_tab() в качестве имени выходного файла.

16.1.6.13 Attribute - Атрибут схемы таблицы

class axipy.da.**Attribute**(name, typedef)
Атрибут схемы таблицы.

Используется для создания и инспектирования атрибутов и схем axipy.da.Schema. Для создания атрибутов используйте функции string(), decimal() и другие.

Параметры

- **name** (str) – Название.
- **typedef** (str) – Описание типа.

Список 56: Пример создания

```
string_attr = Attribute.string('attribute_name', 80)
```

Methods:

bool(name)	Создает атрибут логического типа.
date(name)	Создает атрибут типа дата.
datetime(name)	Создает атрибут типа дата и время.
decimal(name[, length, precision])	Создает атрибут десятичного типа.
double(name)	Создает атрибут вещественного типа.
float(name)	Создает атрибут вещественного типа.
integer(name)	Создает атрибут целого типа.
string(name[, length])	Создает атрибут строкового типа.
time(name)	Создает атрибут типа время.

Attributes:

length	Длина атрибута.
name	Имя атрибута.
precision	Точность.
type_string	Тип в виде строки без длины и точности.
typedef	Описание типа.

static bool(name)
Создает атрибут логического типа.

Параметры **name** (str) – Имя атрибута.

Тип результата Attribute

static date(name)
Создает атрибут типа дата.

Параметры **name** (str) – Имя атрибута.

Тип результата Attribute

static datetime(name)
Создает атрибут типа дата и время.

Параметры **name** (str) – Имя атрибута.

Тип результата Attribute

static decimal(name, length=10, precision=5)
Создает атрибут десятичного типа.

Параметры

- **name** (str) – Имя атрибута.
- **length** (int) – Длина атрибута. Количество символов, включая запятую.
- **precision** (int) – Число знаков после запятой.

Тип результата Attribute

static double(name)
Создает атрибут вещественного типа.

Параметры **name** (str) – Имя атрибута.

Тип результата Attribute

static float(name)
Создает атрибут вещественного типа.

То же, что и double()

Параметры **name** (str) – Имя атрибута.

Тип результата Attribute

static integer(name)
Создает атрибут целого типа.

Параметры **name** (str) – Имя атрибута.

Тип результата Attribute

property length
Длина атрибута.

Тип результата int

property name
Имя атрибута.

Тип результата str

property precision
Точность.

Тип результата int

static string(name, length=80)
Создает атрибут строкового типа.

Параметры

- **name** (str) – Имя атрибута.

- **length** (int) – Длина атрибута.

Тип результата Attribute

static time(name)
Создает атрибут типа время.

Параметры **name** (str) – Имя атрибута.

Тип результата Attribute

property type_string
Тип в виде строки без длины и точности.

Тип результата str

property typedef
Описание типа.

Строка вида `<тип>[:длина][.точность]`.

Тип результата str

16.1.6.14 TypeSqlDialect - Диалект при выполнении запросов

class axipy.da.**TypeSqlDialect**(value)
Используемый диалект при выполнении SQL запросов. Есть отличия как в скорости, так и в функционале. выбор зависит от конкретной задачи. Значение, установленное по умолчанию можно получить посредством свойства `data_manager.sql_dialect`

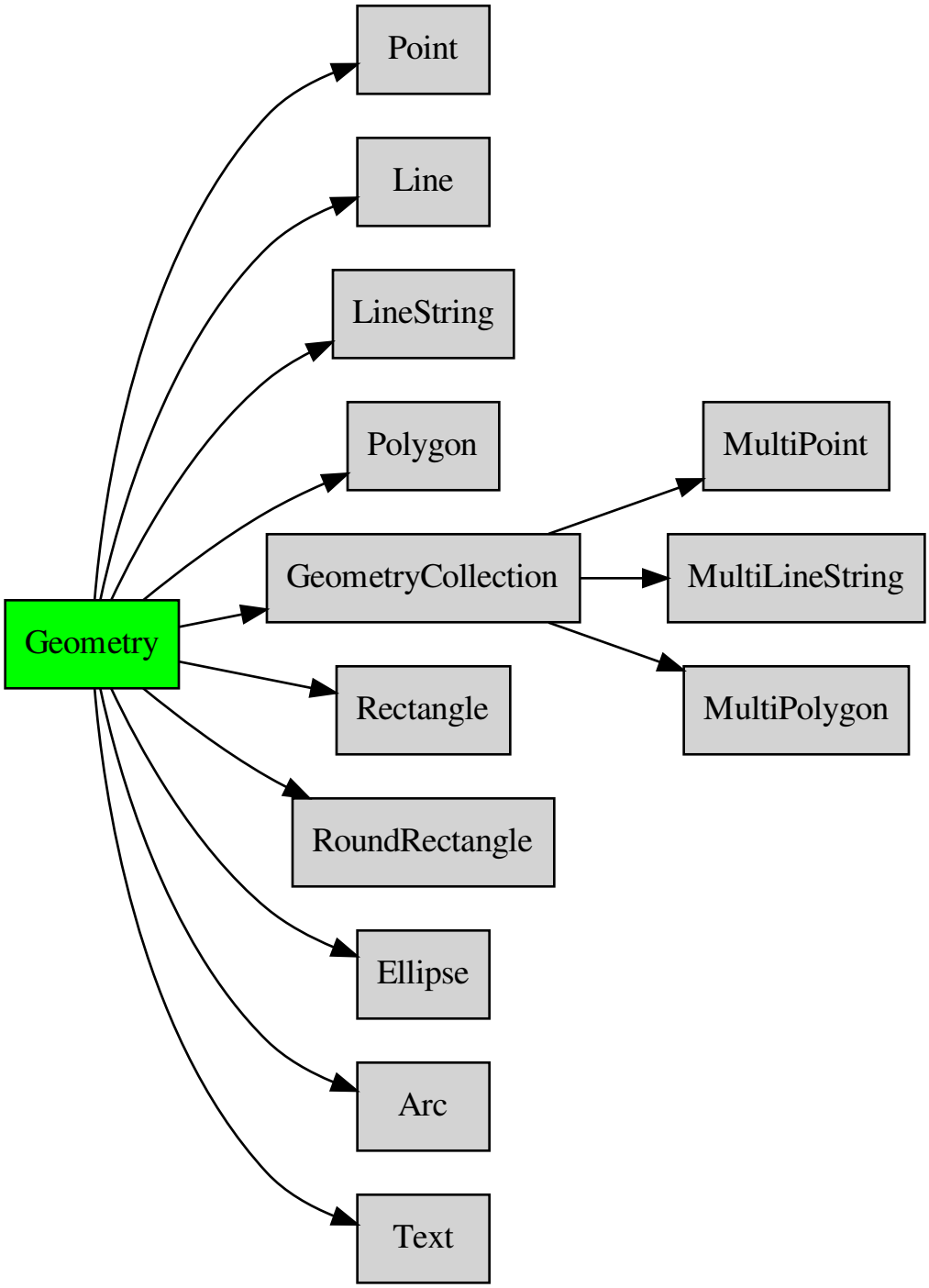
Таблица 48: Значения

Значение	Наименование
axioma	Использовать собственную реализацию инструмента выполнения запросов
sqlite	Использовать при выполнении реализацию, являющийся надстройкой над sqlite

16.1.6.15 axipy.da geometry

Geometry - Геометрия

Иерархия геометрических классов:



class axipy.da.**Geometry**
Абстрактный класс геометрического объекта (геометрии).

Примечание: Для получения краткого текстового представления геометрии можно воспользоваться функцией `str`. Для более полного - `repr()`.

---

affine_transform(trans)
Трансформирует объект исходя из заданных параметров трансформации.

См.также:

Для простых операций типа сдвиг, масштабирование и поворот, рекомендуется использовать `shift()`, `scale()`, `rotate()` соответственно.

Параметры **trans** (`QTransform`) – Матрица трансформации.

Тип результата `Geometry`

almost_equals(other, tolerance)
Производит примерное сравнения с другой геометрией в пределах заданной точности.

Параметры

- **other** (`Geometry`) – Сравниваемый объект.
- **tolerance** (`float`) – Точность сравнения.

Тип результата `bool`

Результат Возвращает True, если геометрии равны в пределах заданного отклонения.

boundary()
Возвращает границы геометрии в виде полилинии.

Тип результата `Geometry`

property bounds
Возвращает минимальный ограничивающий прямоугольник.

Тип результата `Rect`

buffer(distance, resolution=16, capStyle=1, joinStyle=1, mitreLimit=5.0)
Производит построение буфера.

Параметры

- **distance** (`float`) – Ширина буфера.
- **resolution** (`int`) – Количество сегментов на квадрант.
- **capStyle** (`int`) – Стиль окончания.
- **joinStyle** (`int`) – Стиль соединения.
- **mitreLimit** (`float`) – Предел среза.

Тип результата `Geometry`

centroid()
Возвращает центроид геометрии.

Тип результата `Geometry`

clone()
Создает копию объекта.

Тип результата Geometry

contains(other)
Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.

Тип результата bool

convex_hull()
Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.

Тип результата Geometry

property coordsystem
Система Координат (СК) геометрии.

Тип результата CoordSystem

covers(other)
Возвращает True, если геометрия охватывает геометрию other.

Тип результата bool

crosses(other)
Возвращает True, если при пересечении геометрий объекты частично пересекаются.

Тип результата bool

difference(other)
Возвращает область первой геометрии, которая не пересечена второй геометрией.

Тип результата Geometry

disjoint(other)
Возвращает True, если геометрии не пересекаются и не соприкасаются.

Тип результата bool

static distance_by_points(start, end, cs=None)
Производит расчет расстояния между двумя точками и азимут от первой до второй точки.

См.также:

Обратная функция point_by_azimuth()

Параметры

- **start** (Union[Pnt, Tuple[float, float]]) – Начальная точка. Если задана СК, то координаты в ней.
- **end** (Union[Pnt, Tuple[float, float]]) – Конечная точка. Если задана СК, то координаты в ней.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Tuple

Результат Возвращается пара значений (расстояние в метрах, азимут в градусах)

envelope()
Возвращает полигон, описывающий заданную геометрию.

Тип результата Geometry

equals(other)
Производит сравнение с другой геометрией.

Параметры **other** (Geometry) – Сравниваемая геометрия.

Тип результата bool

Результат Возвращает True, если геометрии равны.

static from_geojson(json, cs=None)
Возвращает геометрию из ее „GeoJSON“ представления.

Параметры

- **json** (str) – json представление в виде строки.
- **cs** (Optional[CoordSystem]) – Система координат.

Тип результата Geometry

static from_wkb(wkb, coordsystem=None)
Создает геометрический объект из строки формата WKB.

Параметры

- **wkb** (bytes) – Строка WKB
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии.

Список 57: Пример.

```
wkb = b'\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00$@\x00\x00\x00\x00\x00\x00
↪$@'
pnt = Geometry.from_wkb(wkb)
```

Тип результата Geometry

static from_wkt(wkt, coordsystem=None)
Создает геометрический объект из строки формата WKT.

Параметры

- **wkt** (str) – Строка WKT. Допустимо задание строки в формате с указанием SRID (EWKT). В данном случае система координат для создаваемой геометрии будет установлено исходя их этого значения.
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии. Если строка задана в виде EWKT и указано значение SRID, игнорируется.

Список 58: Пример.

```
polygon = Geometry.from_wkt('POLYGON ((10 10, 100 100, 100 10, 10 10))')
point = Geometry.from_wkt('SRID=4326;POINT (10 10)')
crs = CoordSystem.from_epsg(4326)
pline = Geometry.from_wkt('LINESTRING (30 10, 10 30, 40 40)', crs)
```

Тип результата Geometry

get_area(u=None)

Рассчитывает площадь, если объект площадной. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в квадратных метрах.

Список 59: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
poly = Polygon([(0,0), (0,2), (2, 2), (2,0)], cs=csMercatorKm)
print('Area Mercator km:', poly.get_area())
print('Area Mercator m:', poly.get_area(Unit.sq_m))
print('Perimeter Mercator km:', poly.get_perimeter())
print('Perimeter Mercator m:', poly.get_perimeter(Unit.m))
# В Широте/Долготе
poly.coordsystem = CoordSystem.from_prj("1, 104")
print('Area LL m:', poly.get_area())
print('Area LL km:', poly.get_area(Unit.sq_km))
print('Perimeter LL m:', poly.get_perimeter())
print('Perimeter LL km:', poly.get_perimeter(Unit.km))
'''
>>> Area Mercator km: 4.017946986632519
>>> Area Mercator m: 4017946.9866325194
>>> Perimeter Mercator km: 8.017972048421552
>>> Perimeter Mercator m: 8017.972048421552
>>> Area LL m: 49452172514.0342
>>> Area LL km: 49452.1725140342
>>> Perimeter LL m: 889423.5067063896
>>> Perimeter LL km: 889.4235067063896
'''
```

Параметры **u** (Optional[AreaUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_distance(other, u=None)

Производит расчет расстояния до объекта other. Результат возвращает в СК текущего объекта.

Параметры

- **other** (Geometry) – Анализируемый объект.
- **u** (Optional[LinearUnit]) – Единицы измерения, в которых требуется получить результат.

Список 60: Пример.

```
# Создадим геометрию без СК
first = Point(0, 0)
second = Point(10, 0)
print('В плане:', first.get_distance(second))
#Установим разные СК для точек
first.coordsystem = CoordSystem.from_prj("10, 104, 7, 0")
second.coordsystem = CoordSystem.from_prj("1, 104")
print('На сфере м:', first.get_distance(second))
print('На сфере км:', first.get_distance(second, Unit.km))
'''
>>> В плане: 10.0
>>> На сфере м: 1111948.7428468117
>>> На сфере км: 1111.9487428468117
'''
```

Тип результата float

get_length(u=None)

Рассчитывает длину геометрии. Метод применим только для линейных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Список 61: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
ls = Line((0,0), (10,0), csMercatorKm)
print('Length Mercator km:', ls.get_length())
print('Lenght Mercator m:', ls.get_length(Unit.m))
# В Широте/Долготе
csLL = CoordSystem.from_prj("1, 104")
ls = Line((0,0), (10,0), csLL)
print('Length LL m:', ls.get_length())
print('Length LL km:', ls.get_length(Unit.km))
'''
>>> Length Mercator km: 9.988805508567783
>>> Lenght Mercator m: 9988.805508567782
>>> Length LL m: 1111948.7428468117
>>> Length LL km: 1111.9487428468117
'''
```

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_perimeter(u=None)

Рассчитывает периметр геометрии. Метод применим только для площадных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Пример см. get_area()

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

intersection(other)
Возвращает область пересечения с другой геометрией.

Тип результата Geometry

intersects(other)
Возвращает True, если геометрии пересекаются.

Тип результата bool

property is_valid
Проверяет геометрию на валидность.

Тип результата bool

property is_valid_reason
Если геометрия неправильная, возвращает краткую аннотацию причины.

Тип результата str

property name
Возвращает наименование геометрического объекта.

Тип результата str

overlaps(other)
Возвращает True, если пересечение геометрий отличается от обеих геометрий.

Тип результата bool

static point_by_azimuth(point, azimuth, distance, cs=None)
Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.

См.также:

Обратная функция distance_by_points()

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, относительно которой производится расчет. Если задана СК, то в ней.
- **azimuth** (float) – Азимут в градусах, указывающий направление.
- **distance** (float) – Расстояние по азимуту. Задается в метрах.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Pnt

Результат Возвращает результирующую точку.

relate(other)
Проверяет отношения между объектами. Подробнее см. DE-9IM

Тип результата str

reproject(cs)
Перепроецирует геометрию в другую систему координат.

Параметры **cs** (CoordSystem) – СК, в которой требуется получить объект.

Тип результата Geometry

rotate(point, angle)
Поворот геометрии относительно заданной точки. Поворот производится против часовой стрелки.

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, вокруг которой необходимо произвести поворот.
- **angle** (float) – Угол поворота в градусах.

Тип результата Geometry

scale(kx, ky)
Масштабирует объект по заданным коэффициентам масштабирования. После выполнения операции центр результирующего объекта остается прежним.

Параметры

- **kx** (float) – Масштабирование по координате X.
- **ky** (float) – Масштабирование по координате Y.

Тип результата Geometry

shift(dx, dy)
Смещает объект на заданную величину. Значения задаются в текущей проекции.

Параметры

- **dx** (float) – Сдвиг по координате X.
- **dy** (float) – Сдвиг по координате Y.

Тип результата Geometry

symmetric_difference(other)
Возвращает логический XOR областей геометрий (объединение разниц).

Параметры **other** (Geometry) – Геометрия для анализа.

Тип результата Geometry

to_geojson()
Преобразует геометрию в формат „GeoJSON“

Тип результата str

to_linestring()
Пробует геометрию преобразовать в линейный объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

to_polygon()
Пробует геометрию преобразовать в площадной объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

touches(other)
Возвращает True, если геометрии соприкасаются.

Тип результата bool

property type
Возвращает тип геометрического элемента.

Таблица 49: Возможные значения

Значение	Наименование
Unknown	Не определен
Point	Точка
Line	Линия
LineString	Полилиния
Polygon	Полигон
MultiPoint	Коллекция точек
MultiLineString	Коллекция полилиний
MultiPolygon	Коллекция полигонов
GeometryCollection	Смешанная коллекция
Arc	Дуга
Ellipse	Эллипс
Rectangle	Прямоугольник
RoundedRectangle	Скругленный прямоугольник
Text	Текст

Список 62: Пример.

```
point = Point(10,10)
if point.type == GeometryType.Point:
    print('Это точка')
```

Тип результата GeometryType

union(other)
Возвращает результат объединения двух геометрий.

Тип результата Geometry

within(other)
Возвращает True, если геометрия находится полностью внутри геометрии other.

Тип результата bool

property wkb
Возвращает WKB строку для геометрии.

Тип результата bytes

property wkt
Возвращает WKT строку для геометрии.

Тип результата str

Point - Точечный объект

class axipy.da.**Point**(x, y, cs=None)

Базовые классы: axipy.da.Geometry

Геометрический объект типа точка.

Параметры

- **x** (float) – X координата
- **y** (float) – Y координата
- **cs** (Optional[CoordSystem]) – Система Координат, в которой создается геометрия.

Список 63: Пример.

```
cs = CoordSystem.from_prj("1, 104")
p = Point(23, 45, cs) # Создадим точку.
p.x = 55 # Изменим значение координаты X
```

Methods:

affine_transform(trans)	Трансформирует объект исходя из заданных параметров трансформации.
almost_equals(other, tolerance)	Производит примерное сравнения с другой геометрией в пределах заданной точности.
boundary()	Возвращает границы геометрии в виде полилинии.
buffer(distance[, resolution, capStyle, ...])	Производит построение буфера.
centroid()	Возвращает центроид геометрии.
clone()	Создает копию объекта.
contains(other)	Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.
convex_hull()	Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.
covers(other)	Возвращает True, если геометрия охватывает геометрию other.
crosses(other)	Возвращает True, если при пересечении геометрий объекты частично пересекаются.
difference(other)	Возвращает область первой геометрии, которая не пересечена второй геометрией.
disjoint(other)	Возвращает True, если геометрии не пересекаются и не соприкасаются.
distance_by_points(start, end[, cs])	Производит расчет расстояния между двумя точками и азимут от первой до второй точки.

continues on next page

Таблица 50 – продолжение с предыдущей страницы

envelope()	Возвращает полигон, описывающий заданную геометрию.
equals(other)	Производит сравнение с другой геометрией.
from_geojson(json[, cs])	Возвращает геометрию из ее „GeoJSON“ представления.
from_wkb(wkb[, coordsystem])	Создает геометрический объект из строки формата WKB.
from_wkt(wkt[, coordsystem])	Создает геометрический объект из строки формата WKT.
get_area([u])	Рассчитывает площадь, если объект площадной.
get_distance(other[, u])	Производит расчет расстояния до объекта other.
get_length([u])	Рассчитывает длину геометрии.
get_perimeter([u])	Рассчитывает периметр геометрии.
intersection(other)	Возвращает область пересечения с другой геометрией.
intersects(other)	Возвращает True, если геометрии пересекаются.
overlaps(other)	Возвращает True, если пересечение геометрий отличается от обеих геометрий.
point_by_azimuth(point, azimuth, distance[, cs])	Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.
relate(other)	Проверяет отношения между объектами.
reproject(cs)	Перепроецирует геометрию в другую систему координат.
rotate(point, angle)	Поворот геометрии относительно заданной точки.
scale(kx, ky)	Масштабирует объект по заданным коэффициентам масштабирования.
shift(dx, dy)	Смещает объект на заданную величину.
symmetric_difference(other)	Возвращает логический XOR областей геометрий (объединение разниц).
to_geojson()	Преобразует геометрию в формат „GeoJSON“
to_linestring()	Пробует геометрию преобразовать в линейный объект.
to_polygon()	Пробует геометрию преобразовать в площадной объект.
touches(other)	Возвращает True, если геометрии соприкасаются.
union(other)	Возвращает результат объединения двух геометрий.

continues on next page

Таблица 50 – продолжение с предыдущей страницы

within(other)	Возвращает True, если геометрия находится полностью внутри геометрии other.

Attributes:

bounds	Возвращает минимальный ограничивающий прямоугольник.
coordsystem	Система Координат (СК) геометрии.
is_valid	Проверяет геометрию на валидность.
is_valid_reason	Если геометрия неправильная, возвращает краткую аннотацию причины.
name	Возвращает наименование геометрического объекта.
type	Возвращает тип геометрического элемента.
wkb	Возвращает WKB строку для геометрии.
wkt	Возвращает WKT строку для геометрии.
x	X Координата.
y	Y Координата.

affine_transform(trans)
Трансформирует объект исходя из заданных параметров трансформации.

См.также:

Для простых операций типа сдвиг, масштабирование и поворот, рекомендуется использовать shift(), scale(), rotate() соответственно.

Параметры **trans** (QTransform) – Матрица трансформации.

Тип результата Geometry

almost_equals(other, tolerance)
Производит примерное сравнения с другой геометрией в пределах заданной точности.

Параметры

- **other** (Geometry) – Сравниваемый объект.
- **tolerance** (float) – Точность сравнения.

Тип результата bool

Результат Возвращает True, если геометрии равны в пределах заданного отклонения.

boundary()
Возвращает границы геометрии в виде полилинии.

Тип результата Geometry

property bounds
Возвращает минимальный ограничивающий прямоугольник.

Тип результата Rect

buffer(distance, resolution=16, capStyle=1, joinStyle=1, mitreLimit=5.0)
Производит построение буфера.

Параметры

- **distance** (float) – Ширина буфера.
- **resolution** (int) – Количество сегментов на квадрант.
- **capStyle** (int) – Стиль окончания.
- **joinStyle** (int) – Стиль соединения.
- **mitreLimit** (float) – Предел среза.

Тип результата Geometry

centroid()
Возвращает центроид геометрии.

Тип результата Geometry

clone()
Создает копию объекта.

Тип результата Geometry

contains(other)
Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.

Тип результата bool

convex_hull()
Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.

Тип результата Geometry

property coordsystem
Система Координат (СК) геометрии.

Тип результата CoordSystem

covers(other)
Возвращает True, если геометрия охватывает геометрию other.

Тип результата bool

crosses(other)
Возвращает True, если при пересечении геометрий объекты частично пересекаются.

Тип результата bool

difference(other)
Возвращает область первой геометрии, которая не пересечена второй геометрией.

Тип результата Geometry

disjoint(other)
Возвращает True, если геометрии не пересекаются и не соприкасаются.

Тип результата bool

static distance_by_points(start, end, cs=None)

Производит расчет расстояния между двумя точками и азимут от первой до второй точки.

См.также:

Обратная функция point_by_azimuth()

Параметры

- **start** (Union[Pnt, Tuple[float, float]]) – Начальная точка. Если задана СК, то координаты в ней.
- **end** (Union[Pnt, Tuple[float, float]]) – Конечная точка. Если задана СК, то координаты в ней.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Tuple

Результат Возвращается пара значений (расстояние в метрах, азимут в градусах)

envelope()

Возвращает полигон, описывающий заданную геометрию.

Тип результата Geometry

equals(other)

Производит сравнение с другой геометрией.

Параметры **other** (Geometry) – Сравниваемая геометрия.

Тип результата bool

Результат Возвращает True, если геометрии равны.

static from_geojson(json, cs=None)

Возвращает геометрию из ее „GeoJSON“ представления.

Параметры

- **json** (str) – json представление в виде строки.
- **cs** (Optional[CoordSystem]) – Система координат.

Тип результата Geometry

static from_wkb(wkb, coordsystem=None)

Создает геометрический объект из строки формата WKB.

Параметры

- **wkb** (bytes) – Строка WKB
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии.

Список 64: Пример.

```
wkb = b'\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00$@\x00\x00\x00\x00\x00\x00
↪$@'
pnt = Geometry.from_wkb(wkb)
```

Тип результата Geometry

static from_wkt(wkt, coordsystem=None)

Создает геометрический объект из строки формата WKT.

Параметры

- **wkt** (str) – Строка WKT. Допустимо задание строки в формате с указанием SRID (EWKT). В данном случае система координат для создаваемой геометрии будет установлено исходя их этого значения.
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии. Если строка задана в виде EWKT и указано значение SRID, игнорируется.

Список 65: Пример.

```
polygon = Geometry.from_wkt('POLYGON ((10 10, 100 100, 100 10, 10 10))')
point = Geometry.from_wkt('SRID=4326;POINT (10 10)')
crs = CoordSystem.from_epsg(4326)
pline = Geometry.from_wkt('LINESTRING (30 10, 10 30, 40 40)', crs)
```

Тип результата Geometry

get_area(u=None)

Рассчитывает площадь, если объект площадной. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в квадратных метрах.

Список 66: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
poly = Polygon([(0,0), (0,2), (2, 2), (2,0)], cs=csMercatorKm)
print('Area Mercator km:', poly.get_area())
print('Area Mercator m:', poly.get_area(Unit.sq_m))
print('Perimeter Mercator km:', poly.get_perimeter())
print('Perimeter Mercator m:', poly.get_perimeter(Unit.m))
# В Широте/Долготе
poly.coordsystem = CoordSystem.from_prj("1, 104")
print('Area LL m:', poly.get_area())
print('Area LL km:', poly.get_area(Unit.sq_km))
print('Perimeter LL m:', poly.get_perimeter())
print('Perimeter LL km:', poly.get_perimeter(Unit.km))
'''
>>> Area Mercator km: 4.017946986632519
>>> Area Mercator m: 4017946.9866325194
>>> Perimeter Mercator km: 8.017972048421552
>>> Perimeter Mercator m: 8017.972048421552
```

(continues on next page)

(продолжение с предыдущей страницы)

```
>>> Area LL m: 49452172514.0342
>>> Area LL km: 49452.1725140342
>>> Perimeter LL m: 889423.5067063896
>>> Perimeter LL km: 889.4235067063896
'''
```

Параметры **u** (Optional[AreaUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_distance(other, u=None)

Производит расчет расстояния до объекта other. Результат возвращает в СК текущего объекта.

Параметры

- **other** (Geometry) – Анализируемый объект.
- **u** (Optional[LinearUnit]) – Единицы измерения, в которых требуется получить результат.

Список 67: Пример.

```
# Создадим геометрию без СК
first = Point(0, 0)
second = Point(10, 0)
print('В плане:', first.get_distance(second))
#Установим разные СК для точек
first.coordsystem = CoordSystem.from_prj("10, 104, 7, 0")
second.coordsystem = CoordSystem.from_prj("1, 104")
print('На сфере м:', first.get_distance(second))
print('На сфере км:', first.get_distance(second, Unit.km))
'''
>>> В плане: 10.0
>>> На сфере м: 1111948.7428468117
>>> На сфере км: 1111.9487428468117
'''
```

Тип результата float

get_length(u=None)

Рассчитывает длину геометрии. Метод применим только для линейных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Список 68: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
ls = Line((0,0), (10,0), csMercatorKm)
print('Length Mercator km:', ls.get_length())
print('Lenght Mercator m:', ls.get_length(Unit.m))
# В Широте/Долготе
```

(continues on next page)

(продолжение с предыдущей страницы)

```
csLL = CoordSystem.from_prj("1, 104")
ls = Line((0,0), (10,0), csLL)
print('Length LL m:', ls.get_length())
print('Length LL km:', ls.get_length(Unit.km))
'''
>>> Length Mercator km: 9.988805508567783
>>> Lenght Mercator m: 9988.805508567782
>>> Length LL m: 1111948.7428468117
>>> Length LL km: 1111.9487428468117
'''
```

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_perimeter(u=None)

Рассчитывает периметр геометрии. Метод применим только для площадных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Пример см. get_area()

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

intersection(other)

Возвращает область пересечения с другой геометрией.

Тип результата Geometry

intersects(other)

Возвращает True, если геометрии пересекаются.

Тип результата bool

property is_valid

Проверяет геометрию на валидность.

Тип результата bool

property is_valid_reason

Если геометрия неправильная, возвращает краткую аннотацию причины.

Тип результата str

property name

Возвращает наименование геометрического объекта.

Тип результата str

overlaps(other)

Возвращает True, если пересечение геометрий отличается от обеих геометрий.

Тип результата bool

static point_by_azimuth(point, azimuth, distance, cs=None)
Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.

См.также:

Обратная функция distance_by_points()

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, относительно которой производится расчет. Если задана СК, то в ней.
- **azimuth** (float) – Азимут в градусах, указывающий направление.
- **distance** (float) – Расстояние по азимуту. Задается в метрах.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Pnt

Результат Возвращает результирующую точку.

relate(other)
Проверяет отношения между объектами. Подробнее см. DE-9IM

Тип результата str

reproject(cs)
Перепроецирует геометрию в другую систему координат.

Параметры **cs** (CoordSystem) – СК, в которой требуется получить объект.

Тип результата Geometry

rotate(point, angle)
Поворот геометрии относительно заданной точки. Поворот производится против часовой стрелки.

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, вокруг которой необходимо произвести поворот.
- **angle** (float) – Угол поворота в градусах.

Тип результата Geometry

scale(kx, ky)
Масштабирует объект по заданным коэффициентам масштабирования. После выполнения операции центр результирующего объекта остается прежним.

Параметры

- **kx** (float) – Масштабирование по координате X.
- **ky** (float) – Масштабирование по координате Y.

Тип результата Geometry

shift(dx, dy)
Смещает объект на заданную величину. Значения задаются в текущей проекции.

Параметры

- **dx** (float) – Сдвиг по координате X.
- **dy** (float) – Сдвиг по координате Y.

Тип результата Geometry

symmetric_difference(other)
Возвращает логический XOR областей геометрий (объединение разниц).

Параметры **other** (Geometry) – Геометрия для анализа.

Тип результата Geometry

to_geojson()
Преобразует геометрию в формат „GeoJSON“

Тип результата str

to_linestring()
Пробует геометрию преобразовать в линейный объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

to_polygon()
Пробует геометрию преобразовать в площадной объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

touches(other)
Возвращает True, если геометрии соприкасаются.

Тип результата bool

property type
Возвращает тип геометрического элемента.

Таблица 52: Возможные значения

Значение	Наименование
Unknown	Не определен
Point	Точка
Line	Линия
LineString	Полилиния
Polygon	Полигон
MultiPoint	Коллекция точек
MultiLineString	Коллекция полилиний
MultiPolygon	Коллекция полигонов
GeometryCollection	Смешанная коллекция
Arc	Дуга
Ellipse	Эллипс
Rectangle	Прямоугольник
RoundedRectangle	Скругленный прямоугольник
Text	Текст

Список 69: Пример.

```
point = Point(10,10)
if point.type == GeometryType.Point:
    print('Это точка')
```

Тип результата GeometryType

union(other)
Возвращает результат объединения двух геометрий.

Тип результата Geometry

within(other)
Возвращает True, если геометрия находится полностью внутри геометрии other.

Тип результата bool

property wkb
Возвращает WKB строку для геометрии.

Тип результата bytes

property wkt
Возвращает WKT строку для геометрии.

Тип результата str

property x
X Координата.

Тип результата float

property y
Y Координата.

Тип результата float

Line - Линия

class axipy.da.**Line**(begin, end, cs=None)

Базовые классы: axipy.da.Geometry

Геометрический объект типа линия.

Параметры

- **begin** (Pnt) – Начальная точка линии.
- **end** (Pnt) – Конечная точка линии.
- **cs** (Optional[CoordSystem]) – Система Координат, в которой создается геометрия.

Список 70: Пример.

```
cs = CoordSystem.from_prj("1, 104")
line = Line(Pnt(22, 44), (100, 101), cs) # Создадим линию с различным подходом␣
↪при указании начальной о конечной точки
line.begin = (88, 99) # Заменим координаты начальной точки.
line.end = Pnt(120, 120)
```

Methods:

affine_transform(trans)	Трансформирует объект исходя из заданных параметров трансформации.
almost_equals(other, tolerance)	Производит примерное сравнения с другой геометрией в пределах заданной точности.
boundary()	Возвращает границы геометрии в виде полилинии.
buffer(distance[, resolution, capStyle, ...])	Производит построение буфера.
centroid()	Возвращает центроид геометрии.
clone()	Создает копию объекта.
contains(other)	Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.
convex_hull()	Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.
covers(other)	Возвращает True, если геометрия охватывает геометрию other.
crosses(other)	Возвращает True, если при пересечении геометрий объекты частично пересекаются.
difference(other)	Возвращает область первой геометрии, которая не пересечена второй геометрией.
disjoint(other)	Возвращает True, если геометрии не пересекаются и не соприкасаются.
distance_by_points(start, end[, cs])	Производит расчет расстояния между двумя точками и азимут от первой до второй точки.
envelope()	Возвращает полигон, описывающий заданную геометрию.
equals(other)	Производит сравнение с другой геометрией.
from_geojson(json[, cs])	Возвращает геометрию из ее „GeoJSON“ представления.
from_wkb(wkb[, coordsystem])	Создает геометрический объект из строки формата WKB.
from_wkt(wkt[, coordsystem])	Создает геометрический объект из строки формата WKT.
get_area([u])	Рассчитывает площадь, если объект площадной.

continues on next page

Таблица 53 – продолжение с предыдущей страницы

get_distance(other[, u])	Производит расчет расстояния до объекта other.
get_length([u])	Рассчитывает длину геометрии.
get_perimeter([u])	Рассчитывает периметр геометрии.
intersection(other)	Возвращает область пересечения с другой геометрией.
intersects(other)	Возвращает True, если геометрии пересекаются.
overlaps(other)	Возвращает True, если пересечение геометрий отличается от обеих геометрий.
point_by_azimuth(point, azimuth, distance[, cs])	Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.
relate(other)	Проверяет отношения между объектами.
reproject(cs)	Перепроецирует геометрию в другую систему координат.
rotate(point, angle)	Поворот геометрии относительно заданной точки.
scale(kx, ky)	Масштабирует объект по заданным коэффициентам масштабирования.
shift(dx, dy)	Смещает объект на заданную величину.
symmetric_difference(other)	Возвращает логический XOR областей геометрий (объединение разниц).
to_geojson()	Преобразует геометрию в формат „GeoJSON“
to_linestring()	Пробует геометрию преобразовать в линейный объект.
to_polygon()	Пробует геометрию преобразовать в площадной объект.
touches(other)	Возвращает True, если геометрии соприкасаются.
union(other)	Возвращает результат объединения двух геометрий.
within(other)	Возвращает True, если геометрия находится полностью внутри геометрии other.

Attributes:

begin	Начальная точка линии.
bounds	Возвращает минимальный ограничивающий прямоугольник.
coordsystem	Система Координат (СК) геометрии.
end	Конечная точка линии.
is_valid	Проверяет геометрию на валидность.

continues on next page

Таблица 54 – продолжение с предыдущей страницы

is_valid_reason	Если геометрия неправильная, возвращает краткую аннотацию причины.
name	Возвращает наименование геометрического объекта.
type	Возвращает тип геометрического элемента.
wkb	Возвращает WKB строку для геометрии.
wkt	Возвращает WKT строку для геометрии.

affine_transform(trans)
Трансформирует объект исходя из заданных параметров трансформации.

См.также:

Для простых операций типа сдвиг, масштабирование и поворот, рекомендуется использовать shift(), scale(), rotate() соответственно.

Параметры **trans** (QTransform) – Матрица трансформации.

Тип результата Geometry

almost_equals(other, tolerance)
Производит примерное сравнения с другой геометрией в пределах заданной точности.

Параметры

- **other** (Geometry) – Сравниваемый объект.
- **tolerance** (float) – Точность сравнения.

Тип результата bool

Результат Возвращает True, если геометрии равны в пределах заданного отклонения.

property begin
Начальная точка линии. Точки допустимо создавать как экземпляр Pnt либо в виде пары float значений tuple

Тип результата Pnt

boundary()
Возвращает границы геометрии в виде полилинии.

Тип результата Geometry

property bounds
Возвращает минимальный ограничивающий прямоугольник.

Тип результата Rect

buffer(distance, resolution=16, capStyle=1, joinStyle=1, mitreLimit=5.0)
Производит построение буфера.

Параметры

- **distance** (float) – Ширина буфера.

- **resolution** (int) – Количество сегментов на квадрант.
- **capStyle** (int) – Стиль окончания.
- **joinStyle** (int) – Стиль соединения.
- **mitreLimit** (float) – Предел среза.

Тип результата Geometry

centroid()
Возвращает центроид геометрии.

Тип результата Geometry

clone()
Создает копию объекта.

Тип результата Geometry

contains(other)
Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.

Тип результата bool

convex_hull()
Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.

Тип результата Geometry

property coordsystem
Система Координат (СК) геометрии.

Тип результата CoordSystem

covers(other)
Возвращает True, если геометрия охватывает геометрию other.

Тип результата bool

crosses(other)
Возвращает True, если при пересечении геометрий объекты частично пересекаются.

Тип результата bool

difference(other)
Возвращает область первой геометрии, которая не пересечена второй геометрией.

Тип результата Geometry

disjoint(other)
Возвращает True, если геометрии не пересекаются и не соприкасаются.

Тип результата bool

static distance_by_points(start, end, cs=None)
Производит расчет расстояния между двумя точками и азимут от первой до второй точки.

См.также:

Обратная функция point_by_azimuth()

Параметры

- **start** (Union[Pnt, Tuple[float, float]]) – Начальная точка. Если задана СК, то координаты в ней.
- **end** (Union[Pnt, Tuple[float, float]]) – Конечная точка. Если задана СК, то координаты в ней.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Tuple

Результат Возвращается пара значений (расстояние в метрах, азимут в градусах)

property end
Конечная точка линии.

Тип результата Pnt

envelope()
Возвращает полигон, описывающий заданную геометрию.

Тип результата Geometry

equals(other)
Производит сравнение с другой геометрией.

Параметры **other** (Geometry) – Сравниваемая геометрия.

Тип результата bool

Результат Возвращает True, если геометрии равны.

static from_geojson(json, cs=None)
Возвращает геометрию из ее „GeoJSON“ представления.

Параметры

- **json** (str) – json представление в виде строки.
- **cs** (Optional[CoordSystem]) – Система координат.

Тип результата Geometry

static from_wkb(wkb, coordsystem=None)
Создает геометрический объект из строки формата WKB.

Параметры

- **wkb** (bytes) – Строка WKB
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии.

Список 71: Пример.

```
wkb = b'\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00$@\x00\x00\x00\x00\x00\x00
↪$@'
pnt = Geometry.from_wkb(wkb)
```

Тип результата Geometry

static from_wkt(wkt, coordsystem=None)
Создает геометрический объект из строки формата WKT.

Параметры

- **wkt** (str) – Строка WKT. Допустимо задание строки в формате с указанием SRID (EWKT). В данном случае система координат для создаваемой геометрии будет установлено исходя их этого значения.
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии. Если строка задана в виде EWKT и указано значение SRID, игнорируется.

Список 72: Пример.

```
polygon = Geometry.from_wkt('POLYGON ((10 10, 100 100, 100 10, 10 10))')
point = Geometry.from_wkt('SRID=4326;POINT (10 10)')
crs = CoordSystem.from_epsg(4326)
pline = Geometry.from_wkt('LINESTRING (30 10, 10 30, 40 40)', crs)
```

Тип результата Geometry

get_area(u=None)
Рассчитывает площадь, если объект площадной. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в квадратных метрах.

Список 73: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
poly = Polygon([(0,0), (0,2), (2, 2), (2,0)], cs=csMercatorKm)
print('Area Mercator km:', poly.get_area())
print('Area Mercator m:', poly.get_area(Unit.sq_m))
print('Perimeter Mercator km:', poly.get_perimeter())
print('Perimeter Mercator m:', poly.get_perimeter(Unit.m))
# В Широте/Долготе
poly.coordsystem = CoordSystem.from_prj("1, 104")
print('Area LL m:', poly.get_area())
print('Area LL km:', poly.get_area(Unit.sq_km))
print('Perimeter LL m:', poly.get_perimeter())
print('Perimeter LL km:', poly.get_perimeter(Unit.km))
'''
>>> Area Mercator km: 4.017946986632519
>>> Area Mercator m: 4017946.9866325194
>>> Perimeter Mercator km: 8.017972048421552
>>> Perimeter Mercator m: 8017.972048421552
>>> Area LL m: 49452172514.0342
>>> Area LL km: 49452.1725140342
>>> Perimeter LL m: 889423.5067063896
>>> Perimeter LL km: 889.4235067063896
'''
```

Параметры u (Optional[AreaUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится

расчет на плоскости.

Тип результата float

get_distance(other, u=None)

Производит расчет расстояния до объекта other. Результат возвращает в СК текущего объекта.

Параметры

- **other** (Geometry) – Анализируемый объект.
- **u** (Optional[LinearUnit]) – Единицы измерения, в которых требуется получить результат.

Список 74: Пример.

```
# Создадим геометрию без СК
first = Point(0, 0)
second = Point(10, 0)
print('В плане:', first.get_distance(second))
#Установим разные СК для точек
first.coordsystem = CoordSystem.from_prj("10, 104, 7, 0")
second.coordsystem = CoordSystem.from_prj("1, 104")
print('На сфере м:', first.get_distance(second))
print('На сфере км:', first.get_distance(second, Unit.km))
'''
>>> В плане: 10.0
>>> На сфере м: 1111948.7428468117
>>> На сфере км: 1111.9487428468117
'''
```

Тип результата float

get_length(u=None)

Рассчитывает длину геометрии. Метод применим только для линейных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Список 75: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
ls = Line((0,0), (10,0), csMercatorKm)
print('Length Mercator km:', ls.get_length())
print('Lenght Mercator m:', ls.get_length(Unit.m))
# В Широте/Долготе
csLL = CoordSystem.from_prj("1, 104")
ls = Line((0,0), (10,0), csLL)
print('Length LL m:', ls.get_length())
print('Length LL km:', ls.get_length(Unit.km))
'''
>>> Length Mercator km: 9.988805508567783
>>> Lenght Mercator m: 9988.805508567782
>>> Length LL m: 1111948.7428468117
>>> Length LL km: 1111.9487428468117
'''
```

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_perimeter(u=None)
Рассчитывает периметр геометрии. Метод применим только для площадных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Пример см. get_area()

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

intersection(other)
Возвращает область пересечения с другой геометрией.

Тип результата Geometry

intersects(other)
Возвращает True, если геометрии пересекаются.

Тип результата bool

property is_valid
Проверяет геометрию на валидность.

Тип результата bool

property is_valid_reason
Если геометрия неправильная, возвращает краткую аннотацию причины.

Тип результата str

property name
Возвращает наименование геометрического объекта.

Тип результата str

overlaps(other)
Возвращает True, если пересечение геометрий отличается от обеих геометрий.

Тип результата bool

static point_by_azimuth(point, azimuth, distance, cs=None)
Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.

См.также:

Обратная функция distance_by_points()

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, относительно которой производится расчет. Если задана СК, то в ней.

- **azimuth** (float) – Азимут в градусах, указывающий направление.
- **distance** (float) – Расстояние по азимуту. Задается в метрах.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Pnt

Результат Возвращает результирующую точку.

relate(other)
Проверяет отношения между объектами. Подробнее см. DE-9IM

Тип результата str

reproject(cs)
Перепроецирует геометрию в другую систему координат.

Параметры **cs** (CoordSystem) – СК, в которой требуется получить объект.

Тип результата Geometry

rotate(point, angle)
Поворот геометрии относительно заданной точки. Поворот производится против часовой стрелки.

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, вокруг которой необходимо произвести поворот.
- **angle** (float) – Угол поворота в градусах.

Тип результата Geometry

scale(kx, ky)
Масштабирует объект по заданным коэффициентам масштабирования. После выполнения операции центр результирующего объекта остается прежним.

Параметры

- **kx** (float) – Масштабирование по координате X.
- **ky** (float) – Масштабирование по координате Y.

Тип результата Geometry

shift(dx, dy)
Смещает объект на заданную величину. Значения задаются в текущей проекции.

Параметры

- **dx** (float) – Сдвиг по координате X.
- **dy** (float) – Сдвиг по координате Y.

Тип результата Geometry

symmetric_difference(other)
Возвращает логический XOR областей геометрий (объединение разниц).

Параметры **other** (Geometry) – Геометрия для анализа.

Тип результата Geometry

to_geojson()
Преобразует геометрию в формат „GeoJSON“

Тип результата str

to_linestring()
Пробует геометрию преобразовать в линейный объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

to_polygon()
Пробует геометрию преобразовать в площадной объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

touches(other)
Возвращает True, если геометрии соприкасаются.

Тип результата bool

property type
Возвращает тип геометрического элемента.

Таблица 55: Возможные значения

Значение	Наименование
Unknown	Не определен
Point	Точка
Line	Линия
LineString	Полилиния
Polygon	Полигон
MultiPoint	Коллекция точек
MultiLineString	Коллекция полилиний
MultiPolygon	Коллекция полигонов
GeometryCollection	Смешанная коллекция
Arc	Дуга
Ellipse	Эллипс
Rectangle	Прямоугольник
RoundedRectangle	Скругленный прямоугольник
Text	Текст

Список 76: Пример.

```
point = Point(10,10)
if point.type == GeometryType.Point:
    print('Это точка')
```

Тип результата GeometryType

union(other)
Возвращает результат объединения двух геометрий.

Тип результата Geometry

within(other)
Возвращает True, если геометрия находится полностью внутри геометрии other.

Тип результата bool

property wkb
Возвращает WKB строку для геометрии.

Тип результата bytes

property wkt
Возвращает WKT строку для геометрии.

Тип результата str

LineString - Полилиния

class axipy.da.**LineString**(*points, cs=None)

Базовые классы: axipy.da.Geometry

Геометрический объект типа полилиния.

Параметры

- **points** (Union[Pnt, Tuple[float, float]]) – Список точек. Может задаваться следующим образом:
  - В виде списка list из пар tuple.
  - В виде перечня точек. В данном случае, если необходимо задать СК, то требуется явно указать наименование параметра.
  - В виде итератора по элементам, состоящих из пар tuple.
- **cs** (Optional[CoordSystem]) – Система Координат, в которой создается геометрия.

Список 77: Пример.

```
csLL = CoordSystem.from_prj("1, 104")
ls = LineString([(1, 2), Pnt(3, 4), Pnt(5, 6), (7, 8)]) # Создадим полилинию без␣
↪СК
ls.points[1] = (33, 44) # Обновим точку с индексом 1. Допустимо только обновление␣
↪точки целиком. Изменение координат по одиночке не поддерживается.
ls.points.append((9,10)) # Добавим точку в конец
ls.points.remove(2) # Удалим вторую точку
ls.points.insert(3, (11,12)) # Добавим точку на позицию 3
for p in ls.points: # Просмотр всех точек
    print("point:", p)
ls2 = LineString((1, 2), (3, 4), (5, 6), (7, 8), cs=csLL) # Создание полинии,␣
↪передав перечень точек.
itr = (a for a in ls.points) # Создадим итератор на базе точек первой полилинии
ls3 = LineString(itr) #
```

Methods:

affine_transform(trans)	Трансформирует объект исходя из заданных параметров трансформации.

continues on next page

Таблица 56 – продолжение с предыдущей страницы

almost_equals(other, tolerance)	Производит примерное сравнения с другой геометрией в пределах заданной точности.
boundary()	Возвращает границы геометрии в виде полилинии.
buffer(distance[, resolution, capStyle, ...])	Производит построение буфера.
centroid()	Возвращает центроид геометрии.
clone()	Создает копию объекта.
contains(other)	Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.
convex_hull()	Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.
covers(other)	Возвращает True, если геометрия охватывает геометрию other.
crosses(other)	Возвращает True, если при пересечении геометрий объекты частично пересекаются.
difference(other)	Возвращает область первой геометрии, которая не пересечена второй геометрией.
disjoint(other)	Возвращает True, если геометрии не пересекаются и не соприкасаются.
distance_by_points(start, end[, cs])	Производит расчет расстояния между двумя точками и азимут от первой до второй точки.
envelope()	Возвращает полигон, описывающий заданную геометрию.
equals(other)	Производит сравнение с другой геометрией.
from_geojson(json[, cs])	Возвращает геометрию из ее „GeoJSON“ представления.
from_wkb(wkb[, coordsystem])	Создает геометрический объект из строки формата WKB.
from_wkt(wkt[, coordsystem])	Создает геометрический объект из строки формата WKT.
get_area([u])	Рассчитывает площадь, если объект площадной.
get_distance(other[, u])	Производит расчет расстояния до объекта other.
get_length([u])	Рассчитывает длину геометрии.
get_perimeter([u])	Рассчитывает периметр геометрии.
intersection(other)	Возвращает область пересечения с другой геометрией.
intersects(other)	Возвращает True, если геометрии пересекаются.
overlaps(other)	Возвращает True, если пересечение геометрий отличается от обеих геометрий.

continues on next page

Таблица 56 – продолжение с предыдущей страницы

point_by_azimuth(point, azimuth, distance[, cs])	Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.
relate(other)	Проверяет отношения между объектами.
reproject(cs)	Перепроецирует геометрию в другую систему координат.
rotate(point, angle)	Поворот геометрии относительно заданной точки.
scale(kx, ky)	Масштабирует объект по заданным коэффициентам масштабирования.
shift(dx, dy)	Смещает объект на заданную величину.
symmetric_difference(other)	Возвращает логический XOR областей геометрий (объединение разниц).
to_geojson()	Преобразует геометрию в формат „GeoJSON“
to_linestring()	Пробует геометрию преобразовать в линейный объект.
to_polygon()	Пробует геометрию преобразовать в площадной объект.
touches(other)	Возвращает True, если геометрии соприкасаются.
union(other)	Возвращает результат объединения двух геометрий.
within(other)	Возвращает True, если геометрия находится полностью внутри геометрии other.

Attributes:

bounds	Возвращает минимальный ограничивающий прямоугольник.
coordsystem	Система Координат (СК) геометрии.
is_valid	Проверяет геометрию на валидность.
is_valid_reason	Если геометрия неправильная, возвращает краткую аннотацию причины.
name	Возвращает наименование геометрического объекта.
points	Точки полилинии.
type	Возвращает тип геометрического элемента.
wkb	Возвращает WKB строку для геометрии.
wkt	Возвращает WKT строку для геометрии.

affine_transform(trans)

Трансформирует объект исходя из заданных параметров трансформации.

См.также:

Для простых операций типа сдвиг, масштабирование и поворот, рекомендуется использовать `shift()`, `scale()`, `rotate()` соответственно.

Параметры **trans** (`QTransform`) – Матрица трансформации.

Тип результата `Geometry`

almost_equals(other, tolerance)
Производит примерное сравнения с другой геометрией в пределах заданной точности.

Параметры

- **other** (`Geometry`) – Сравниваемый объект.
- **tolerance** (`float`) – Точность сравнения.

Тип результата `bool`

Результат Возвращает True, если геометрии равны в пределах заданного отклонения.

boundary()
Возвращает границы геометрии в виде полилинии.

Тип результата `Geometry`

property bounds
Возвращает минимальный ограничивающий прямоугольник.

Тип результата `Rect`

buffer(distance, resolution=16, capStyle=1, joinStyle=1, mitreLimit=5.0)
Производит построение буфера.

Параметры

- **distance** (`float`) – Ширина буфера.
- **resolution** (`int`) – Количество сегментов на квадрант.
- **capStyle** (`int`) – Стиль окончания.
- **joinStyle** (`int`) – Стиль соединения.
- **mitreLimit** (`float`) – Предел среза.

Тип результата `Geometry`

centroid()
Возвращает центроид геометрии.

Тип результата `Geometry`

clone()
Создает копию объекта.

Тип результата `Geometry`

contains(other)
Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.

Тип результата `bool`

convex_hull()
Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.

Тип результата Geometry

property coordsystem
Система Координат (СК) геометрии.

Тип результата CoordSystem

covers(other)
Возвращает True, если геометрия охватывает геометрию other.

Тип результата bool

crosses(other)
Возвращает True, если при пересечении геометрий объекты частично пересекаются.

Тип результата bool

difference(other)
Возвращает область первой геометрии, которая не пересечена второй геометрией.

Тип результата Geometry

disjoint(other)
Возвращает True, если геометрии не пересекаются и не соприкасаются.

Тип результата bool

static distance_by_points(start, end, cs=None)
Производит расчет расстояния между двумя точками и азимут от первой до второй точки.

См.также:

Обратная функция point_by_azimuth()

Параметры

- **start** (Union[Pnt, Tuple[float, float]]) – Начальная точка. Если задана СК, то координаты в ней.
- **end** (Union[Pnt, Tuple[float, float]]) – Конечная точка. Если задана СК, то координаты в ней.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Tuple

Результат Возвращается пара значений (расстояние в метрах, азимут в градусах)

envelope()
Возвращает полигон, описывающий заданную геометрию.

Тип результата Geometry

equals(other)
Производит сравнение с другой геометрией.

Параметры **other** (Geometry) – Сравниваемая геометрия.

Тип результата bool

Результат Возвращает True, если геометрии равны.

static from_geojson(json, cs=None)
Возвращает геометрию из ее „GeoJSON“ представления.

Параметры

- **json** (str) – json представление в виде строки.
- **cs** (Optional[CoordSystem]) – Система координат.

Тип результата Geometry

static from_wkb(wkb, coordsystem=None)
Создает геометрический объект из строки формата WKB.

Параметры

- **wkb** (bytes) – Строка WKB
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии.

Список 78: Пример.

```
wkb = b'\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00$@\x00\x00\x00\x00\x00\x00
↪$@'
pnt = Geometry.from_wkb(wkb)
```

Тип результата Geometry

static from_wkt(wkt, coordsystem=None)
Создает геометрический объект из строки формата WKT.

Параметры

- **wkt** (str) – Строка WKT. Допустимо задание строки в формате с указанием SRID (EWKT). В данном случае система координат для создаваемой геометрии будет установлено исходя их этого значения.
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии. Если строка задана в виде EWKT и указано значение SRID, игнорируется.

Список 79: Пример.

```
polygon = Geometry.from_wkt('POLYGON ((10 10, 100 100, 100 10, 10 10))')
point = Geometry.from_wkt('SRID=4326;POINT (10 10)')
crs = CoordSystem.from_epsg(4326)
pline = Geometry.from_wkt('LINESTRING (30 10, 10 30, 40 40)', crs)
```

Тип результата Geometry

get_area(u=None)
Рассчитывает площадь, если объект площадной. В противном случае

возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в квадратных метрах.

Список 80: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
poly = Polygon([(0,0), (0,2), (2, 2), (2,0)], cs=csMercatorKm)
print('Area Mercator km:', poly.get_area())
print('Area Mercator m:', poly.get_area(Unit.sq_m))
print('Perimeter Mercator km:', poly.get_perimeter())
print('Perimeter Mercator m:', poly.get_perimeter(Unit.m))
# В Широте/Долготе
poly.coordsystem = CoordSystem.from_prj("1, 104")
print('Area LL m:', poly.get_area())
print('Area LL km:', poly.get_area(Unit.sq_km))
print('Perimeter LL m:', poly.get_perimeter())
print('Perimeter LL km:', poly.get_perimeter(Unit.km))
'''
>>> Area Mercator km: 4.017946986632519
>>> Area Mercator m: 4017946.9866325194
>>> Perimeter Mercator km: 8.017972048421552
>>> Perimeter Mercator m: 8017.972048421552
>>> Area LL m: 49452172514.0342
>>> Area LL km: 49452.1725140342
>>> Perimeter LL m: 889423.5067063896
>>> Perimeter LL km: 889.4235067063896
'''
```

Параметры **u** (Optional[AreaUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_distance(other, u=None)

Производит расчет расстояния до объекта other. Результат возвращает в СК текущего объекта.

Параметры

- **other** (Geometry) – Анализируемый объект.
- **u** (Optional[LinearUnit]) – Единицы измерения, в которых требуется получить результат.

Список 81: Пример.

```
# Создадим геометрию без СК
first = Point(0, 0)
second = Point(10, 0)
print('В плане:', first.get_distance(second))
#Установим разные СК для точек
first.coordsystem = CoordSystem.from_prj("10, 104, 7, 0")
second.coordsystem = CoordSystem.from_prj("1, 104")
print('На сфере м:', first.get_distance(second))
print('На сфере км:', first.get_distance(second, Unit.km))
```

(continues on next page)

(продолжение с предыдущей страницы)

```
'''
>>> В плане: 10.0
>>> На сфере м: 1111948.7428468117
>>> На сфере км: 1111.9487428468117
'''
```

Тип результата float

get_length(u=None)

Рассчитывает длину геометрии. Метод применим только для линейных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Список 82: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
ls = Line((0,0), (10,0), csMercatorKm)
print('Length Mercator km:', ls.get_length())
print('Lenght Mercator m:', ls.get_length(Unit.m))
# В Широте/Долготе
csLL = CoordSystem.from_prj("1, 104")
ls = Line((0,0), (10,0), csLL)
print('Length LL m:', ls.get_length())
print('Length LL km:', ls.get_length(Unit.km))
'''
>>> Length Mercator km: 9.988805508567783
>>> Lenght Mercator m: 9988.805508567782
>>> Length LL m: 1111948.7428468117
>>> Length LL km: 1111.9487428468117
'''
```

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_perimeter(u=None)

Рассчитывает периметр геометрии. Метод применим только для площадных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Пример см. get_area()

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

intersection(other)

Возвращает область пересечения с другой геометрией.

Тип результата Geometry

intersects(other)
Возвращает True, если геометрии пересекаются.

Тип результата bool

property is_valid
Проверяет геометрию на валидность.

Тип результата bool

property is_valid_reason
Если геометрия неправильная, возвращает краткую аннотацию причины.

Тип результата str

property name
Возвращает наименование геометрического объекта.

Тип результата str

overlaps(other)
Возвращает True, если пересечение геометрий отличается от обеих геометрий.

Тип результата bool

static point_by_azimuth(point, azimuth, distance, cs=None)
Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.

См.также:

Обратная функция distance_by_points()

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, относительно которой производится расчет. Если задана СК, то в ней.
- **azimuth** (float) – Азимут в градусах, указывающий направление.
- **distance** (float) – Расстояние по азимуту. Задается в метрах.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Pnt

Результат Возвращает результирующую точку.

property points
Точки полилинии. Реализован как список python list точек Pnt. Также поддерживаются список пар tuple.

Примечание: При обновление значения точки допустимо только изменение ее заменой.

Тип результата List[Pnt]

relate(other)
Проверяет отношения между объектами. Подробнее см. DE-9IM

Тип результата str

reproject(cs)
Перепроецирует геометрию в другую систему координат.

Параметры **cs** (CoordSystem) – СК, в которой требуется получить объект.

Тип результата Geometry

rotate(point, angle)
Поворот геометрии относительно заданной точки. Поворот производится против часовой стрелки.

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, вокруг которой необходимо произвести поворот.
- **angle** (float) – Угол поворота в градусах.

Тип результата Geometry

scale(kx, ky)
Масштабирует объект по заданным коэффициентам масштабирования. После выполнения операции центр результирующего объекта остается прежним.

Параметры

- **kx** (float) – Масштабирование по координате X.
- **ky** (float) – Масштабирование по координате Y.

Тип результата Geometry

shift(dx, dy)
Смещает объект на заданную величину. Значения задаются в текущей проекции.

Параметры

- **dx** (float) – Сдвиг по координате X.
- **dy** (float) – Сдвиг по координате Y.

Тип результата Geometry

symmetric_difference(other)
Возвращает логический XOR областей геометрий (объединение разниц).

Параметры **other** (Geometry) – Геометрия для анализа.

Тип результата Geometry

to_geojson()
Преобразует геометрию в формат „GeoJSON“

Тип результата str

to_linestring()
Пробует геометрию преобразовать в линейный объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

to_polygon()
Пробует геометрию преобразовать в площадной объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

touches(other)
Возвращает True, если геометрии соприкасаются.

Тип результата bool

property type
Возвращает тип геометрического элемента.

Таблица 58: Возможные значения

Значение	Наименование
Unknown	Не определен
Point	Точка
Line	Линия
LineString	Полилиния
Polygon	Полигон
MultiPoint	Коллекция точек
MultiLineString	Коллекция полилиний
MultiPolygon	Коллекция полигонов
GeometryCollection	Смешанная коллекция
Arc	Дуга
Ellipse	Эллипс
Rectangle	Прямоугольник
RoundedRectangle	Скругленный прямоугольник
Text	Текст

Список 83: Пример.

```
point = Point(10,10)
if point.type == GeometryType.Point:
    print('Это точка')
```

Тип результата GeometryType

union(other)
Возвращает результат объединения двух геометрий.

Тип результата Geometry

within(other)
Возвращает True, если геометрия находится полностью внутри геометрии other.

Тип результата bool

property wkb
Возвращает WKB строку для геометрии.

Тип результата bytes

property wkt
Возвращает WKT строку для геометрии.

Тип результата str

Polygon - Полигон

class axipy.da.**Polygon**(*points, cs=None)

Базовые классы: axipy.da.Geometry

Геометрический объект типа полигон. Представляет собой часть плоскости, ограниченной замкнутой полилинией. Кроме внешней границы, полигон может иметь одну или несколько внутренних (дырок).

Параметры

- **points** (Union[Pnt, Tuple[float, float]]) – Список точек внешнего контура. Может задаваться следующим образом:
  - В виде списка list из пар tuple.
  - В виде перечня точек. В данном случае, если необходимо задать СК, то требуется явно указать наименование параметра.
  - В виде итератора по элементам, состоящих из пар tuple.
- **cs** (Optional[CoordSystem]) – Система Координат, в которой создается геометрия.

Список 84: Пример.

```
poly = Polygon([(0, 0), Pnt(1, 10), Pnt(10, 11), (10, 2)]) # Создадим объект
poly.points[2] = (14, 15) # Поменяем вторую точку
poly.points.insert(3, (11, 5)) # Добавим точку
for p in poly.points: # Просмотр точек полигона
    print("point:", p)
poly.points.remove(2) # Удалим вторую точку
```

Methods:

affine_transform(trans)	Трансформирует объект исходя из заданных параметров трансформации.
almost_equals(other, tolerance)	Производит примерное сравнения с другой геометрией в пределах заданной точности.
boundary()	Возвращает границы геометрии в виде полилинии.
buffer(distance[, resolution, capStyle, ...])	Производит построение буфера.
centroid()	Возвращает центроид геометрии.
clone()	Создает копию объекта.
contains(other)	Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.
convex_hull()	Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.
covers(other)	Возвращает True, если геометрия охватывает геометрию other.

continues on next page

Таблица 59 – продолжение с предыдущей страницы

crosses(other)	Возвращает True, если при пересечении геометрий объекты частично пересекаются.
difference(other)	Возвращает область первой геометрии, которая не пересечена второй геометрией.
disjoint(other)	Возвращает True, если геометрии не пересекаются и не соприкасаются.
distance_by_points(start, end[, cs])	Производит расчет расстояния между двумя точками и азимут от первой до второй точки.
envelope()	Возвращает полигон, описывающий заданную геометрию.
equals(other)	Производит сравнение с другой геометрией.
from_geojson(json[, cs])	Возвращает геометрию из ее „GeoJSON“ представления.
from_rect(rect[, cs])	Создает полигон на базе прямоугольника.
from_wkb(wkb[, coordsystem])	Создает геометрический объект из строки формата WKB.
from_wkt(wkt[, coordsystem])	Создает геометрический объект из строки формата WKT.
get_area([u])	Рассчитывает площадь, если объект площадной.
get_distance(other[, u])	Производит расчет расстояния до объекта other.
get_length([u])	Рассчитывает длину геометрии.
get_perimeter([u])	Рассчитывает периметр геометрии.
intersection(other)	Возвращает область пересечения с другой геометрией.
intersects(other)	Возвращает True, если геометрии пересекаются.
overlaps(other)	Возвращает True, если пересечение геометрий отличается от обеих геометрий.
point_by_azimuth(point, azimuth, distance[, cs])	Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.
relate(other)	Проверяет отношения между объектами.
reproject(cs)	Перепроецирует геометрию в другую систему координат.
rotate(point, angle)	Поворот геометрии относительно заданной точки.
scale(kx, ky)	Масштабирует объект по заданным коэффициентам масштабирования.
shift(dx, dy)	Смещает объект на заданную величину.

continues on next page

Таблица 59 – продолжение с предыдущей страницы

symmetric_difference(other)	Возвращает логический XOR областей геометрий (объединение разниц).
to_geojson()	Преобразует геометрию в формат „GeoJSON“
to_linestring()	Пробует геометрию преобразовать в линейный объект.
to_polygon()	Пробует геометрию преобразовать в площадной объект.
touches(other)	Возвращает True, если геометрии соприкасаются.
union(other)	Возвращает результат объединения двух геометрий.
within(other)	Возвращает True, если геометрия находится полностью внутри геометрии other.

Attributes:

bounds	Возвращает минимальный ограничивающий прямоугольник.
coordsystem	Система Координат (СК) геометрии.
holes	Дырки полигона.
is_valid	Проверяет геометрию на валидность.
is_valid_reason	Если геометрия неправильная, возвращает краткую аннотацию причины.
name	Возвращает наименование геометрического объекта.
points	Точки полигона.
type	Возвращает тип геометрического элемента.
wkb	Возвращает WKB строку для геометрии.
wkt	Возвращает WKT строку для геометрии.

affine_transform(trans)
Трансформирует объект исходя из заданных параметров трансформации.

См.также:

Для простых операций типа сдвиг, масштабирование и поворот, рекомендуется использовать shift(), scale(), rotate() соответственно.

Параметры **trans** (QTransform) – Матрица трансформации.

Тип результата Geometry

almost_equals(other, tolerance)
Производит примерное сравнения с другой геометрией в пределах заданной точности.

Параметры

- **other** (Geometry) – Сравниваемый объект.
- **tolerance** (float) – Точность сравнения.

Тип результата bool

Результат Возвращает True, если геометрии равны в пределах заданного отклонения.

boundary()
Возвращает границы геометрии в виде полилинии.

Тип результата Geometry

property bounds
Возвращает минимальный ограничивающий прямоугольник.

Тип результата Rect

buffer(distance, resolution=16, capStyle=1, joinStyle=1, mitreLimit=5.0)
Производит построение буфера.

Параметры

- **distance** (float) – Ширина буфера.
- **resolution** (int) – Количество сегментов на квадрант.
- **capStyle** (int) – Стиль окончания.
- **joinStyle** (int) – Стиль соединения.
- **mitreLimit** (float) – Предел среза.

Тип результата Geometry

centroid()
Возвращает центроид геометрии.

Тип результата Geometry

clone()
Создает копию объекта.

Тип результата Geometry

contains(other)
Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.

Тип результата bool

convex_hull()
Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.

Тип результата Geometry

property coordsystem
Система Координат (СК) геометрии.

Тип результата CoordSystem

covers(other)
Возвращает True, если геометрия охватывает геометрию other.

Тип результата bool

crosses(other)
Возвращает True, если при пересечении геометрий объекты частично пересекаются.

Тип результата bool

difference(other)
Возвращает область первой геометрии, которая не пересечена второй геометрией.

Тип результата Geometry

disjoint(other)
Возвращает True, если геометрии не пересекаются и не соприкасаются.

Тип результата bool

static distance_by_points(start, end, cs=None)
Производит расчет расстояния между двумя точками и азимут от первой до второй точки.

См.также:

Обратная функция point_by_azimuth()

Параметры

- **start** (Union[Pnt, Tuple[float, float]]) – Начальная точка. Если задана СК, то координаты в ней.
- **end** (Union[Pnt, Tuple[float, float]]) – Конечная точка. Если задана СК, то координаты в ней.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Tuple

Результат Возвращается пара значений (расстояние в метрах, азимут в градусах)

envelope()
Возвращает полигон, описывающий заданную геометрию.

Тип результата Geometry

equals(other)
Производит сравнение с другой геометрией.

Параметры **other** (Geometry) – Сравниваемая геометрия.

Тип результата bool

Результат Возвращает True, если геометрии равны.

static from_geojson(json, cs=None)
Возвращает геометрию из ее „GeoJSON“ представления.

Параметры

- **json** (str) – json представление в виде строки.
- **cs** (Optional[CoordSystem]) – Система координат.

Тип результата Geometry

static from_rect(rect, cs=None)
Создает полигон на базе прямоугольника.

Параметры

- **rect** (Rect) – Прямоугольник, на основе которого формируются координаты.
- **cs** (Optional[CoordSystem]) – Система Координат, в которой создается геометрия.

Тип результата Polygon

static from_wkb(wkb, coordsystem=None)
Создает геометрический объект из строки формата WKB.

Параметры

- **wkb** (bytes) – Строка WKB
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии.

Список 85: Пример.

```
wkb = b'\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00$@\x00\x00\x00\x00\x00\x00
↪$@'
pnt = Geometry.from_wkb(wkb)
```

Тип результата Geometry

static from_wkt(wkt, coordsystem=None)
Создает геометрический объект из строки формата WKT.

Параметры

- **wkt** (str) – Строка WKT. Допустимо задание строки в формате с указанием SRID (EWKT). В данном случае система координат для создаваемой геометрии будет установлено исходя их этого значения.
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии. Если строка задана в виде EWKT и указано значение SRID, игнорируется.

Список 86: Пример.

```
polygon = Geometry.from_wkt('POLYGON ((10 10, 100 100, 100 10, 10 10))')
point = Geometry.from_wkt('SRID=4326;POINT (10 10)')
crs = CoordSystem.from_epsg(4326)
pline = Geometry.from_wkt('LINESTRING (30 10, 10 30, 40 40)', crs)
```

Тип результата Geometry

get_area(u=None)
Рассчитывает площадь, если объект площадной. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в квадратных метрах.

Список 87: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
poly = Polygon([(0,0), (0,2), (2, 2), (2,0)], cs=csMercatorKm)
print('Area Mercator km:', poly.get_area())
print('Area Mercator m:', poly.get_area(Unit.sq_m))
print('Perimeter Mercator km:', poly.get_perimeter())
print('Perimeter Mercator m:', poly.get_perimeter(Unit.m))
# В Широте/Долготе
poly.coordsystem = CoordSystem.from_prj("1, 104")
print('Area LL m:', poly.get_area())
print('Area LL km:', poly.get_area(Unit.sq_km))
print('Perimeter LL m:', poly.get_perimeter())
print('Perimeter LL km:', poly.get_perimeter(Unit.km))
'''
>>> Area Mercator km: 4.017946986632519
>>> Area Mercator m: 4017946.9866325194
>>> Perimeter Mercator km: 8.017972048421552
>>> Perimeter Mercator m: 8017.972048421552
>>> Area LL m: 49452172514.0342
>>> Area LL km: 49452.1725140342
>>> Perimeter LL m: 889423.5067063896
>>> Perimeter LL km: 889.4235067063896
'''
```

Параметры **u** (Optional[AreaUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_distance(other, u=None)

Производит расчет расстояния до объекта other. Результат возвращает в СК текущего объекта.

Параметры

- **other** (Geometry) – Анализируемый объект.
- **u** (Optional[LinearUnit]) – Единицы измерения, в которых требуется получить результат.

Список 88: Пример.

```
# Создадим геометрию без СК
first = Point(0, 0)
second = Point(10, 0)
print('В плане:', first.get_distance(second))
#Установим разные СК для точек
first.coordsystem = CoordSystem.from_prj("10, 104, 7, 0")
second.coordsystem = CoordSystem.from_prj("1, 104")
print('На сфере м:', first.get_distance(second))
print('На сфере км:', first.get_distance(second, Unit.km))
'''
>>> В плане: 10.0
```

(continues on next page)

(продолжение с предыдущей страницы)

```
>>> На сфере м: 1111948.7428468117
>>> На сфере км: 1111.9487428468117
'''
```

Тип результата float

get_length(u=None)

Рассчитывает длину геометрии. Метод применим только для линейных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Список 89: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
ls = Line((0,0), (10,0), csMercatorKm)
print('Length Mercator km:', ls.get_length())
print('Lenght Mercator m:', ls.get_length(Unit.m))
# В Широте/Долготе
csLL = CoordSystem.from_prj("1, 104")
ls = Line((0,0), (10,0), csLL)
print('Length LL m:', ls.get_length())
print('Length LL km:', ls.get_length(Unit.km))
'''
>>> Length Mercator km: 9.988805508567783
>>> Lenght Mercator m: 9988.805508567782
>>> Length LL m: 1111948.7428468117
>>> Length LL km: 1111.9487428468117
'''
```

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_perimeter(u=None)

Рассчитывает периметр геометрии. Метод применим только для площадных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Пример см. get_area()

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

property holes

Дырки полигона. Реализован в виде списка list.

Список 90: Пример.

```
poly = Polygon((0, 0), (1, 10), (10, 1))
poly.holes.append([(2,2), (2,4), (5,3)]) # Добавим дырку
for p in poly.holes[0]: # Просмотр точек дырки полигона
    print("Point of hole:", p)
print('Вторая точка первой дырки:', poly.holes[0][1])
poly.holes[0][1] = (33,44) # Обновим значение этой точки
```

Тип результата List[Pnt]

intersection(other)
Возвращает область пересечения с другой геометрией.

Тип результата Geometry

intersects(other)
Возвращает True, если геометрии пересекаются.

Тип результата bool

property is_valid
Проверяет геометрию на валидность.

Тип результата bool

property is_valid_reason
Если геометрия неправильная, возвращает краткую аннотацию причины.

Тип результата str

property name
Возвращает наименование геометрического объекта.

Тип результата str

overlaps(other)
Возвращает True, если пересечение геометрий отличается от обеих геометрий.

Тип результата bool

static point_by_azimuth(point, azimuth, distance, cs=None)
Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.

См.также:

Обратная функция distance_by_points()

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, относительно которой производится расчет. Если задана СК, то в ней.
- **azimuth** (float) – Азимут в градусах, указывающий направление.
- **distance** (float) – Расстояние по азимуту. Задается в метрах.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Pnt

Результат Возвращает результирующую точку.

property points
Точки полигона. Реализован как список python `list` точек `Pnt`. Также поддерживаются список пар `tuple`.

Тип результата `List[Pnt]`

relate(other)
Проверяет отношения между объектами. Подробнее см. DE-9IM

Тип результата `str`

reproject(cs)
Перепроецирует геометрию в другую систему координат.

Параметры **cs** (`CoordSystem`) – СК, в которой требуется получить объект.

Тип результата `Geometry`

rotate(point, angle)
Поворот геометрии относительно заданной точки. Поворот производится против часовой стрелки.

Параметры

- **point** (`Union[Pnt, Tuple[float, float]]`) – Точка, вокруг которой необходимо произвести поворот.
- **angle** (`float`) – Угол поворота в градусах.

Тип результата `Geometry`

scale(kx, ky)
Масштабирует объект по заданным коэффициентам масштабирования. После выполнения операции центр результирующего объекта остается прежним.

Параметры

- **kx** (`float`) – Масштабирование по координате X.
- **ky** (`float`) – Масштабирование по координате Y.

Тип результата `Geometry`

shift(dx, dy)
Смещает объект на заданную величину. Значения задаются в текущей проекции.

Параметры

- **dx** (`float`) – Сдвиг по координате X.
- **dy** (`float`) – Сдвиг по координате Y.

Тип результата `Geometry`

symmetric_difference(other)
Возвращает логический XOR областей геометрий (объединение разниц).

Параметры **other** (`Geometry`) – Геометрия для анализа.

Тип результата `Geometry`

to_geojson()
Преобразует геометрию в формат „GeoJSON“

Тип результата str

to_linestring()
Пробует геометрию преобразовать в линейный объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

to_polygon()
Пробует геометрию преобразовать в площадной объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

touches(other)
Возвращает True, если геометрии соприкасаются.

Тип результата bool

property type
Возвращает тип геометрического элемента.

Таблица 61: Возможные значения

Значение	Наименование
Unknown	Не определен
Point	Точка
Line	Линия
LineString	Полилиния
Polygon	Полигон
MultiPoint	Коллекция точек
MultiLineString	Коллекция полилиний
MultiPolygon	Коллекция полигонов
GeometryCollection	Смешанная коллекция
Arc	Дуга
Ellipse	Эллипс
Rectangle	Прямоугольник
RoundedRectangle	Скругленный прямоугольник
Text	Текст

Список 91: Пример.

```
point = Point(10,10)
if point.type == GeometryType.Point:
    print('Это точка')
```

Тип результата GeometryType

union(other)
Возвращает результат объединения двух геометрий.

Тип результата Geometry

within(other)
Возвращает True, если геометрия находится полностью внутри геометрии other.

Тип результата bool

property wkb
Возвращает WKB строку для геометрии.

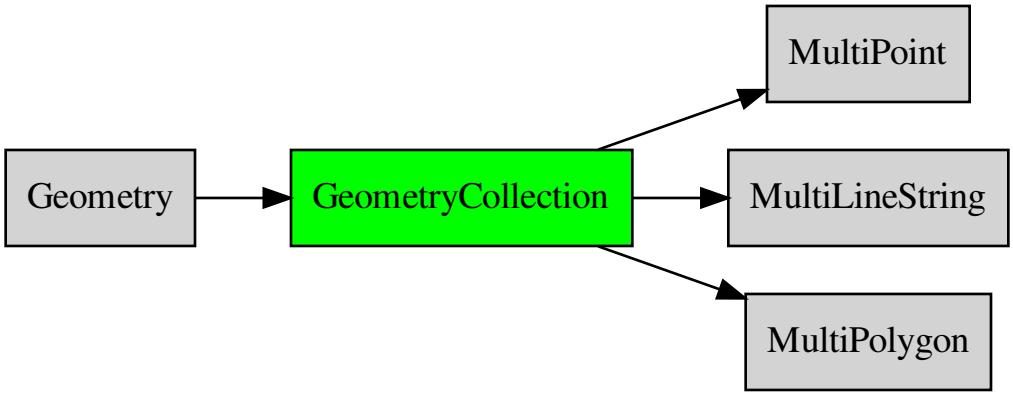
Тип результата bytes

property wkt
Возвращает WKT строку для геометрии.

Тип результата str

GeometryCollection - Коллекция геометрий

Иерархия геометрических классов:



class axipy.da.**GeometryCollection**(cs=None)

Базовые классы: axipy.da.Geometry

Коллекция разнотипных геометрических объектов. Допустимо хранение геометрических объектов различного типа, за исключением коллекций. Доступ к элементам производится по аналогии работы со списком list.

Параметры cs (Optional[CoordSystem]) – Система Координат, в которой создается геометрия.

Для получения размера коллекции используйте функцию len:

```
cnt = len(coll)
```

Доступ к элементам производится по индексу. Нумерация начинается с 0.

В качестве примера получим первый элемент:

```
coll[1]
```

Обновление геометрии в коллекции так же производится по ее индексу. Так же допустимо изменение некоторых свойств геометрии в зависимости от ее типа.

```
# Примеры установки элемента с индексом 1
coll[1] = (2,2) # Как точки с координатами (2,2)
coll[1] = [(101, 102), (103, 104), (105, 106)] # Как полилинии
coll[1] = Polygon((101, 102), (103, 104), (105, 106)) # Как полигона

# Доступ к элементам коллекции::
for g in coll:
    print('Геометрия:', g)
```

Methods:

affine_transform(trans)	Трансформирует объект исходя из заданных параметров трансформации.
almost_equals(other, tolerance)	Производит примерное сравнения с другой геометрией в пределах заданной точности.
append(*value)	Добавление геометрии в коллекцию.
boundary()	Возвращает границы геометрии в виде полилинии.
buffer(distance[, resolution, capStyle, …])	Производит построение буфера.
centroid()	Возвращает центроид геометрии.
clone()	Создает копию объекта.
contains(other)	Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.
convex_hull()	Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.
covers(other)	Возвращает True, если геометрия охватывает геометрию other.
crosses(other)	Возвращает True, если при пересечении геометрий объекты частично пересекаются.
difference(other)	Возвращает область первой геометрии, которая не пересечена второй геометрией.
disjoint(other)	Возвращает True, если геометрии не пересекаются и не соприкасаются.
distance_by_points(start, end[, cs])	Производит расчет расстояния между двумя точками и азимут от первой до второй точки.
envelope()	Возвращает полигон, описывающий заданную геометрию.
equals(other)	Производит сравнение с другой геометрией.
from_geojson(json[, cs])	Возвращает геометрию из ее „GeoJSON“ представления.

continues on next page

Таблица 62 – продолжение с предыдущей страницы

from_wkb(wkb[, coordsystem])	Создает геометрический объект из строки формата WKB.
from_wkt(wkt[, coordsystem])	Создает геометрический объект из строки формата WKT.
get_area([u])	Рассчитывает площадь, если объект площадной.
get_distance(other[, u])	Производит расчет расстояния до объекта other.
get_length([u])	Рассчитывает длину геометрии.
get_perimeter([u])	Рассчитывает периметр геометрии.
intersection(other)	Возвращает область пересечения с другой геометрией.
intersects(other)	Возвращает True, если геометрии пересекаются.
overlaps(other)	Возвращает True, если пересечение геометрий отличается от обеих геометрий.
point_by_azimuth(point, azimuth, distance[, cs])	Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.
relate(other)	Проверяет отношения между объектами.
remove(idx)	» Удаление геометрии из коллекции.
reproject(cs)	Перепроецирует геометрию в другую систему координат.
rotate(point, angle)	Поворот геометрии относительно заданной точки.
scale(kx, ky)	Масштабирует объект по заданным коэффициентам масштабирования.
shift(dx, dy)	Смещает объект на заданную величину.
symmetric_difference(other)	Возвращает логический XOR областей геометрий (объединение разниц).
to_geojson()	Преобразует геометрию в формат „GeoJSON“
to_linestring()	Пробует геометрию преобразовать в линейный объект.
to_polygon()	Пробует геометрию преобразовать в площадной объект.
touches(other)	Возвращает True, если геометрии соприкасаются.
try_to_simplified()	Создает копию коллекции при этом пробует упростить ее тип.
union(other)	Возвращает результат объединения двух геометрий.
within(other)	Возвращает True, если геометрия находится полностью внутри геометрии other.

Attributes:

bounds	Возвращает минимальный ограничивающий прямоугольник.
coordsystem	Система Координат (СК) геометрии.
is_valid	Проверяет геометрию на валидность.
is_valid_reason	Если геометрия неправильная, возвращает краткую аннотацию причины.
name	Возвращает наименование геометрического объекта.
type	Возвращает тип геометрического элемента.
wkb	Возвращает WKB строку для геометрии.
wkt	Возвращает WKT строку для геометрии.

affine_transform(trans)
Трансформирует объект исходя из заданных параметров трансформации.

См.также:

Для простых операций типа сдвиг, масштабирование и поворот, рекомендуется использовать shift(), scale(), rotate() соответственно.

Параметры **trans** (QTransform) – Матрица трансформации.

Тип результата Geometry

almost_equals(other, tolerance)
Производит примерное сравнения с другой геометрией в пределах заданной точности.

Параметры

- **other** (Geometry) – Сравниваемый объект.
- **tolerance** (float) – Точность сравнения.

Тип результата bool

Результат Возвращает True, если геометрии равны в пределах заданного отклонения.

append(*value)
Добавление геометрии в коллекцию. Задание параметров аналогично указанию их при создании объектов конкретного типа. Если опустить указание класса, то для одной точки или пары значений float будет создан точечный объект Point, если точек больше, - объект класса LineString.

Список 92: Пример.

```
# Создадим коллекцию и добавим в нее несколько объектов различного типа.
coll = GeometryCollection()
coll.append((1,2)) # Точка
coll.append(1,2) # Точка
coll.append([(3,4), (5, 5), (10, 0)]) # Полилиния в виде :class:`list`. Можно
↪это сделать через конструктор :class:`LinearString`.
```

(continues on next page)

(продолжение с предыдущей страницы)

```
coll.append((3,4), (5, 5), (10, 0)) # Полилиния
coll.append(Polygon([(3,4), (5, 5), (10, 0)])) # Полигон
```

boundary()
Возвращает границы геометрии в виде полилинии.

Тип результата Geometry

property bounds
Возвращает минимальный ограничивающий прямоугольник.

Тип результата Rect

buffer(distance, resolution=16, capStyle=1, joinStyle=1, mitreLimit=5.0)
Производит построение буфера.

Параметры

- **distance** (float) – Ширина буфера.
- **resolution** (int) – Количество сегментов на квадрант.
- **capStyle** (int) – Стиль окончания.
- **joinStyle** (int) – Стиль соединения.
- **mitreLimit** (float) – Предел среза.

Тип результата Geometry

centroid()
Возвращает центроид геометрии.

Тип результата Geometry

clone()
Создает копию объекта.

Тип результата Geometry

contains(other)
Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.

Тип результата bool

convex_hull()
Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.

Тип результата Geometry

property coordsystem
Система Координат (СК) геометрии.

Тип результата CoordSystem

covers(other)
Возвращает True, если геометрия охватывает геометрию other.

Тип результата bool

crosses(other)
Возвращает True, если при пересечении геометрий объекты частично пересекаются.

Тип результата bool

difference(other)
Возвращает область первой геометрии, которая не пересечена второй геометрией.

Тип результата Geometry

disjoint(other)
Возвращает True, если геометрии не пересекаются и не соприкасаются.

Тип результата bool

static distance_by_points(start, end, cs=None)
Производит расчет расстояния между двумя точками и азимут от первой до второй точки.

См.также:

Обратная функция point_by_azimuth()

Параметры

- **start** (Union[Pnt, Tuple[float, float]]) – Начальная точка. Если задана СК, то координаты в ней.
- **end** (Union[Pnt, Tuple[float, float]]) – Конечная точка. Если задана СК, то координаты в ней.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Tuple

Результат Возвращается пара значений (расстояние в метрах, азимут в градусах)

envelope()
Возвращает полигон, описывающий заданную геометрию.

Тип результата Geometry

equals(other)
Производит сравнение с другой геометрией.

Параметры **other** (Geometry) – Сравниваемая геометрия.

Тип результата bool

Результат Возвращает True, если геометрии равны.

static from_geojson(json, cs=None)
Возвращает геометрию из ее „GeoJSON“ представления.

Параметры

- **json** (str) – json представление в виде строки.
- **cs** (Optional[CoordSystem]) – Система координат.

Тип результата Geometry

static from_wkb(wkb, coordsystem=None)
Создает геометрический объект из строки формата WKB.

Параметры

- **wkb** (bytes) – Строка WKB
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии.

Список 93: Пример.

```
wkb = b'\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00$@\x00\x00\x00\x00\x00\x00
↪$@'
pnt = Geometry.from_wkb(wkb)
```

Тип результата Geometry

static from_wkt(wkt, coordsystem=None)

Создает геометрический объект из строки формата WKT.

Параметры

- **wkt** (str) – Строка WKT. Допустимо задание строки в формате с указанием SRID (EWKT). В данном случае система координат для создаваемой геометрии будет установлено исходя их этого значения.
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии. Если строка задана в виде EWKT и указано значение SRID, игнорируется.

Список 94: Пример.

```
polygon = Geometry.from_wkt('POLYGON ((10 10, 100 100, 100 10, 10 10))')
point = Geometry.from_wkt('SRID=4326;POINT (10 10)')
crs = CoordSystem.from_epsg(4326)
pline = Geometry.from_wkt('LINESTRING (30 10, 10 30, 40 40)', crs)
```

Тип результата Geometry

get_area(u=None)

Рассчитывает площадь, если объект площадной. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в квадратных метрах.

Список 95: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
poly = Polygon([(0,0), (0,2), (2, 2), (2,0)], cs=csMercatorKm)
print('Area Mercator km:', poly.get_area())
print('Area Mercator m:', poly.get_area(Unit.sq_m))
print('Perimeter Mercator km:', poly.get_perimeter())
print('Perimeter Mercator m:', poly.get_perimeter(Unit.m))
# В Широте/Долготе
poly.coordsystem = CoordSystem.from_prj("1, 104")
print('Area LL m:', poly.get_area())
print('Area LL km:', poly.get_area(Unit.sq_km))
print('Perimeter LL m:', poly.get_perimeter())
```

(continues on next page)

(продолжение с предыдущей страницы)

```
print('Perimeter LL km:', poly.get_perimeter(Unit.km))
'''
>>> Area Mercator km: 4.017946986632519
>>> Area Mercator m: 4017946.9866325194
>>> Perimeter Mercator km: 8.017972048421552
>>> Perimeter Mercator m: 8017.972048421552
>>> Area LL m: 49452172514.0342
>>> Area LL km: 49452.1725140342
>>> Perimeter LL m: 889423.5067063896
>>> Perimeter LL km: 889.4235067063896
'''
```

Параметры **u** (Optional[AreaUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_distance(other, u=None)

Производит расчет расстояния до объекта other. Результат возвращает в СК текущего объекта.

Параметры

- **other** (Geometry) – Анализируемый объект.
- **u** (Optional[LinearUnit]) – Единицы измерения, в которых требуется получить результат.

Список 96: Пример.

```
# Создадим геометрию без СК
first = Point(0, 0)
second = Point(10, 0)
print('В плане:', first.get_distance(second))
#Установим разные СК для точек
first.coordsystem = CoordSystem.from_prj("10, 104, 7, 0")
second.coordsystem = CoordSystem.from_prj("1, 104")
print('На сфере м:', first.get_distance(second))
print('На сфере км:', first.get_distance(second, Unit.km))
'''
>>> В плане: 10.0
>>> На сфере м: 1111948.7428468117
>>> На сфере км: 1111.9487428468117
'''
```

Тип результата float

get_length(u=None)

Рассчитывает длину геометрии. Метод применим только для линейных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Список 97: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
ls = Line((0,0), (10,0), csMercatorKm)
print('Length Mercator km:', ls.get_length())
print('Lenght Mercator m:', ls.get_length(Unit.m))
# В Широте/Долготе
csLL = CoordSystem.from_prj("1, 104")
ls = Line((0,0), (10,0), csLL)
print('Length LL m:', ls.get_length())
print('Length LL km:', ls.get_length(Unit.km))
'''
>>> Length Mercator km: 9.988805508567783
>>> Lenght Mercator m: 9988.805508567782
>>> Length LL m: 1111948.7428468117
>>> Length LL km: 1111.9487428468117
'''
```

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_perimeter(u=None)

Рассчитывает периметр геометрии. Метод применим только для площадных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Пример см. get_area()

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

intersection(other)

Возвращает область пересечения с другой геометрией.

Тип результата Geometry

intersects(other)

Возвращает True, если геометрии пересекаются.

Тип результата bool

property is_valid

Проверяет геометрию на валидность.

Тип результата bool

property is_valid_reason

Если геометрия неправильная, возвращает краткую аннотацию причины.

Тип результата str

property name
Возвращает наименование геометрического объекта.

Тип результата str

overlaps(other)
Возвращает True, если пересечение геометрий отличается от обеих геометрий.

Тип результата bool

static point_by_azimuth(point, azimuth, distance, cs=None)
Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.

См.также:

Обратная функция distance_by_points()

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, относительно которой производится расчет. Если задана СК, то в ней.
- **azimuth** (float) – Азимут в градусах, указывающий направление.
- **distance** (float) – Расстояние по азимуту. Задается в метрах.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Pnt

Результат Возвращает результирующую точку.

relate(other)
Проверяет отношения между объектами. Подробнее см. DE-9IM

Тип результата str

remove(idx)
» Удаление геометрии из коллекции.

Параметры **idx** (int) – Индекс геометрии в коллекции.

reproject(cs)
Перепроецирует геометрию в другую систему координат.

Параметры **cs** (CoordSystem) – СК, в которой требуется получить объект.

Тип результата Geometry

rotate(point, angle)
Поворот геометрии относительно заданной точки. Поворот производится против часовой стрелки.

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, вокруг которой необходимо произвести поворот.
- **angle** (float) – Угол поворота в градусах.

Тип результата Geometry

scale(kx, ky)
Масштабирует объект по заданным коэффициентам масштабирования. После выполнения операции центр результирующего объекта остается прежним.

Параметры

- **kx** (float) – Масштабирование по координате X.
- **ky** (float) – Масштабирование по координате Y.

Тип результата Geometry

shift(dx, dy)
Смещает объект на заданную величину. Значения задаются в текущей проекции.

Параметры

- **dx** (float) – Сдвиг по координате X.
- **dy** (float) – Сдвиг по координате Y.

Тип результата Geometry

symmetric_difference(other)
Возвращает логический XOR областей геометрий (объединение разниц).

Параметры **other** (Geometry) – Геометрия для анализа.

Тип результата Geometry

to_geojson()
Преобразует геометрию в формат „GeoJSON“

Тип результата str

to_linestring()
Пробует геометрию преобразовать в линейный объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

to_polygon()
Пробует геометрию преобразовать в площадной объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

touches(other)
Возвращает True, если геометрии соприкасаются.

Тип результата bool

try_to_simplified()
Создает копию коллекции при этом пробует упростить ее тип. К примеру, если в коллекции только полигоны, то вернет объект типа MultiPolygon. Если исходная коллекция разнородна, возвращается копия исходной.

Тип результата Union[MultiPoint, MultiLineString, MultiPolygon, GeometryCollection]

property type
Возвращает тип геометрического элемента.

Таблица 64: Возможные значения

Значение	Наименование
Unknown	Не определен
Point	Точка
Line	Линия
LineString	Полилиния
Polygon	Полигон
MultiPoint	Коллекция точек
MultiLineString	Коллекция полилиний
MultiPolygon	Коллекция полигонов
GeometryCollection	Смешанная коллекция
Arc	Дуга
Ellipse	Эллипс
Rectangle	Прямоугольник
RoundedRectangle	Скругленный прямоугольник
Text	Текст

Список 98: Пример.

```
point = Point(10,10)
if point.type == GeometryType.Point:
    print('Это точка')
```

Тип результата GeometryType

union(other)
Возвращает результат объединения двух геометрий.

Тип результата Geometry

within(other)
Возвращает True, если геометрия находится полностью внутри геометрии other.

Тип результата bool

property wkb
Возвращает WKB строку для геометрии.

Тип результата bytes

property wkt
Возвращает WKT строку для геометрии.

Тип результата str

MultiPoint - Коллекция точек

class axipy.da.**MultiPoint**(cs=None)

Базовые классы: axipy.da.GeometryCollection

Коллекция точечных объектов. Может содержать только объекты типа точка.

Параметры cs (Optional[CoordSystem]) – Система Координат, в которой создается геометрия.

Список 99: Пример.

```
mpoint = MultiPoint() # Создаем коллекцию.
# Добавим точку разными способами.
p = Point(23, 34)
mpoint.append(p)
mpoint.append((12, 12))
mpoint.append(Pnt(10,10))
mpoint[0] = (66,66) # Заменяем первый объект (индекс 0)
mpoint[0].x = 77 # Заменим только координату x для первого объекта в коллекции
mpoint.remove(1) # Удаляем второй объект
```

Methods:

affine_transform(trans)	Трансформирует объект исходя из заданных параметров трансформации.
almost_equals(other, tolerance)	Производит примерное сравнения с другой геометрией в пределах заданной точности.
append(*value)	Добавление геометрии в коллекцию.
boundary()	Возвращает границы геометрии в виде полилинии.
buffer(distance[, resolution, capStyle, ...])	Производит построение буфера.
centroid()	Возвращает центроид геометрии.
clone()	Создает копию объекта.
contains(other)	Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.
convex_hull()	Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.
covers(other)	Возвращает True, если геометрия охватывает геометрию other.
crosses(other)	Возвращает True, если при пересечении геометрий объекты частично пересекаются.
difference(other)	Возвращает область первой геометрии, которая не пересечена второй геометрией.
disjoint(other)	Возвращает True, если геометрии не пересекаются и не соприкасаются.

continues on next page

Таблица 65 – продолжение с предыдущей страницы

distance_by_points(start, end[, cs])	Производит расчет расстояния между двумя точками и азимут от первой до второй точки.
envelope()	Возвращает полигон, описывающий заданную геометрию.
equals(other)	Производит сравнение с другой геометрией.
from_geojson(json[, cs])	Возвращает геометрию из ее „GeoJSON“ представления.
from_wkb(wkb[, coordsystem])	Создает геометрический объект из строки формата WKB.
from_wkt(wkt[, coordsystem])	Создает геометрический объект из строки формата WKT.
get_area([u])	Рассчитывает площадь, если объект площадной.
get_distance(other[, u])	Производит расчет расстояния до объекта other.
get_length([u])	Рассчитывает длину геометрии.
get_perimeter([u])	Рассчитывает периметр геометрии.
intersection(other)	Возвращает область пересечения с другой геометрией.
intersects(other)	Возвращает True, если геометрии пересекаются.
overlaps(other)	Возвращает True, если пересечение геометрий отличается от обеих геометрий.
point_by_azimuth(point, azimuth, distance[, cs])	Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.
relate(other)	Проверяет отношения между объектами.
remove(idx)	» Удаление геометрии из коллекции.
reproject(cs)	Перепроецирует геометрию в другую систему координат.
rotate(point, angle)	Поворот геометрии относительно заданной точки.
scale(kx, ky)	Масштабирует объект по заданным коэффициентам масштабирования.
shift(dx, dy)	Смещает объект на заданную величину.
symmetric_difference(other)	Возвращает логический XOR областей геометрий (объединение разниц).
to_geojson()	Преобразует геометрию в формат „GeoJSON“
to_linestring()	Пробует геометрию преобразовать в линейный объект.
to_polygon()	Пробует геометрию преобразовать в площадной объект.
touches(other)	Возвращает True, если геометрии соприкасаются.

continues on next page

Таблица 65 – продолжение с предыдущей страницы

try_to_simplified()	Создает копию коллекции при этом пробует упростить ее тип.
union(other)	Возвращает результат объединения двух геометрий.
within(other)	Возвращает True, если геометрия находится полностью внутри геометрии other.

Attributes:

bounds	Возвращает минимальный ограничивающий прямоугольник.
coordsystem	Система Координат (СК) геометрии.
is_valid	Проверяет геометрию на валидность.
is_valid_reason	Если геометрия неправильная, возвращает краткую аннотацию причины.
name	Возвращает наименование геометрического объекта.
type	Возвращает тип геометрического элемента.
wkb	Возвращает WKB строку для геометрии.
wkt	Возвращает WKT строку для геометрии.

affine_transform(trans)
Трансформирует объект исходя из заданных параметров трансформации.

См.также:

Для простых операций типа сдвиг, масштабирование и поворот, рекомендуется использовать shift(), scale(), rotate() соответственно.

Параметры **trans** (QTransform) – Матрица трансформации.

Тип результата Geometry

almost_equals(other, tolerance)
Производит примерное сравнения с другой геометрией в пределах заданной точности.

Параметры

- **other** (Geometry) – Сравниваемый объект.
- **tolerance** (float) – Точность сравнения.

Тип результата bool

Результат Возвращает True, если геометрии равны в пределах заданного отклонения.

append(*value)
Добавление геометрии в коллекцию. Задание параметров аналогично указанию их при создании объектов конкретного типа. Если опустить указание класса, то

для одной точки или пары значений float будет создан точечный объект Point, если точек больше, - объект класса LineString.

Список 100: Пример.

```
# Создадим коллекцию и добавим в нее несколько объектов различного типа.
coll = GeometryCollection()
coll.append((1,2)) # Точка
coll.append(1,2) # Точка
coll.append([(3,4), (5, 5), (10, 0)]) # Полилиния в виде :class:`list`. Можно
↪это сделать через конструктор :class:`LinearString`.
coll.append((3,4), (5, 5), (10, 0)) # Полилиния
coll.append(Polygon([(3,4), (5, 5), (10, 0)])) # Полигон
```

boundary()
Возвращает границы геометрии в виде полилинии.

Тип результата Geometry

property bounds
Возвращает минимальный ограничивающий прямоугольник.

Тип результата Rect

buffer(distance, resolution=16, capStyle=1, joinStyle=1, mitreLimit=5.0)
Производит построение буфера.

Параметры

- **distance** (float) – Ширина буфера.
- **resolution** (int) – Количество сегментов на квадрант.
- **capStyle** (int) – Стиль окончания.
- **joinStyle** (int) – Стиль соединения.
- **mitreLimit** (float) – Предел среза.

Тип результата Geometry

centroid()
Возвращает центроид геометрии.

Тип результата Geometry

clone()
Создает копию объекта.

Тип результата Geometry

contains(other)
Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.

Тип результата bool

convex_hull()
Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.

Тип результата Geometry

property coordsystem
Система Координат (СК) геометрии.

Тип результата CoordSystem

covers(other)
Возвращает True, если геометрия охватывает геометрию other.

Тип результата bool

crosses(other)
Возвращает True, если при пересечении геометрий объекты частично пересекаются.

Тип результата bool

difference(other)
Возвращает область первой геометрии, которая не пересечена второй геометрией.

Тип результата Geometry

disjoint(other)
Возвращает True, если геометрии не пересекаются и не соприкасаются.

Тип результата bool

static distance_by_points(start, end, cs=None)
Производит расчет расстояния между двумя точками и азимут от первой до второй точки.

См.также:

Обратная функция point_by_azimuth()

Параметры

- **start** (Union[Pnt, Tuple[float, float]]) – Начальная точка. Если задана СК, то координаты в ней.
- **end** (Union[Pnt, Tuple[float, float]]) – Конечная точка. Если задана СК, то координаты в ней.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Tuple

Результат Возвращается пара значений (расстояние в метрах, азимут в градусах)

envelope()
Возвращает полигон, описывающий заданную геометрию.

Тип результата Geometry

equals(other)
Производит сравнение с другой геометрией.

Параметры **other** (Geometry) – Сравниваемая геометрия.

Тип результата bool

Результат Возвращает True, если геометрии равны.

static from_geojson(json, cs=None)
Возвращает геометрию из ее „GeoJSON“ представления.

Параметры

- **json** (str) – json представление в виде строки.
- **cs** (Optional[CoordSystem]) – Система координат.

Тип результата Geometry

static from_wkb(wkb, coordsystem=None)

Создает геометрический объект из строки формата WKB.

Параметры

- **wkb** (bytes) – Строка WKB
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии.

Список 101: Пример.

```
wkb = b'\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00$@\x00\x00\x00\x00\x00\x00
↪$@'
pnt = Geometry.from_wkb(wkb)
```

Тип результата Geometry

static from_wkt(wkt, coordsystem=None)

Создает геометрический объект из строки формата WKT.

Параметры

- **wkt** (str) – Строка WKT. Допустимо задание строки в формате с указанием SRID (EWKT). В данном случае система координат для создаваемой геометрии будет установлено исходя их этого значения.
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии. Если строка задана в виде EWKT и указано значение SRID, игнорируется.

Список 102: Пример.

```
polygon = Geometry.from_wkt('POLYGON ((10 10, 100 100, 100 10, 10 10))')
point = Geometry.from_wkt('SRID=4326;POINT (10 10)')
crs = CoordSystem.from_epsg(4326)
pline = Geometry.from_wkt('LINESTRING (30 10, 10 30, 40 40)', crs)
```

Тип результата Geometry

get_area(u=None)

Рассчитывает площадь, если объект площадной. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в квадратных метрах.

Список 103: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
poly = Polygon([(0,0), (0,2), (2, 2), (2,0)], cs=csMercatorKm)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
print('Area Mercator km:', poly.get_area())
print('Area Mercator m:', poly.get_area(Unit.sq_m))
print('Perimeter Mercator km:', poly.get_perimeter())
print('Perimeter Mercator m:', poly.get_perimeter(Unit.m))
# В Широте/Долготе
poly.coordsystem = CoordSystem.from_prj("1, 104")
print('Area LL m:', poly.get_area())
print('Area LL km:', poly.get_area(Unit.sq_km))
print('Perimeter LL m:', poly.get_perimeter())
print('Perimeter LL km:', poly.get_perimeter(Unit.km))
'''
>>> Area Mercator km: 4.017946986632519
>>> Area Mercator m: 4017946.9866325194
>>> Perimeter Mercator km: 8.017972048421552
>>> Perimeter Mercator m: 8017.972048421552
>>> Area LL m: 49452172514.0342
>>> Area LL km: 49452.1725140342
>>> Perimeter LL m: 889423.5067063896
>>> Perimeter LL km: 889.4235067063896
'''
```

Параметры **u** (Optional[AreaUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_distance(other, u=None)

Производит расчет расстояния до объекта other. Результат возвращает в СК текущего объекта.

Параметры

- **other** (Geometry) – Анализируемый объект.
- **u** (Optional[LinearUnit]) – Единицы измерения, в которых требуется получить результат.

Список 104: Пример.

```
# Создадим геометрию без СК
first = Point(0, 0)
second = Point(10, 0)
print('В плане:', first.get_distance(second))
#Установим разные СК для точек
first.coordsystem = CoordSystem.from_prj("10, 104, 7, 0")
second.coordsystem = CoordSystem.from_prj("1, 104")
print('На сфере м:', first.get_distance(second))
print('На сфере км:', first.get_distance(second, Unit.km))
'''
>>> В плане: 10.0
>>> На сфере м: 1111948.7428468117
>>> На сфере км: 1111.9487428468117
'''
```

Тип результата float

get_length(u=None)
Рассчитывает длину геометрии. Метод применим только для линейных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Список 105: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
ls = Line((0,0), (10,0), csMercatorKm)
print('Length Mercator km:', ls.get_length())
print('Lenght Mercator m:', ls.get_length(Unit.m))
# В Широте/Долготе
csLL = CoordSystem.from_prj("1, 104")
ls = Line((0,0), (10,0), csLL)
print('Length LL m:', ls.get_length())
print('Length LL km:', ls.get_length(Unit.km))
'''
>>> Length Mercator km: 9.988805508567783
>>> Lenght Mercator m: 9988.805508567782
>>> Length LL m: 1111948.7428468117
>>> Length LL km: 1111.9487428468117
'''
```

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_perimeter(u=None)
Рассчитывает периметр геометрии. Метод применим только для площадных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Пример см. get_area()

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

intersection(other)
Возвращает область пересечения с другой геометрией.

Тип результата Geometry

intersects(other)
Возвращает True, если геометрии пересекаются.

Тип результата bool

property is_valid
Проверяет геометрию на валидность.

Тип результата bool

property is_valid_reason
Если геометрия неправильная, возвращает краткую аннотацию причины.

Тип результата str

property name
Возвращает наименование геометрического объекта.

Тип результата str

overlaps(other)
Возвращает True, если пересечение геометрий отличается от обеих геометрий.

Тип результата bool

static point_by_azimuth(point, azimuth, distance, cs=None)
Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.

См.также:

Обратная функция distance_by_points()

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, относительно которой производится расчет. Если задана СК, то в ней.
- **azimuth** (float) – Азимут в градусах, указывающий направление.
- **distance** (float) – Расстояние по азимуту. Задается в метрах.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Pnt

Результат Возвращает результирующую точку.

relate(other)
Проверяет отношения между объектами. Подробнее см. DE-9IM

Тип результата str

remove(idx)
» Удаление геометрии из коллекции.

Параметры idx (int) – Индекс геометрии в коллекции.

reproject(cs)
Перепроецирует геометрию в другую систему координат.

Параметры cs (CoordSystem) – СК, в которой требуется получить объект.

Тип результата Geometry

rotate(point, angle)
Поворот геометрии относительно заданной точки. Поворот производится против часовой стрелки.

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, вокруг которой необходимо произвести поворот.
- **angle** (float) – Угол поворота в градусах.

Тип результата Geometry

scale(kx, ky)
Масштабирует объект по заданным коэффициентам масштабирования. После выполнения операции центр результирующего объекта остается прежним.

Параметры

- **kx** (float) – Масштабирование по координате X.
- **ky** (float) – Масштабирование по координате Y.

Тип результата Geometry

shift(dx, dy)
Смещает объект на заданную величину. Значения задаются в текущей проекции.

Параметры

- **dx** (float) – Сдвиг по координате X.
- **dy** (float) – Сдвиг по координате Y.

Тип результата Geometry

symmetric_difference(other)
Возвращает логический XOR областей геометрий (объединение разниц).

Параметры **other** (Geometry) – Геометрия для анализа.

Тип результата Geometry

to_geojson()
Преобразует геометрию в формат „GeoJSON“

Тип результата str

to_linestring()
Пробует геометрию преобразовать в линейный объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

to_polygon()
Пробует геометрию преобразовать в площадной объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

touches(other)
Возвращает True, если геометрии соприкасаются.

Тип результата bool

try_to_simplified()
Создает копию коллекции при этом пробует упростить ее тип. К примеру, если в коллекции только полигоны, то вернет объект типа MultiPolygon. Если исходная коллекция разнородна, возвращается копия исходной.

Тип результата Union[MultiPoint, MultiLineString, MultiPolygon, GeometryCollection]

property type
Возвращает тип геометрического элемента.

Таблица 67: Возможные значения

Значение	Наименование
Unknown	Не определен
Point	Точка
Line	Линия
LineString	Полилиния
Polygon	Полигон
MultiPoint	Коллекция точек
MultiLineString	Коллекция полилиний
MultiPolygon	Коллекция полигонов
GeometryCollection	Смешанная коллекция
Arc	Дуга
Ellipse	Эллипс
Rectangle	Прямоугольник
RoundedRectangle	Скругленный прямоугольник
Text	Текст

Список 106: Пример.

```
point = Point(10,10)
if point.type == GeometryType.Point:
    print('Это точка')
```

Тип результата GeometryType

union(other)
Возвращает результат объединения двух геометрий.

Тип результата Geometry

within(other)
Возвращает True, если геометрия находится полностью внутри геометрии other.

Тип результата bool

property wkb
Возвращает WKB строку для геометрии.

Тип результата bytes

property wkt
Возвращает WKT строку для геометрии.

Тип результата str

MultiLineString - Коллекция полилиний

class axipy.da.**MultiLineString**(cs=None)

Базовые классы: axipy.da.GeometryCollection

Коллекция полилиний. Может содержать только объекты типа полилиния.

Параметры cs (Optional[CoordSystem]) – Система Координат, в которой создается геометрия.

Список 107: Пример.

```
msl = MultiLineString() # Создадим саму коллекцию.
ls = LineString([(1, 2), (3, 4), (5, 6), (7, 8)])
msl.append(ls) # Добавим как объект по ссылке
msl.append(LineString([(11, 12), (13, 14), (15, 16)])) # Добавим как объект
msl.append([(21, 22), (23, 24), (25, 26)]) # Добавим как перечень точек как
↪:class:`list`
msl[2].points[1] = (101, 102) # Обновим значение точки 3 полилинии по индексу 2.
msl.remove(0) # Удалим первый объект из коллекции.
msl[1].points.remove(2) # Удалим точку с индексом 1 из полилинии 2
msl[0] = [(101, 102), (103, 104), (105, 106), (107, 108)] # Обновим первую
↪геометрию
```

Methods:

affine_transform(trans)	Трансформирует объект исходя из заданных параметров трансформации.
almost_equals(other, tolerance)	Производит примерное сравнения с другой геометрией в пределах заданной точности.
append(*value)	Добавление геометрии в коллекцию.
boundary()	Возвращает границы геометрии в виде полилинии.
buffer(distance[, resolution, capStyle, ...])	Производит построение буфера.
centroid()	Возвращает центроид геометрии.
clone()	Создает копию объекта.
contains(other)	Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.
convex_hull()	Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.
covers(other)	Возвращает True, если геометрия охватывает геометрию other.
crosses(other)	Возвращает True, если при пересечении геометрий объекты частично пересекаются.
difference(other)	Возвращает область первой геометрии, которая не пересечена второй геометрией.
disjoint(other)	Возвращает True, если геометрии не пересекаются и не соприкасаются.

continues on next page

Таблица 68 – продолжение с предыдущей страницы

distance_by_points(start, end[, cs])	Производит расчет расстояния между двумя точками и азимут от первой до второй точки.
envelope()	Возвращает полигон, описывающий заданную геометрию.
equals(other)	Производит сравнение с другой геометрией.
from_geojson(json[, cs])	Возвращает геометрию из ее „GeoJSON“ представления.
from_wkb(wkb[, coordsystem])	Создает геометрический объект из строки формата WKB.
from_wkt(wkt[, coordsystem])	Создает геометрический объект из строки формата WKT.
get_area([u])	Рассчитывает площадь, если объект площадной.
get_distance(other[, u])	Производит расчет расстояния до объекта other.
get_length([u])	Рассчитывает длину геометрии.
get_perimeter([u])	Рассчитывает периметр геометрии.
intersection(other)	Возвращает область пересечения с другой геометрией.
intersects(other)	Возвращает True, если геометрии пересекаются.
overlaps(other)	Возвращает True, если пересечение геометрий отличается от обеих геометрий.
point_by_azimuth(point, azimuth, distance[, cs])	Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.
relate(other)	Проверяет отношения между объектами.
remove(idx)	» Удаление геометрии из коллекции.
reproject(cs)	Перепроецирует геометрию в другую систему координат.
rotate(point, angle)	Поворот геометрии относительно заданной точки.
scale(kx, ky)	Масштабирует объект по заданным коэффициентам масштабирования.
shift(dx, dy)	Смещает объект на заданную величину.
symmetric_difference(other)	Возвращает логический XOR областей геометрий (объединение разниц).
to_geojson()	Преобразует геометрию в формат „GeoJSON“
to_linestring()	Пробует геометрию преобразовать в линейный объект.
to_polygon()	Пробует геометрию преобразовать в площадной объект.
touches(other)	Возвращает True, если геометрии соприкасаются.

continues on next page

Таблица 68 – продолжение с предыдущей страницы

try_to_simplified()	Создает копию коллекции при этом пробует упростить ее тип.
union(other)	Возвращает результат объединения двух геометрий.
within(other)	Возвращает True, если геометрия находится полностью внутри геометрии other.

Attributes:

bounds	Возвращает минимальный ограничивающий прямоугольник.
coordsystem	Система Координат (СК) геометрии.
is_valid	Проверяет геометрию на валидность.
is_valid_reason	Если геометрия неправильная, возвращает краткую аннотацию причины.
name	Возвращает наименование геометрического объекта.
type	Возвращает тип геометрического элемента.
wkb	Возвращает WKB строку для геометрии.
wkt	Возвращает WKT строку для геометрии.

affine_transform(trans)
Трансформирует объект исходя из заданных параметров трансформации.

См.также:

Для простых операций типа сдвиг, масштабирование и поворот, рекомендуется использовать shift(), scale(), rotate() соответственно.

Параметры **trans** (QTransform) – Матрица трансформации.

Тип результата Geometry

almost_equals(other, tolerance)
Производит примерное сравнения с другой геометрией в пределах заданной точности.

Параметры

- **other** (Geometry) – Сравниваемый объект.
- **tolerance** (float) – Точность сравнения.

Тип результата bool

Результат Возвращает True, если геометрии равны в пределах заданного отклонения.

append(*value)
Добавление геометрии в коллекцию. Задание параметров аналогично указанию их при создании объектов конкретного типа. Если опустить указание класса, то

для одной точки или пары значений float будет создан точечный объект Point, если точек больше, - объект класса LineString.

Список 108: Пример.

```
# Создадим коллекцию и добавим в нее несколько объектов различного типа.
coll = GeometryCollection()
coll.append((1,2)) # Точка
coll.append(1,2) # Точка
coll.append([(3,4), (5, 5), (10, 0)]) # Полилиния в виде :class:`list`. Можно
↪это сделать через конструктор :class:`LinearString`.
coll.append((3,4), (5, 5), (10, 0)) # Полилиния
coll.append(Polygon([(3,4), (5, 5), (10, 0)])) # Полигон
```

boundary()
Возвращает границы геометрии в виде полилинии.

Тип результата Geometry

property bounds
Возвращает минимальный ограничивающий прямоугольник.

Тип результата Rect

buffer(distance, resolution=16, capStyle=1, joinStyle=1, mitreLimit=5.0)
Производит построение буфера.

Параметры

- **distance** (float) – Ширина буфера.
- **resolution** (int) – Количество сегментов на квадрант.
- **capStyle** (int) – Стиль окончания.
- **joinStyle** (int) – Стиль соединения.
- **mitreLimit** (float) – Предел среза.

Тип результата Geometry

centroid()
Возвращает центроид геометрии.

Тип результата Geometry

clone()
Создает копию объекта.

Тип результата Geometry

contains(other)
Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.

Тип результата bool

convex_hull()
Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.

Тип результата Geometry

property coordsystem
Система Координат (СК) геометрии.

Тип результата CoordSystem

covers(other)
Возвращает True, если геометрия охватывает геометрию other.

Тип результата bool

crosses(other)
Возвращает True, если при пересечении геометрий объекты частично пересекаются.

Тип результата bool

difference(other)
Возвращает область первой геометрии, которая не пересечена второй геометрией.

Тип результата Geometry

disjoint(other)
Возвращает True, если геометрии не пересекаются и не соприкасаются.

Тип результата bool

static distance_by_points(start, end, cs=None)
Производит расчет расстояния между двумя точками и азимут от первой до второй точки.

См.также:

Обратная функция point_by_azimuth()

Параметры

- **start** (Union[Pnt, Tuple[float, float]]) – Начальная точка. Если задана СК, то координаты в ней.
- **end** (Union[Pnt, Tuple[float, float]]) – Конечная точка. Если задана СК, то координаты в ней.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Tuple

Результат Возвращается пара значений (расстояние в метрах, азимут в градусах)

envelope()
Возвращает полигон, описывающий заданную геометрию.

Тип результата Geometry

equals(other)
Производит сравнение с другой геометрией.

Параметры **other** (Geometry) – Сравниваемая геометрия.

Тип результата bool

Результат Возвращает True, если геометрии равны.

static from_geojson(json, cs=None)
Возвращает геометрию из ее „GeoJSON“ представления.

Параметры

- **json** (str) – json представление в виде строки.
- **cs** (Optional[CoordSystem]) – Система координат.

Тип результата Geometry

static from_wkb(wkb, coordsystem=None)

Создает геометрический объект из строки формата WKB.

Параметры

- **wkb** (bytes) – Строка WKB
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии.

Список 109: Пример.

```
wkb = b'\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00$@\x00\x00\x00\x00\x00\x00
↪$@'
pnt = Geometry.from_wkb(wkb)
```

Тип результата Geometry

static from_wkt(wkt, coordsystem=None)

Создает геометрический объект из строки формата WKT.

Параметры

- **wkt** (str) – Строка WKT. Допустимо задание строки в формате с указанием SRID (EWKT). В данном случае система координат для создаваемой геометрии будет установлено исходя их этого значения.
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии. Если строка задана в виде EWKT и указано значение SRID, игнорируется.

Список 110: Пример.

```
polygon = Geometry.from_wkt('POLYGON ((10 10, 100 100, 100 10, 10 10))')
point = Geometry.from_wkt('SRID=4326;POINT (10 10)')
crs = CoordSystem.from_epsg(4326)
pline = Geometry.from_wkt('LINESTRING (30 10, 10 30, 40 40)', crs)
```

Тип результата Geometry

get_area(u=None)

Рассчитывает площадь, если объект площадной. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в квадратных метрах.

Список 111: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
poly = Polygon([(0,0), (0,2), (2, 2), (2,0)], cs=csMercatorKm)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
print('Area Mercator km:', poly.get_area())
print('Area Mercator m:', poly.get_area(Unit.sq_m))
print('Perimeter Mercator km:', poly.get_perimeter())
print('Perimeter Mercator m:', poly.get_perimeter(Unit.m))
# В Широте/Долготе
poly.coordsystem = CoordSystem.from_prj("1, 104")
print('Area LL m:', poly.get_area())
print('Area LL km:', poly.get_area(Unit.sq_km))
print('Perimeter LL m:', poly.get_perimeter())
print('Perimeter LL km:', poly.get_perimeter(Unit.km))
'''
>>> Area Mercator km: 4.017946986632519
>>> Area Mercator m: 4017946.9866325194
>>> Perimeter Mercator km: 8.017972048421552
>>> Perimeter Mercator m: 8017.972048421552
>>> Area LL m: 49452172514.0342
>>> Area LL km: 49452.1725140342
>>> Perimeter LL m: 889423.5067063896
>>> Perimeter LL km: 889.4235067063896
'''
```

Параметры **u** (Optional[AreaUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_distance(other, u=None)

Производит расчет расстояния до объекта other. Результат возвращает в СК текущего объекта.

Параметры

- **other** (Geometry) – Анализируемый объект.
- **u** (Optional[LinearUnit]) – Единицы измерения, в которых требуется получить результат.

Список 112: Пример.

```
# Создадим геометрию без СК
first = Point(0, 0)
second = Point(10, 0)
print('В плане:', first.get_distance(second))
#Установим разные СК для точек
first.coordsystem = CoordSystem.from_prj("10, 104, 7, 0")
second.coordsystem = CoordSystem.from_prj("1, 104")
print('На сфере м:', first.get_distance(second))
print('На сфере км:', first.get_distance(second, Unit.km))
'''
>>> В плане: 10.0
>>> На сфере м: 1111948.7428468117
>>> На сфере км: 1111.9487428468117
'''
```

Тип результата float

get_length(u=None)
Рассчитывает длину геометрии. Метод применим только для линейных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Список 113: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
ls = Line((0,0), (10,0), csMercatorKm)
print('Length Mercator km:', ls.get_length())
print('Lenght Mercator m:', ls.get_length(Unit.m))
# В Широте/Долготе
csLL = CoordSystem.from_prj("1, 104")
ls = Line((0,0), (10,0), csLL)
print('Length LL m:', ls.get_length())
print('Length LL km:', ls.get_length(Unit.km))
'''
>>> Length Mercator km: 9.988805508567783
>>> Lenght Mercator m: 9988.805508567782
>>> Length LL m: 1111948.7428468117
>>> Length LL km: 1111.9487428468117
'''
```

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_perimeter(u=None)
Рассчитывает периметр геометрии. Метод применим только для площадных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Пример см. get_area()

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

intersection(other)
Возвращает область пересечения с другой геометрией.

Тип результата Geometry

intersects(other)
Возвращает True, если геометрии пересекаются.

Тип результата bool

property is_valid
Проверяет геометрию на валидность.

Тип результата bool

property is_valid_reason
Если геометрия неправильная, возвращает краткую аннотацию причины.

Тип результата str

property name
Возвращает наименование геометрического объекта.

Тип результата str

overlaps(other)
Возвращает True, если пересечение геометрий отличается от обеих геометрий.

Тип результата bool

static point_by_azimuth(point, azimuth, distance, cs=None)
Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.

См.также:

Обратная функция distance_by_points()

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, относительно которой производится расчет. Если задана СК, то в ней.
- **azimuth** (float) – Азимут в градусах, указывающий направление.
- **distance** (float) – Расстояние по азимуту. Задается в метрах.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Pnt

Результат Возвращает результирующую точку.

relate(other)
Проверяет отношения между объектами. Подробнее см. DE-9IM

Тип результата str

remove(idx)
» Удаление геометрии из коллекции.

Параметры idx (int) – Индекс геометрии в коллекции.

reproject(cs)
Перепроецирует геометрию в другую систему координат.

Параметры cs (CoordSystem) – СК, в которой требуется получить объект.

Тип результата Geometry

rotate(point, angle)
Поворот геометрии относительно заданной точки. Поворот производится против часовой стрелки.

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, вокруг которой необходимо произвести поворот.
- **angle** (float) – Угол поворота в градусах.

Тип результата Geometry

scale(kx, ky)
Масштабирует объект по заданным коэффициентам масштабирования. После выполнения операции центр результирующего объекта остается прежним.

Параметры

- **kx** (float) – Масштабирование по координате X.
- **ky** (float) – Масштабирование по координате Y.

Тип результата Geometry

shift(dx, dy)
Смещает объект на заданную величину. Значения задаются в текущей проекции.

Параметры

- **dx** (float) – Сдвиг по координате X.
- **dy** (float) – Сдвиг по координате Y.

Тип результата Geometry

symmetric_difference(other)
Возвращает логический XOR областей геометрий (объединение разниц).

Параметры **other** (Geometry) – Геометрия для анализа.

Тип результата Geometry

to_geojson()
Преобразует геометрию в формат „GeoJSON“

Тип результата str

to_linestring()
Пробует геометрию преобразовать в линейный объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

to_polygon()
Пробует геометрию преобразовать в площадной объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

touches(other)
Возвращает True, если геометрии соприкасаются.

Тип результата bool

try_to_simplified()
Создает копию коллекции при этом пробует упростить ее тип. К примеру, если в коллекции только полигоны, то вернет объект типа MultiPolygon. Если исходная коллекция разнородна, возвращается копия исходной.

Тип результата Union[MultiPoint, MultiLineString, MultiPolygon, GeometryCollection]

property type
Возвращает тип геометрического элемента.

Таблица 70: Возможные значения

Значение	Наименование
Unknown	Не определен
Point	Точка
Line	Линия
LineString	Полилиния
Polygon	Полигон
MultiPoint	Коллекция точек
MultiLineString	Коллекция полилиний
MultiPolygon	Коллекция полигонов
GeometryCollection	Смешанная коллекция
Arc	Дуга
Ellipse	Эллипс
Rectangle	Прямоугольник
RoundedRectangle	Скругленный прямоугольник
Text	Текст

Список 114: Пример.

```
point = Point(10,10)
if point.type == GeometryType.Point:
    print('Это точка')
```

Тип результата GeometryType

union(other)
Возвращает результат объединения двух геометрий.

Тип результата Geometry

within(other)
Возвращает True, если геометрия находится полностью внутри геометрии other.

Тип результата bool

property wkb
Возвращает WKB строку для геометрии.

Тип результата bytes

property wkt
Возвращает WKT строку для геометрии.

Тип результата str

MultiPolygon - Коллекция полигонов

class axipy.da.**MultiPolygon**(cs=None)

Базовые классы: axipy.da.GeometryCollection

Коллекция полигонов. Может содержать только объекты типа полигон.

Параметры cs (Optional[CoordSystem]) – Система Координат, в которой создается геометрия.

Список 115: Пример.

```
poly = Polygon((0, 0), (1, 10), (10, 1))
poly.holes.append([(2,2), (2,4), (5,3)]) # Добавим дырку
mpoly = MultiPolygon() # Создадим саму коллекцию.
mpoly.append([(1, 2), (3, 4), (5, 6), (7, 8)]) # Добавим полигон в виде списка␣
↪точек
mpoly.append(poly) # Добавим ранее созданный с дыркой
mpoly[1].holes[0][1] = (99,99) # Изменение второй точки дырки
mpoly[1].points[0] = (0, 0) # Заменим первую (она же последняя) точку полигона
poly2 = Polygon([(11, 12), (13, 14), (15, 16), (17, 18)])
mpoly[0] = poly2 # Полностью заменим первый полигон
```

Methods:

affine_transform(trans)	Трансформирует объект исходя из заданных параметров трансформации.
almost_equals(other, tolerance)	Производит примерное сравнения с другой геометрией в пределах заданной точности.
append(*value)	Добавление геометрии в коллекцию.
boundary()	Возвращает границы геометрии в виде полилинии.
buffer(distance[, resolution, capStyle, ...])	Производит построение буфера.
centroid()	Возвращает центроид геометрии.
clone()	Создает копию объекта.
contains(other)	Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.
convex_hull()	Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.
covers(other)	Возвращает True, если геометрия охватывает геометрию other.
crosses(other)	Возвращает True, если при пересечении геометрий объекты частично пересекаются.
difference(other)	Возвращает область первой геометрии, которая не пересечена второй геометрией.
disjoint(other)	Возвращает True, если геометрии не пересекаются и не соприкасаются.

continues on next page

Таблица 71 – продолжение с предыдущей страницы

distance_by_points(start, end[, cs])	Производит расчет расстояния между двумя точками и азимут от первой до второй точки.
envelope()	Возвращает полигон, описывающий заданную геометрию.
equals(other)	Производит сравнение с другой геометрией.
from_geojson(json[, cs])	Возвращает геометрию из ее „GeoJSON“ представления.
from_wkb(wkb[, coordsystem])	Создает геометрический объект из строки формата WKB.
from_wkt(wkt[, coordsystem])	Создает геометрический объект из строки формата WKT.
get_area([u])	Рассчитывает площадь, если объект площадной.
get_distance(other[, u])	Производит расчет расстояния до объекта other.
get_length([u])	Рассчитывает длину геометрии.
get_perimeter([u])	Рассчитывает периметр геометрии.
intersection(other)	Возвращает область пересечения с другой геометрией.
intersects(other)	Возвращает True, если геометрии пересекаются.
overlaps(other)	Возвращает True, если пересечение геометрий отличается от обеих геометрий.
point_by_azimuth(point, azimuth, distance[, cs])	Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.
relate(other)	Проверяет отношения между объектами.
remove(idx)	» Удаление геометрии из коллекции.
reproject(cs)	Перепроецирует геометрию в другую систему координат.
rotate(point, angle)	Поворот геометрии относительно заданной точки.
scale(kx, ky)	Масштабирует объект по заданным коэффициентам масштабирования.
shift(dx, dy)	Смещает объект на заданную величину.
symmetric_difference(other)	Возвращает логический XOR областей геометрий (объединение разниц).
to_geojson()	Преобразует геометрию в формат „GeoJSON“
to_linestring()	Пробует геометрию преобразовать в линейный объект.
to_polygon()	Пробует геометрию преобразовать в площадной объект.
touches(other)	Возвращает True, если геометрии соприкасаются.

continues on next page

Таблица 71 – продолжение с предыдущей страницы

try_to_simplified()	Создает копию коллекции при этом пробует упростить ее тип.
union(other)	Возвращает результат объединения двух геометрий.
within(other)	Возвращает True, если геометрия находится полностью внутри геометрии other.

Attributes:

bounds	Возвращает минимальный ограничивающий прямоугольник.
coordsystem	Система Координат (СК) геометрии.
is_valid	Проверяет геометрию на валидность.
is_valid_reason	Если геометрия неправильная, возвращает краткую аннотацию причины.
name	Возвращает наименование геометрического объекта.
type	Возвращает тип геометрического элемента.
wkb	Возвращает WKB строку для геометрии.
wkt	Возвращает WKT строку для геометрии.

affine_transform(trans)
Трансформирует объект исходя из заданных параметров трансформации.

См.также:

Для простых операций типа сдвиг, масштабирование и поворот, рекомендуется использовать shift(), scale(), rotate() соответственно.

Параметры **trans** (QTransform) – Матрица трансформации.

Тип результата Geometry

almost_equals(other, tolerance)
Производит примерное сравнения с другой геометрией в пределах заданной точности.

Параметры

- **other** (Geometry) – Сравниваемый объект.
- **tolerance** (float) – Точность сравнения.

Тип результата bool

Результат Возвращает True, если геометрии равны в пределах заданного отклонения.

append(*value)
Добавление геометрии в коллекцию. Задание параметров аналогично указанию их при создании объектов конкретного типа. Если опустить указание класса, то

для одной точки или пары значений float будет создан точечный объект Point, если точек больше, - объект класса LineString.

Список 116: Пример.

```
# Создадим коллекцию и добавим в нее несколько объектов различного типа.
coll = GeometryCollection()
coll.append((1,2)) # Точка
coll.append(1,2) # Точка
coll.append([(3,4), (5, 5), (10, 0)]) # Полилиния в виде :class:`list`. Можно
↪это сделать через конструктор :class:`LinearString`.
coll.append((3,4), (5, 5), (10, 0)) # Полилиния
coll.append(Polygon([(3,4), (5, 5), (10, 0)])) # Полигон
```

boundary()
Возвращает границы геометрии в виде полилинии.

Тип результата Geometry

property bounds
Возвращает минимальный ограничивающий прямоугольник.

Тип результата Rect

buffer(distance, resolution=16, capStyle=1, joinStyle=1, mitreLimit=5.0)
Производит построение буфера.

Параметры

- **distance** (float) – Ширина буфера.
- **resolution** (int) – Количество сегментов на квадрант.
- **capStyle** (int) – Стиль окончания.
- **joinStyle** (int) – Стиль соединения.
- **mitreLimit** (float) – Предел среза.

Тип результата Geometry

centroid()
Возвращает центроид геометрии.

Тип результата Geometry

clone()
Создает копию объекта.

Тип результата Geometry

contains(other)
Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.

Тип результата bool

convex_hull()
Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.

Тип результата Geometry

property coordsystem
Система Координат (СК) геометрии.

Тип результата CoordSystem

covers(other)
Возвращает True, если геометрия охватывает геометрию other.

Тип результата bool

crosses(other)
Возвращает True, если при пересечении геометрий объекты частично пересекаются.

Тип результата bool

difference(other)
Возвращает область первой геометрии, которая не пересечена второй геометрией.

Тип результата Geometry

disjoint(other)
Возвращает True, если геометрии не пересекаются и не соприкасаются.

Тип результата bool

static distance_by_points(start, end, cs=None)
Производит расчет расстояния между двумя точками и азимут от первой до второй точки.

См.также:

Обратная функция point_by_azimuth()

Параметры

- **start** (Union[Pnt, Tuple[float, float]]) – Начальная точка. Если задана СК, то координаты в ней.
- **end** (Union[Pnt, Tuple[float, float]]) – Конечная точка. Если задана СК, то координаты в ней.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Tuple

Результат Возвращается пара значений (расстояние в метрах, азимут в градусах)

envelope()
Возвращает полигон, описывающий заданную геометрию.

Тип результата Geometry

equals(other)
Производит сравнение с другой геометрией.

Параметры **other** (Geometry) – Сравниваемая геометрия.

Тип результата bool

Результат Возвращает True, если геометрии равны.

static from_geojson(json, cs=None)
Возвращает геометрию из ее „GeoJSON“ представления.

Параметры

- **json** (str) – json представление в виде строки.
- **cs** (Optional[CoordSystem]) – Система координат.

Тип результата Geometry

static from_wkb(wkb, coordsystem=None)
Создает геометрический объект из строки формата WKB.

Параметры

- **wkb** (bytes) – Строка WKB
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии.

Список 117: Пример.

```
wkb = b'\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00$@\x00\x00\x00\x00\x00\x00
↪$@'
pnt = Geometry.from_wkb(wkb)
```

Тип результата Geometry

static from_wkt(wkt, coordsystem=None)
Создает геометрический объект из строки формата WKT.

Параметры

- **wkt** (str) – Строка WKT. Допустимо задание строки в формате с указанием SRID (EWKT). В данном случае система координат для создаваемой геометрии будет установлено исходя их этого значения.
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии. Если строка задана в виде EWKT и указано значение SRID, игнорируется.

Список 118: Пример.

```
polygon = Geometry.from_wkt('POLYGON ((10 10, 100 100, 100 10, 10 10))')
point = Geometry.from_wkt('SRID=4326;POINT (10 10)')
crs = CoordSystem.from_epsg(4326)
pline = Geometry.from_wkt('LINESTRING (30 10, 10 30, 40 40)', crs)
```

Тип результата Geometry

get_area(u=None)
Рассчитывает площадь, если объект площадной. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в квадратных метрах.

Список 119: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
poly = Polygon([(0,0), (0,2), (2, 2), (2,0)], cs=csMercatorKm)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
print('Area Mercator km:', poly.get_area())
print('Area Mercator m:', poly.get_area(Unit.sq_m))
print('Perimeter Mercator km:', poly.get_perimeter())
print('Perimeter Mercator m:', poly.get_perimeter(Unit.m))
# В Широте/Долготе
poly.coordsystem = CoordSystem.from_prj("1, 104")
print('Area LL m:', poly.get_area())
print('Area LL km:', poly.get_area(Unit.sq_km))
print('Perimeter LL m:', poly.get_perimeter())
print('Perimeter LL km:', poly.get_perimeter(Unit.km))
'''
>>> Area Mercator km: 4.017946986632519
>>> Area Mercator m: 4017946.9866325194
>>> Perimeter Mercator km: 8.017972048421552
>>> Perimeter Mercator m: 8017.972048421552
>>> Area LL m: 49452172514.0342
>>> Area LL km: 49452.1725140342
>>> Perimeter LL m: 889423.5067063896
>>> Perimeter LL km: 889.4235067063896
'''
```

Параметры **u** (Optional[AreaUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_distance(other, u=None)

Производит расчет расстояния до объекта other. Результат возвращает в СК текущего объекта.

Параметры

- **other** (Geometry) – Анализируемый объект.
- **u** (Optional[LinearUnit]) – Единицы измерения, в которых требуется получить результат.

Список 120: Пример.

```
# Создадим геометрию без СК
first = Point(0, 0)
second = Point(10, 0)
print('В плане:', first.get_distance(second))
#Установим разные СК для точек
first.coordsystem = CoordSystem.from_prj("10, 104, 7, 0")
second.coordsystem = CoordSystem.from_prj("1, 104")
print('На сфере м:', first.get_distance(second))
print('На сфере км:', first.get_distance(second, Unit.km))
'''
>>> В плане: 10.0
>>> На сфере м: 1111948.7428468117
>>> На сфере км: 1111.9487428468117
'''
```

Тип результата float

get_length(u=None)
Рассчитывает длину геометрии. Метод применим только для линейных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Список 121: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
ls = Line((0,0), (10,0), csMercatorKm)
print('Length Mercator km:', ls.get_length())
print('Lenght Mercator m:', ls.get_length(Unit.m))
# В Широте/Долготе
csLL = CoordSystem.from_prj("1, 104")
ls = Line((0,0), (10,0), csLL)
print('Length LL m:', ls.get_length())
print('Length LL km:', ls.get_length(Unit.km))
'''
>>> Length Mercator km: 9.988805508567783
>>> Lenght Mercator m: 9988.805508567782
>>> Length LL m: 1111948.7428468117
>>> Length LL km: 1111.9487428468117
'''
```

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_perimeter(u=None)
Рассчитывает периметр геометрии. Метод применим только для площадных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Пример см. get_area()

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

intersection(other)
Возвращает область пересечения с другой геометрией.

Тип результата Geometry

intersects(other)
Возвращает True, если геометрии пересекаются.

Тип результата bool

property is_valid
Проверяет геометрию на валидность.

Тип результата bool

property is_valid_reason
Если геометрия неправильная, возвращает краткую аннотацию причины.

Тип результата str

property name
Возвращает наименование геометрического объекта.

Тип результата str

overlaps(other)
Возвращает True, если пересечение геометрий отличается от обеих геометрий.

Тип результата bool

static point_by_azimuth(point, azimuth, distance, cs=None)
Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.

См.также:

Обратная функция distance_by_points()

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, относительно которой производится расчет. Если задана СК, то в ней.
- **azimuth** (float) – Азимут в градусах, указывающий направление.
- **distance** (float) – Расстояние по азимуту. Задается в метрах.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Pnt

Результат Возвращает результирующую точку.

relate(other)
Проверяет отношения между объектами. Подробнее см. DE-9IM

Тип результата str

remove(idx)
» Удаление геометрии из коллекции.

Параметры idx (int) – Индекс геометрии в коллекции.

reproject(cs)
Перепроецирует геометрию в другую систему координат.

Параметры cs (CoordSystem) – СК, в которой требуется получить объект.

Тип результата Geometry

rotate(point, angle)
Поворот геометрии относительно заданной точки. Поворот производится против часовой стрелки.

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, вокруг которой необходимо произвести поворот.
- **angle** (float) – Угол поворота в градусах.

Тип результата Geometry

scale(kx, ky)
Масштабирует объект по заданным коэффициентам масштабирования. После выполнения операции центр результирующего объекта остается прежним.

Параметры

- **kx** (float) – Масштабирование по координате X.
- **ky** (float) – Масштабирование по координате Y.

Тип результата Geometry

shift(dx, dy)
Смещает объект на заданную величину. Значения задаются в текущей проекции.

Параметры

- **dx** (float) – Сдвиг по координате X.
- **dy** (float) – Сдвиг по координате Y.

Тип результата Geometry

symmetric_difference(other)
Возвращает логический XOR областей геометрий (объединение разниц).

Параметры **other** (Geometry) – Геометрия для анализа.

Тип результата Geometry

to_geojson()
Преобразует геометрию в формат „GeoJSON“

Тип результата str

to_linestring()
Пробует геометрию преобразовать в линейный объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

to_polygon()
Пробует геометрию преобразовать в площадной объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

touches(other)
Возвращает True, если геометрии соприкасаются.

Тип результата bool

try_to_simplified()
Создает копию коллекции при этом пробует упростить ее тип. К примеру, если в коллекции только полигоны, то вернет объект типа MultiPolygon. Если исходная коллекция разнородна, возвращается копия исходной.

Тип результата Union[MultiPoint, MultiLineString, MultiPolygon, GeometryCollection]

property type
Возвращает тип геометрического элемента.

Таблица 73: Возможные значения

Значение	Наименование
Unknown	Не определен
Point	Точка
Line	Линия
LineString	Полилиния
Polygon	Полигон
MultiPoint	Коллекция точек
MultiLineString	Коллекция полилиний
MultiPolygon	Коллекция полигонов
GeometryCollection	Смешанная коллекция
Arc	Дуга
Ellipse	Эллипс
Rectangle	Прямоугольник
RoundedRectangle	Скругленный прямоугольник
Text	Текст

Список 122: Пример.

```
point = Point(10,10)
if point.type == GeometryType.Point:
    print('Это точка')
```

Тип результата GeometryType

union(other)
Возвращает результат объединения двух геометрий.

Тип результата Geometry

within(other)
Возвращает True, если геометрия находится полностью внутри геометрии other.

Тип результата bool

property wkb
Возвращает WKB строку для геометрии.

Тип результата bytes

property wkt
Возвращает WKT строку для геометрии.

Тип результата str

Rectangle - Прямоугольник

class axipy.mi.**Rectangle**(*par, cs=None)

Базовые классы: axipy.da.Geometry

Геометрический объект типа прямоугольник.

Параметры

- **par** (Union[Rect, float]) – Прямоугольник класса Rect или перечень координат через запятую (xmin, ymin, xmax, ymax) или списком list.
- **cs** (Optional[CoordSystem]) – Система Координат, в которой создается геометрия.

Список 123: Пример.

```
r1 = Rectangle(0, 0, 40, 20) # Создадим объект через перечень координат.
r2 = Rectangle(Rect(0, 0, 40, 20)) # Создадим объект передав объект :class:`Rect`.
r3 = Rectangle([0, 0, 40, 20]) # Создадим объект передав списком :class:`list`.
```

Methods:

affine_transform(trans)	Трансформирует объект исходя из заданных параметров трансформации.
almost_equals(other, tolerance)	Производит примерное сравнения с другой геометрией в пределах заданной точности.
boundary()	Возвращает границы геометрии в виде полилинии.
buffer(distance[, resolution, capStyle, ...])	Производит построение буфера.
centroid()	Возвращает центроид геометрии.
clone()	Создает копию объекта.
contains(other)	Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.
convex_hull()	Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.
covers(other)	Возвращает True, если геометрия охватывает геометрию other.
crosses(other)	Возвращает True, если при пересечении геометрий объекты частично пересекаются.
difference(other)	Возвращает область первой геометрии, которая не пересечена второй геометрией.
disjoint(other)	Возвращает True, если геометрии не пересекаются и не соприкасаются.
distance_by_points(start, end[, cs])	Производит расчет расстояния между двумя точками и азимут от первой до второй точки.

continues on next page

Таблица 74 – продолжение с предыдущей страницы

envelope()	Возвращает полигон, описывающий заданную геометрию.
equals(other)	Производит сравнение с другой геометрией.
from_geojson(json[, cs])	Возвращает геометрию из ее „GeoJSON“ представления.
from_wkb(wkb[, coordsystem])	Создает геометрический объект из строки формата WKB.
from_wkt(wkt[, coordsystem])	Создает геометрический объект из строки формата WKT.
get_area([u])	Рассчитывает площадь, если объект площадной.
get_distance(other[, u])	Производит расчет расстояния до объекта other.
get_length([u])	Рассчитывает длину геометрии.
get_perimeter([u])	Рассчитывает периметр геометрии.
intersection(other)	Возвращает область пересечения с другой геометрией.
intersects(other)	Возвращает True, если геометрии пересекаются.
overlaps(other)	Возвращает True, если пересечение геометрий отличается от обеих геометрий.
point_by_azimuth(point, azimuth, distance[, cs])	Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.
relate(other)	Проверяет отношения между объектами.
reproject(cs)	Перепроецирует геометрию в другую систему координат.
rotate(point, angle)	Поворот геометрии относительно заданной точки.
scale(kx, ky)	Масштабирует объект по заданным коэффициентам масштабирования.
shift(dx, dy)	Смещает объект на заданную величину.
symmetric_difference(other)	Возвращает логический XOR областей геометрий (объединение разниц).
to_geojson()	Преобразует геометрию в формат „GeoJSON“
to_linestring()	Пробует геометрию преобразовать в линейный объект.
to_polygon()	Пробует геометрию преобразовать в площадной объект.
touches(other)	Возвращает True, если геометрии соприкасаются.
union(other)	Возвращает результат объединения двух геометрий.

continues on next page

Таблица 74 – продолжение с предыдущей страницы

within(other)	Возвращает True, если геометрия находится полностью внутри геометрии other.

Attributes:

bounds	Возвращает минимальный ограничивающий прямоугольник.
coordsystem	Система Координат (СК) геометрии.
is_valid	Проверяет геометрию на валидность.
is_valid_reason	Если геометрия неправильная, возвращает краткую аннотацию причины.
name	Возвращает наименование геометрического объекта.
type	Возвращает тип геометрического элемента.
wkb	Возвращает WKB строку для геометрии.
wkt	Возвращает WKT строку для геометрии.
xmax	Максимальное значение X.
xmin	Минимальное значение X.
ymax	Максимальное значение Y.
ymin	Минимальное значение Y.

affine_transform(trans)
Трансформирует объект исходя из заданных параметров трансформации.

См.также:

Для простых операций типа сдвиг, масштабирование и поворот, рекомендуется использовать shift(), scale(), rotate() соответственно.

Параметры **trans** (QTransform) – Матрица трансформации.

Тип результата Geometry

almost_equals(other, tolerance)
Производит примерное сравнения с другой геометрией в пределах заданной точности.

Параметры

- **other** (Geometry) – Сравниваемый объект.
- **tolerance** (float) – Точность сравнения.

Тип результата bool

Результат Возвращает True, если геометрии равны в пределах заданного отклонения.

boundary()
Возвращает границы геометрии в виде полилинии.

Тип результата Geometry

property bounds
Возвращает минимальный ограничивающий прямоугольник.

Тип результата Rect

buffer(distance, resolution=16, capStyle=1, joinStyle=1, mitreLimit=5.0)
Производит построение буфера.

Параметры

- **distance** (float) – Ширина буфера.
- **resolution** (int) – Количество сегментов на квадрант.
- **capStyle** (int) – Стиль окончания.
- **joinStyle** (int) – Стиль соединения.
- **mitreLimit** (float) – Предел среза.

Тип результата Geometry

centroid()
Возвращает центроид геометрии.

Тип результата Geometry

clone()
Создает копию объекта.

Тип результата Geometry

contains(other)
Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.

Тип результата bool

convex_hull()
Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.

Тип результата Geometry

property coordsystem
Система Координат (СК) геометрии.

Тип результата CoordSystem

covers(other)
Возвращает True, если геометрия охватывает геометрию other.

Тип результата bool

crosses(other)
Возвращает True, если при пересечении геометрий объекты частично пересекаются.

Тип результата bool

difference(other)
Возвращает область первой геометрии, которая не пересечена второй геометрией.

Тип результата Geometry

disjoint(other)
Возвращает True, если геометрии не пересекаются и не соприкасаются.

Тип результата bool

static distance_by_points(start, end, cs=None)

Производит расчет расстояния между двумя точками и азимут от первой до второй точки.

См.также:

Обратная функция point_by_azimuth()

Параметры

- **start** (Union[Pnt, Tuple[float, float]]) – Начальная точка. Если задана СК, то координаты в ней.
- **end** (Union[Pnt, Tuple[float, float]]) – Конечная точка. Если задана СК, то координаты в ней.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Tuple

Результат Возвращается пара значений (расстояние в метрах, азимут в градусах)

envelope()

Возвращает полигон, описывающий заданную геометрию.

Тип результата Geometry

equals(other)

Производит сравнение с другой геометрией.

Параметры **other** (Geometry) – Сравниваемая геометрия.

Тип результата bool

Результат Возвращает True, если геометрии равны.

static from_geojson(json, cs=None)

Возвращает геометрию из ее „GeoJSON“ представления.

Параметры

- **json** (str) – json представление в виде строки.
- **cs** (Optional[CoordSystem]) – Система координат.

Тип результата Geometry

static from_wkb(wkb, coordsystem=None)

Создает геометрический объект из строки формата WKB.

Параметры

- **wkb** (bytes) – Строка WKB
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии.

Список 124: Пример.

```
wkb = b'\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00$@\x00\x00\x00\x00\x00\x00
↪$@'
pnt = Geometry.from_wkb(wkb)
```

Тип результата Geometry

static from_wkt(wkt, coordsystem=None)

Создает геометрический объект из строки формата WKT.

Параметры

- **wkt** (str) – Строка WKT. Допустимо задание строки в формате с указанием SRID (EWKT). В данном случае система координат для создаваемой геометрии будет установлено исходя их этого значения.
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии. Если строка задана в виде EWKT и указано значение SRID, игнорируется.

Список 125: Пример.

```
polygon = Geometry.from_wkt('POLYGON ((10 10, 100 100, 100 10, 10 10))')
point = Geometry.from_wkt('SRID=4326;POINT (10 10)')
crs = CoordSystem.from_epsg(4326)
pline = Geometry.from_wkt('LINESTRING (30 10, 10 30, 40 40)', crs)
```

Тип результата Geometry

get_area(u=None)

Рассчитывает площадь, если объект площадной. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в квадратных метрах.

Список 126: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
poly = Polygon([(0,0), (0,2), (2, 2), (2,0)], cs=csMercatorKm)
print('Area Mercator km:', poly.get_area())
print('Area Mercator m:', poly.get_area(Unit.sq_m))
print('Perimeter Mercator km:', poly.get_perimeter())
print('Perimeter Mercator m:', poly.get_perimeter(Unit.m))
# В Широте/Долготе
poly.coordsystem = CoordSystem.from_prj("1, 104")
print('Area LL m:', poly.get_area())
print('Area LL km:', poly.get_area(Unit.sq_km))
print('Perimeter LL m:', poly.get_perimeter())
print('Perimeter LL km:', poly.get_perimeter(Unit.km))
'''
>>> Area Mercator km: 4.017946986632519
>>> Area Mercator m: 4017946.9866325194
>>> Perimeter Mercator km: 8.017972048421552
>>> Perimeter Mercator m: 8017.972048421552
```

(continues on next page)

(продолжение с предыдущей страницы)

```
>>> Area LL m: 49452172514.0342
>>> Area LL km: 49452.1725140342
>>> Perimeter LL m: 889423.5067063896
>>> Perimeter LL km: 889.4235067063896
'''
```

Параметры **u** (Optional[AreaUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_distance(other, u=None)

Производит расчет расстояния до объекта other. Результат возвращает в СК текущего объекта.

Параметры

- **other** (Geometry) – Анализируемый объект.
- **u** (Optional[LinearUnit]) – Единицы измерения, в которых требуется получить результат.

Список 127: Пример.

```
# Создадим геометрию без СК
first = Point(0, 0)
second = Point(10, 0)
print('В плане:', first.get_distance(second))
#Установим разные СК для точек
first.coordsystem = CoordSystem.from_prj("10, 104, 7, 0")
second.coordsystem = CoordSystem.from_prj("1, 104")
print('На сфере м:', first.get_distance(second))
print('На сфере км:', first.get_distance(second, Unit.km))
'''
>>> В плане: 10.0
>>> На сфере м: 1111948.7428468117
>>> На сфере км: 1111.9487428468117
'''
```

Тип результата float

get_length(u=None)

Рассчитывает длину геометрии. Метод применим только для линейных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Список 128: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
ls = Line((0,0), (10,0), csMercatorKm)
print('Length Mercator km:', ls.get_length())
print('Lenght Mercator m:', ls.get_length(Unit.m))
# В Широте/Долготе
```

(continues on next page)

(продолжение с предыдущей страницы)

```
csLL = CoordSystem.from_prj("1, 104")
ls = Line((0,0), (10,0), csLL)
print('Length LL m:', ls.get_length())
print('Length LL km:', ls.get_length(Unit.km))
'''
>>> Length Mercator km: 9.988805508567783
>>> Lenght Mercator m: 9988.805508567782
>>> Length LL m: 1111948.7428468117
>>> Length LL km: 1111.9487428468117
'''
```

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_perimeter(u=None)

Рассчитывает периметр геометрии. Метод применим только для площадных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Пример см. get_area()

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

intersection(other)

Возвращает область пересечения с другой геометрией.

Тип результата Geometry

intersects(other)

Возвращает True, если геометрии пересекаются.

Тип результата bool

property is_valid

Проверяет геометрию на валидность.

Тип результата bool

property is_valid_reason

Если геометрия неправильная, возвращает краткую аннотацию причины.

Тип результата str

property name

Возвращает наименование геометрического объекта.

Тип результата str

overlaps(other)

Возвращает True, если пересечение геометрий отличается от обеих геометрий.

Тип результата bool

static point_by_azimuth(point, azimuth, distance, cs=None)
Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.

См.также:

Обратная функция distance_by_points()

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, относительно которой производится расчет. Если задана СК, то в ней.
- **azimuth** (float) – Азимут в градусах, указывающий направление.
- **distance** (float) – Расстояние по азимуту. Задается в метрах.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Pnt

Результат Возвращает результирующую точку.

relate(other)
Проверяет отношения между объектами. Подробнее см. DE-9IM

Тип результата str

reproject(cs)
Перепроецирует геометрию в другую систему координат.

Параметры **cs** (CoordSystem) – СК, в которой требуется получить объект.

Тип результата Geometry

rotate(point, angle)
Поворот геометрии относительно заданной точки. Поворот производится против часовой стрелки.

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, вокруг которой необходимо произвести поворот.
- **angle** (float) – Угол поворота в градусах.

Тип результата Geometry

scale(kx, ky)
Масштабирует объект по заданным коэффициентам масштабирования. После выполнения операции центр результирующего объекта остается прежним.

Параметры

- **kx** (float) – Масштабирование по координате X.
- **ky** (float) – Масштабирование по координате Y.

Тип результата Geometry

shift(dx, dy)
Смещает объект на заданную величину. Значения задаются в текущей проекции.

Параметры

- **dx** (float) – Сдвиг по координате X.
- **dy** (float) – Сдвиг по координате Y.

Тип результата Geometry

symmetric_difference(other)
Возвращает логический XOR областей геометрий (объединение разниц).

Параметры **other** (Geometry) – Геометрия для анализа.

Тип результата Geometry

to_geojson()
Преобразует геометрию в формат „GeoJSON“

Тип результата str

to_linestring()
Пробует геометрию преобразовать в линейный объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

to_polygon()
Пробует геометрию преобразовать в площадной объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

touches(other)
Возвращает True, если геометрии соприкасаются.

Тип результата bool

property type
Возвращает тип геометрического элемента.

Таблица 76: Возможные значения

Значение	Наименование
Unknown	Не определен
Point	Точка
Line	Линия
LineString	Полилиния
Polygon	Полигон
MultiPoint	Коллекция точек
MultiLineString	Коллекция полилиний
MultiPolygon	Коллекция полигонов
GeometryCollection	Смешанная коллекция
Arc	Дуга
Ellipse	Эллипс
Rectangle	Прямоугольник
RoundedRectangle	Скругленный прямоугольник
Text	Текст

Список 129: Пример.

```
point = Point(10,10)
if point.type == GeometryType.Point:
    print('Это точка')
```

Тип результата GeometryType

union(other)
Возвращает результат объединения двух геометрий.

Тип результата Geometry

within(other)
Возвращает True, если геометрия находится полностью внутри геометрии other.

Тип результата bool

property wkb
Возвращает WKB строку для геометрии.

Тип результата bytes

property wkt
Возвращает WKT строку для геометрии.

Тип результата str

property xmax
Максимальное значение X.

Тип результата float

property xmin
Минимальное значение X.

Тип результата float

property ymax
Максимальное значение Y.

Тип результата float

property ymin
Минимальное значение Y.

Тип результата float

RoundRectangle - Скругленный прямоугольник

class axipy.mi.**RoundRectangle**(rect, xRad, yRad, cs=None)

Базовые классы: axipy.mi.Rectangle

Геометрический объект типа скругленный прямоугольник.

Параметры

- **rect** (Union[Rect, list]) – Прямоугольник класса Rect или как list.
- **xRad** (float) – Скругление по X.

- **yRad** (float) – Скругление по Y.
- **cs** (Optional[CoordSystem]) – Система Координат, в которой создается геометрия.

Список 130: Пример.

```
r1 = RoundRectangle(Rect(0,0,22,33), 0.1, 0.1)
r2 = RoundRectangle([0,0,22,33], 0.1, 0.1)
```

Methods:

affine_transform(trans)	Трансформирует объект исходя из заданных параметров трансформации.
almost_equals(other, tolerance)	Производит примерное сравнения с другой геометрией в пределах заданной точности.
boundary()	Возвращает границы геометрии в виде полилинии.
buffer(distance[, resolution, capStyle, ...])	Производит построение буфера.
centroid()	Возвращает центроид геометрии.
clone()	Создает копию объекта.
contains(other)	Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.
convex_hull()	Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.
covers(other)	Возвращает True, если геометрия охватывает геометрию other.
crosses(other)	Возвращает True, если при пересечении геометрий объекты частично пересекаются.
difference(other)	Возвращает область первой геометрии, которая не пересечена второй геометрией.
disjoint(other)	Возвращает True, если геометрии не пересекаются и не соприкасаются.
distance_by_points(start, end[, cs])	Производит расчет расстояния между двумя точками и азимут от первой до второй точки.
envelope()	Возвращает полигон, описывающий заданную геометрию.
equals(other)	Производит сравнение с другой геометрией.
from_geojson(json[, cs])	Возвращает геометрию из ее „GeoJSON“ представления.
from_wkb(wkb[, coordsystem])	Создает геометрический объект из строки формата WKB.
from_wkt(wkt[, coordsystem])	Создает геометрический объект из строки формата WKT.

continues on next page

Таблица 77 – продолжение с предыдущей страницы

get_area([u])	Рассчитывает площадь, если объект площадной.
get_distance(other[, u])	Производит расчет расстояния до объекта other.
get_length([u])	Рассчитывает длину геометрии.
get_perimeter([u])	Рассчитывает периметр геометрии.
intersection(other)	Возвращает область пересечения с другой геометрией.
intersects(other)	Возвращает True, если геометрии пересекаются.
overlaps(other)	Возвращает True, если пересечение геометрий отличается от обеих геометрий.
point_by_azimuth(point, azimuth, distance[, cs])	Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.
relate(other)	Проверяет отношения между объектами.
reproject(cs)	Перепроецирует геометрию в другую систему координат.
rotate(point, angle)	Поворот геометрии относительно заданной точки.
scale(kx, ky)	Масштабирует объект по заданным коэффициентам масштабирования.
shift(dx, dy)	Смещает объект на заданную величину.
symmetric_difference(other)	Возвращает логический XOR областей геометрий (объединение разниц).
to_geojson()	Преобразует геометрию в формат „GeoJSON“
to_linestring()	Пробует геометрию преобразовать в линейный объект.
to_polygon()	Пробует геометрию преобразовать в площадной объект.
touches(other)	Возвращает True, если геометрии соприкасаются.
union(other)	Возвращает результат объединения двух геометрий.
within(other)	Возвращает True, если геометрия находится полностью внутри геометрии other.

Attributes:

bounds	Возвращает минимальный ограничивающий прямоугольник.
coordsystem	Система Координат (СК) геометрии.
is_valid	Проверяет геометрию на валидность.

continues on next page

Таблица 78 – продолжение с предыдущей страницы

is_valid_reason	Если геометрия неправильная, возвращает краткую аннотацию причины.
name	Возвращает наименование геометрического объекта.
type	Возвращает тип геометрического элемента.
wkb	Возвращает WKB строку для геометрии.
wkt	Возвращает WKT строку для геометрии.
xRadius	Скругление углов по координате X.
xmax	Максимальное значение X.
xmin	Минимальное значение X.
yRadius	Скругление углов по координате Y.
ymax	Максимальное значение Y.
ymin	Минимальное значение Y.

affine_transform(trans)
Трансформирует объект исходя из заданных параметров трансформации.

См.также:

Для простых операций типа сдвиг, масштабирование и поворот, рекомендуется использовать shift(), scale(), rotate() соответственно.

Параметры **trans** (QTransform) – Матрица трансформации.

Тип результата Geometry

almost_equals(other, tolerance)
Производит примерное сравнения с другой геометрией в пределах заданной точности.

Параметры

- **other** (Geometry) – Сравниваемый объект.
- **tolerance** (float) – Точность сравнения.

Тип результата bool

Результат Возвращает True, если геометрии равны в пределах заданного отклонения.

boundary()
Возвращает границы геометрии в виде полилинии.

Тип результата Geometry

property bounds
Возвращает минимальный ограничивающий прямоугольник.

Тип результата Rect

buffer(distance, resolution=16, capStyle=1, joinStyle=1, mitreLimit=5.0)
Производит построение буфера.

Параметры

- **distance** (float) – Ширина буфера.
- **resolution** (int) – Количество сегментов на квадрант.
- **capStyle** (int) – Стиль окончания.
- **joinStyle** (int) – Стиль соединения.
- **mitreLimit** (float) – Предел среза.

Тип результата Geometry

centroid()
Возвращает центроид геометрии.

Тип результата Geometry

clone()
Создает копию объекта.

Тип результата Geometry

contains(other)
Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.

Тип результата bool

convex_hull()
Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.

Тип результата Geometry

property coordsystem
Система Координат (СК) геометрии.

Тип результата CoordSystem

covers(other)
Возвращает True, если геометрия охватывает геометрию other.

Тип результата bool

crosses(other)
Возвращает True, если при пересечении геометрий объекты частично пересекаются.

Тип результата bool

difference(other)
Возвращает область первой геометрии, которая не пересечена второй геометрией.

Тип результата Geometry

disjoint(other)
Возвращает True, если геометрии не пересекаются и не соприкасаются.

Тип результата bool

static distance_by_points(start, end, cs=None)
Производит расчет расстояния между двумя точками и азимут от первой до второй точки.

См.также:

Обратная функция point_by_azimuth()

Параметры

- **start** (Union[Pnt, Tuple[float, float]]) – Начальная точка. Если задана СК, то координаты в ней.
- **end** (Union[Pnt, Tuple[float, float]]) – Конечная точка. Если задана СК, то координаты в ней.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Tuple

Результат Возвращается пара значений (расстояние в метрах, азимут в градусах)

envelope()
Возвращает полигон, описывающий заданную геометрию.

Тип результата Geometry

equals(other)
Производит сравнение с другой геометрией.

Параметры **other** (Geometry) – Сравниваемая геометрия.

Тип результата bool

Результат Возвращает True, если геометрии равны.

static from_geojson(json, cs=None)
Возвращает геометрию из ее „GeoJSON“ представления.

Параметры

- **json** (str) – json представление в виде строки.
- **cs** (Optional[CoordSystem]) – Система координат.

Тип результата Geometry

static from_wkb(wkb, coordsystem=None)
Создает геометрический объект из строки формата WKB.

Параметры

- **wkb** (bytes) – Строка WKB
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии.

Список 131: Пример.

```
wkb = b'\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00$@\x00\x00\x00\x00\x00\x00
 ↪$@'
pnt = Geometry.from_wkb(wkb)
```

Тип результата Geometry

static from_wkt(wkt, coordsystem=None)
Создает геометрический объект из строки формата WKT.

Параметры

- **wkt** (str) – Строка WKT. Допустимо задание строки в формате с указанием SRID (EWKT). В данном случае система координат для создаваемой геометрии будет установлено исходя их этого значения.
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии. Если строка задана в виде EWKT и указано значение SRID, игнорируется.

Список 132: Пример.

```
polygon = Geometry.from_wkt('POLYGON ((10 10, 100 100, 100 10, 10 10))')
point = Geometry.from_wkt('SRID=4326;POINT (10 10)')
crs = CoordSystem.from_epsg(4326)
pline = Geometry.from_wkt('LINESTRING (30 10, 10 30, 40 40)', crs)
```

Тип результата Geometry

get_area(u=None)

Рассчитывает площадь, если объект площадной. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в квадратных метрах.

Список 133: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
poly = Polygon([(0,0), (0,2), (2, 2), (2,0)], cs=csMercatorKm)
print('Area Mercator km:', poly.get_area())
print('Area Mercator m:', poly.get_area(Unit.sq_m))
print('Perimeter Mercator km:', poly.get_perimeter())
print('Perimeter Mercator m:', poly.get_perimeter(Unit.m))
# В Широте/Долготе
poly.coordsystem = CoordSystem.from_prj("1, 104")
print('Area LL m:', poly.get_area())
print('Area LL km:', poly.get_area(Unit.sq_km))
print('Perimeter LL m:', poly.get_perimeter())
print('Perimeter LL km:', poly.get_perimeter(Unit.km))
'''
>>> Area Mercator km: 4.017946986632519
>>> Area Mercator m: 4017946.9866325194
>>> Perimeter Mercator km: 8.017972048421552
>>> Perimeter Mercator m: 8017.972048421552
>>> Area LL m: 49452172514.0342
>>> Area LL km: 49452.1725140342
>>> Perimeter LL m: 889423.5067063896
>>> Perimeter LL km: 889.4235067063896
'''
```

Параметры u (Optional[AreaUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_distance(other, u=None)

Производит расчет расстояния до объекта other. Результат возвращает в СК текущего объекта.

Параметры

- **other** (Geometry) – Анализируемый объект.
- **u** (Optional[LinearUnit]) – Единицы измерения, в которых требуется получить результат.

Список 134: Пример.

```
# Создадим геометрию без СК
first = Point(0, 0)
second = Point(10, 0)
print('В плане:', first.get_distance(second))
#Установим разные СК для точек
first.coordsystem = CoordSystem.from_prj("10, 104, 7, 0")
second.coordsystem = CoordSystem.from_prj("1, 104")
print('На сфере м:', first.get_distance(second))
print('На сфере км:', first.get_distance(second, Unit.km))
'''
>>> В плане: 10.0
>>> На сфере м: 1111948.7428468117
>>> На сфере км: 1111.9487428468117
'''
```

Тип результата float

get_length(u=None)

Рассчитывает длину геометрии. Метод применим только для линейных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Список 135: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
ls = Line((0,0), (10,0), csMercatorKm)
print('Length Mercator km:', ls.get_length())
print('Lenght Mercator m:', ls.get_length(Unit.m))
# В Широте/Долготе
csLL = CoordSystem.from_prj("1, 104")
ls = Line((0,0), (10,0), csLL)
print('Length LL m:', ls.get_length())
print('Length LL km:', ls.get_length(Unit.km))
'''
>>> Length Mercator km: 9.988805508567783
>>> Lenght Mercator m: 9988.805508567782
>>> Length LL m: 1111948.7428468117
>>> Length LL km: 1111.9487428468117
'''
```

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_perimeter(u=None)
Рассчитывает периметр геометрии. Метод применим только для площадных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Пример см. get_area()

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

intersection(other)
Возвращает область пересечения с другой геометрией.

Тип результата Geometry

intersects(other)
Возвращает True, если геометрии пересекаются.

Тип результата bool

property is_valid
Проверяет геометрию на валидность.

Тип результата bool

property is_valid_reason
Если геометрия неправильная, возвращает краткую аннотацию причины.

Тип результата str

property name
Возвращает наименование геометрического объекта.

Тип результата str

overlaps(other)
Возвращает True, если пересечение геометрий отличается от обеих геометрий.

Тип результата bool

static point_by_azimuth(point, azimuth, distance, cs=None)
Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.

См.также:

Обратная функция distance_by_points()

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, относительно которой производится расчет. Если задана СК, то в ней.
- **azimuth** (float) – Азимут в градусах, указывающий направление.
- **distance** (float) – Расстояние по азимуту. Задается в метрах.

- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Pnt

Результат Возвращает результирующую точку.

relate(other)
Проверяет отношения между объектами. Подробнее см. DE-9IM

Тип результата str

reproject(cs)
Перепроецирует геометрию в другую систему координат.

Параметры **cs** (CoordSystem) – СК, в которой требуется получить объект.

Тип результата Geometry

rotate(point, angle)
Поворот геометрии относительно заданной точки. Поворот производится против часовой стрелки.

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, вокруг которой необходимо произвести поворот.
- **angle** (float) – Угол поворота в градусах.

Тип результата Geometry

scale(kx, ky)
Масштабирует объект по заданным коэффициентам масштабирования. После выполнения операции центр результирующего объекта остается прежним.

Параметры

- **kx** (float) – Масштабирование по координате X.
- **ky** (float) – Масштабирование по координате Y.

Тип результата Geometry

shift(dx, dy)
Смещает объект на заданную величину. Значения задаются в текущей проекции.

Параметры

- **dx** (float) – Сдвиг по координате X.
- **dy** (float) – Сдвиг по координате Y.

Тип результата Geometry

symmetric_difference(other)
Возвращает логический XOR областей геометрий (объединение разниц).

Параметры **other** (Geometry) – Геометрия для анализа.

Тип результата Geometry

to_geojson()
Преобразует геометрию в формат „GeoJSON“

Тип результата str

to_linestring()
Пробует геометрию преобразовать в линейный объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

to_polygon()
Пробует геометрию преобразовать в площадной объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

touches(other)
Возвращает True, если геометрии соприкасаются.

Тип результата bool

property type
Возвращает тип геометрического элемента.

Таблица 79: Возможные значения

Значение	Наименование
Unknown	Не определен
Point	Точка
Line	Линия
LineString	Полилиния
Polygon	Полигон
MultiPoint	Коллекция точек
MultiLineString	Коллекция полилиний
MultiPolygon	Коллекция полигонов
GeometryCollection	Смешанная коллекция
Arc	Дуга
Ellipse	Эллипс
Rectangle	Прямоугольник
RoundedRectangle	Скругленный прямоугольник
Text	Текст

Список 136: Пример.

```
point = Point(10,10)
if point.type == GeometryType.Point:
    print('Это точка')
```

Тип результата GeometryType

union(other)
Возвращает результат объединения двух геометрий.

Тип результата Geometry

within(other)
Возвращает True, если геометрия находится полностью внутри геометрии other.

Тип результата bool

property wkb
Возвращает WKB строку для геометрии.

Тип результата bytes

property wkt
Возвращает WKT строку для геометрии.

Тип результата str

property xRadius
Скругление углов по координате X.

Тип результата float

property xmax
Максимальное значение X.

Тип результата float

property xmin
Минимальное значение X.

Тип результата float

property yRadius
Скругление углов по координате Y.

Тип результата float

property ymax
Максимальное значение Y.

Тип результата float

property ymin
Минимальное значение Y.

Тип результата float

Ellipse - Эллипс

class axipy.mi.**Ellipse**(rect, cs=None)

Базовые классы: axipy.da.Geometry

Геометрический объект типа эллипс.

Параметры

- **rect** (Union[Rect, list]) – Прямоугольник класса Rect или как list.
- **cs** (Optional[CoordSystem]) – Система Координат, в которой создается геометрия.

Список 137: Пример.

```
e1 = Ellipse(Rect(0,0,22,33))
e2 = Ellipse([0,0,22,33])
e1.center = (10,10) # Переопределим центр
e1.majorSemiAxis = 10 # Задание большой полуоси
e1.minorSemiAxis = 5 # Задание малой полуоси
```

Methods:

affine_transform(trans)	Трансформирует объект исходя из заданных параметров трансформации.
almost_equals(other, tolerance)	Производит примерное сравнения с другой геометрией в пределах заданной точности.
boundary()	Возвращает границы геометрии в виде полилинии.
buffer(distance[, resolution, capStyle, ...])	Производит построение буфера.
centroid()	Возвращает центроид геометрии.
clone()	Создает копию объекта.
contains(other)	Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.
convex_hull()	Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.
covers(other)	Возвращает True, если геометрия охватывает геометрию other.
crosses(other)	Возвращает True, если при пересечении геометрий объекты частично пересекаются.
difference(other)	Возвращает область первой геометрии, которая не пересечена второй геометрией.
disjoint(other)	Возвращает True, если геометрии не пересекаются и не соприкасаются.
distance_by_points(start, end[, cs])	Производит расчет расстояния между двумя точками и азимут от первой до второй точки.
envelope()	Возвращает полигон, описывающий заданную геометрию.
equals(other)	Производит сравнение с другой геометрией.
from_geojson(json[, cs])	Возвращает геометрию из ее „GeoJSON“ представления.
from_wkb(wkb[, coordsystem])	Создает геометрический объект из строки формата WKB.
from_wkt(wkt[, coordsystem])	Создает геометрический объект из строки формата WKT.
get_area([u])	Рассчитывает площадь, если объект площадной.

continues on next page

Таблица 80 – продолжение с предыдущей страницы

get_distance(other[, u])	Производит расчет расстояния до объекта other.
get_length([u])	Рассчитывает длину геометрии.
get_perimeter([u])	Рассчитывает периметр геометрии.
intersection(other)	Возвращает область пересечения с другой геометрией.
intersects(other)	Возвращает True, если геометрии пересекаются.
overlaps(other)	Возвращает True, если пересечение геометрий отличается от обеих геометрий.
point_by_azimuth(point, azimuth, distance[, cs])	Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.
relate(other)	Проверяет отношения между объектами.
reproject(cs)	Перепроецирует геометрию в другую систему координат.
rotate(point, angle)	Поворот геометрии относительно заданной точки.
scale(kx, ky)	Масштабирует объект по заданным коэффициентам масштабирования.
shift(dx, dy)	Смещает объект на заданную величину.
symmetric_difference(other)	Возвращает логический XOR областей геометрий (объединение разниц).
to_geojson()	Преобразует геометрию в формат „GeoJSON“
to_linestring()	Пробует геометрию преобразовать в линейный объект.
to_polygon()	Пробует геометрию преобразовать в площадной объект.
touches(other)	Возвращает True, если геометрии соприкасаются.
union(other)	Возвращает результат объединения двух геометрий.
within(other)	Возвращает True, если геометрия находится полностью внутри геометрии other.

Attributes:

bounds	Возвращает минимальный ограничивающий прямоугольник.
center	Центр эллипса.
coordsystem	Система Координат (СК) геометрии.
is_valid	Проверяет геометрию на валидность.
is_valid_reason	Если геометрия неправильная, возвращает краткую аннотацию причины.

continues on next page

Таблица 81 – продолжение с предыдущей страницы

majorSemiAxis	Радиус большой полуоси эллипса.
minorSemiAxis	Радиус малой полуоси эллипса.
name	Возвращает наименование геометрического объекта.
type	Возвращает тип геометрического элемента.
wkb	Возвращает WKB строку для геометрии.
wkt	Возвращает WKT строку для геометрии.

affine_transform(trans)
Трансформирует объект исходя из заданных параметров трансформации.

См.также:

Для простых операций типа сдвиг, масштабирование и поворот, рекомендуется использовать shift(), scale(), rotate() соответственно.

Параметры **trans** (QTransform) – Матрица трансформации.

Тип результата Geometry

almost_equals(other, tolerance)
Производит примерное сравнения с другой геометрией в пределах заданной точности.

Параметры

- **other** (Geometry) – Сравниваемый объект.
- **tolerance** (float) – Точность сравнения.

Тип результата bool

Результат Возвращает True, если геометрии равны в пределах заданного отклонения.

boundary()
Возвращает границы геометрии в виде полилинии.

Тип результата Geometry

property bounds
Возвращает минимальный ограничивающий прямоугольник.

Тип результата Rect

buffer(distance, resolution=16, capStyle=1, joinStyle=1, mitreLimit=5.0)
Производит построение буфера.

Параметры

- **distance** (float) – Ширина буфера.
- **resolution** (int) – Количество сегментов на квадрант.
- **capStyle** (int) – Стиль окончания.
- **joinStyle** (int) – Стиль соединения.
- **mitreLimit** (float) – Предел среза.

Тип результата Geometry

property center
Центр эллипса.

Тип результата Pnt

centroid()
Возвращает центроид геометрии.

Тип результата Geometry

clone()
Создает копию объекта.

Тип результата Geometry

contains(other)
Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.

Тип результата bool

convex_hull()
Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.

Тип результата Geometry

property coordsystem
Система Координат (СК) геометрии.

Тип результата CoordSystem

covers(other)
Возвращает True, если геометрия охватывает геометрию other.

Тип результата bool

crosses(other)
Возвращает True, если при пересечении геометрий объекты частично пересекаются.

Тип результата bool

difference(other)
Возвращает область первой геометрии, которая не пересечена второй геометрией.

Тип результата Geometry

disjoint(other)
Возвращает True, если геометрии не пересекаются и не соприкасаются.

Тип результата bool

static distance_by_points(start, end, cs=None)
Производит расчет расстояния между двумя точками и азимут от первой до второй точки.

См.также:

Обратная функция point_by_azimuth()

Параметры

- **start** (Union[Pnt, Tuple[float, float]]) – Начальная точка. Если задана СК, то координаты в ней.
- **end** (Union[Pnt, Tuple[float, float]]) – Конечная точка. Если задана СК, то координаты в ней.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Tuple

Результат Возвращается пара значений (расстояние в метрах, азимут в градусах)

envelope()
Возвращает полигон, описывающий заданную геометрию.

Тип результата Geometry

equals(other)
Производит сравнение с другой геометрией.

Параметры **other** (Geometry) – Сравниваемая геометрия.

Тип результата bool

Результат Возвращает True, если геометрии равны.

static from_geojson(json, cs=None)
Возвращает геометрию из ее „GeoJSON“ представления.

Параметры

- **json** (str) – json представление в виде строки.
- **cs** (Optional[CoordSystem]) – Система координат.

Тип результата Geometry

static from_wkb(wkb, coordsystem=None)
Создает геометрический объект из строки формата WKB.

Параметры

- **wkb** (bytes) – Строка WKB
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии.

Список 138: Пример.

```
wkb = b'\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00$@\x00\x00\x00\x00\x00\x00
↪$@'
pnt = Geometry.from_wkb(wkb)
```

Тип результата Geometry

static from_wkt(wkt, coordsystem=None)
Создает геометрический объект из строки формата WKT.

Параметры

- **wkt** (str) – Строка WKT. Допустимо задание строки в формате с указанием SRID (EWKT). В данном случае система координат для создаваемой геометрии будет установлено исходя их этого значения.
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии. Если строка задана в виде EWKT и указано значение SRID, игнорируется.

Список 139: Пример.

```
polygon = Geometry.from_wkt('POLYGON ((10 10, 100 100, 100 10, 10 10))')
point = Geometry.from_wkt('SRID=4326;POINT (10 10)')
crs = CoordSystem.from_epsg(4326)
pline = Geometry.from_wkt('LINESTRING (30 10, 10 30, 40 40)', crs)
```

Тип результата Geometry

get_area(u=None)

Рассчитывает площадь, если объект площадной. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в квадратных метрах.

Список 140: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
poly = Polygon([(0,0), (0,2), (2, 2), (2,0)], cs=csMercatorKm)
print('Area Mercator km:', poly.get_area())
print('Area Mercator m:', poly.get_area(Unit.sq_m))
print('Perimeter Mercator km:', poly.get_perimeter())
print('Perimeter Mercator m:', poly.get_perimeter(Unit.m))
# В Широте/Долготе
poly.coordsystem = CoordSystem.from_prj("1, 104")
print('Area LL m:', poly.get_area())
print('Area LL km:', poly.get_area(Unit.sq_km))
print('Perimeter LL m:', poly.get_perimeter())
print('Perimeter LL km:', poly.get_perimeter(Unit.km))
'''
>>> Area Mercator km: 4.017946986632519
>>> Area Mercator m: 4017946.9866325194
>>> Perimeter Mercator km: 8.017972048421552
>>> Perimeter Mercator m: 8017.972048421552
>>> Area LL m: 49452172514.0342
>>> Area LL km: 49452.1725140342
>>> Perimeter LL m: 889423.5067063896
>>> Perimeter LL km: 889.4235067063896
'''
```

Параметры u (Optional[AreaUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_distance(other, u=None)
Производит расчет расстояния до объекта other. Результат возвращает в СК текущего объекта.

Параметры

- **other** (Geometry) – Анализируемый объект.
- **u** (Optional[LinearUnit]) – Единицы измерения, в которых требуется получить результат.

Список 141: Пример.

```
# Создадим геометрию без СК
first = Point(0, 0)
second = Point(10, 0)
print('В плане:', first.get_distance(second))
#Установим разные СК для точек
first.coordsystem = CoordSystem.from_prj("10, 104, 7, 0")
second.coordsystem = CoordSystem.from_prj("1, 104")
print('На сфере м:', first.get_distance(second))
print('На сфере км:', first.get_distance(second, Unit.km))
'''
>>> В плане: 10.0
>>> На сфере м: 1111948.7428468117
>>> На сфере км: 1111.9487428468117
'''
```

Тип результата float

get_length(u=None)
Рассчитывает длину геометрии. Метод применим только для линейных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Список 142: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
ls = Line((0,0), (10,0), csMercatorKm)
print('Length Mercator km:', ls.get_length())
print('Lenght Mercator m:', ls.get_length(Unit.m))
# В Широте/Долготе
csLL = CoordSystem.from_prj("1, 104")
ls = Line((0,0), (10,0), csLL)
print('Length LL m:', ls.get_length())
print('Length LL km:', ls.get_length(Unit.km))
'''
>>> Length Mercator km: 9.988805508567783
>>> Lenght Mercator m: 9988.805508567782
>>> Length LL m: 1111948.7428468117
>>> Length LL km: 1111.9487428468117
'''
```

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_perimeter(u=None)
Рассчитывает периметр геометрии. Метод применим только для площадных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Пример см. get_area()

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

intersection(other)
Возвращает область пересечения с другой геометрией.

Тип результата Geometry

intersects(other)
Возвращает True, если геометрии пересекаются.

Тип результата bool

property is_valid
Проверяет геометрию на валидность.

Тип результата bool

property is_valid_reason
Если геометрия неправильная, возвращает краткую аннотацию причины.

Тип результата str

property majorSemiAxis
Радиус большой полуоси эллипса.

Тип результата float

property minorSemiAxis
Радиус малой полуоси эллипса.

Тип результата float

property name
Возвращает наименование геометрического объекта.

Тип результата str

overlaps(other)
Возвращает True, если пересечение геометрий отличается от обеих геометрий.

Тип результата bool

static point_by_azimuth(point, azimuth, distance, cs=None)
Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.

См.также:

Обратная функция distance_by_points()

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, относительно которой производится расчет. Если задана СК, то в ней.
- **azimuth** (float) – Азимут в градусах, указывающий направление.
- **distance** (float) – Расстояние по азимуту. Задается в метрах.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Pnt

Результат Возвращает результирующую точку.

relate(other)
Проверяет отношения между объектами. Подробнее см. DE-9IM

Тип результата str

reproject(cs)
Перепроецирует геометрию в другую систему координат.

Параметры **cs** (CoordSystem) – СК, в которой требуется получить объект.

Тип результата Geometry

rotate(point, angle)
Поворот геометрии относительно заданной точки. Поворот производится против часовой стрелки.

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, вокруг которой необходимо произвести поворот.
- **angle** (float) – Угол поворота в градусах.

Тип результата Geometry

scale(kx, ky)
Масштабирует объект по заданным коэффициентам масштабирования. После выполнения операции центр результирующего объекта остается прежним.

Параметры

- **kx** (float) – Масштабирование по координате X.
- **ky** (float) – Масштабирование по координате Y.

Тип результата Geometry

shift(dx, dy)
Смещает объект на заданную величину. Значения задаются в текущей проекции.

Параметры

- **dx** (float) – Сдвиг по координате X.
- **dy** (float) – Сдвиг по координате Y.

Тип результата Geometry

symmetric_difference(other)
Возвращает логический XOR областей геометрий (объединение разниц).

Параметры other (Geometry) – Геометрия для анализа.

Тип результата Geometry

to_geojson()
Преобразует геометрию в формат „GeoJSON“

Тип результата str

to_linestring()
Пробует геометрию преобразовать в линейный объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

to_polygon()
Пробует геометрию преобразовать в площадной объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

touches(other)
Возвращает True, если геометрии соприкасаются.

Тип результата bool

property type
Возвращает тип геометрического элемента.

Таблица 82: Возможные значения

Значение	Наименование
Unknown	Не определен
Point	Точка
Line	Линия
LineString	Полилиния
Polygon	Полигон
MultiPoint	Коллекция точек
MultiLineString	Коллекция полилиний
MultiPolygon	Коллекция полигонов
GeometryCollection	Смешанная коллекция
Arc	Дуга
Ellipse	Эллипс
Rectangle	Прямоугольник
RoundedRectangle	Скругленный прямоугольник
Text	Текст

Список 143: Пример.

```
point = Point(10,10)
if point.type == GeometryType.Point:
    print('Это точка')
```

Тип результата GeometryType

union(other)
Возвращает результат объединения двух геометрий.

Тип результата Geometry

within(other)
Возвращает True, если геометрия находится полностью внутри геометрии other.

Тип результата bool

property wkb
Возвращает WKB строку для геометрии.

Тип результата bytes

property wkt
Возвращает WKT строку для геометрии.

Тип результата str

Arc - Дуга

class axipy.mi.**Arc**(rect, startAngle, endAngle, cs=None)

Базовые классы: axipy.da.Geometry

Геометрический объект типа дуга.

Параметры

- **rect** (Union[Rect, list]) – Прямоугольник класса Rect или как list.
- **startAngle** (float) – Начальный угол дуги.
- **endAngle** (float) – Конечный угол дуги.
- **cs** (Optional[CoordSystem]) – Система Координат, в которой создается геометрия.

Список 144: Пример.

```
a1 = Arc(Rect(0,0,22,33), 0, 90)
a2 = Arc([0,0,22,33], 0, 90)
```

Methods:

affine_transform(trans)	Трансформирует объект исходя из заданных параметров трансформации.
almost_equals(other, tolerance)	Производит примерное сравнения с другой геометрией в пределах заданной точности.
boundary()	Возвращает границы геометрии в виде полилинии.
buffer(distance[, resolution, capStyle, ...])	Производит построение буфера.
centroid()	Возвращает центроид геометрии.
clone()	Создает копию объекта.

continues on next page

Таблица 83 – продолжение с предыдущей страницы

contains(other)	Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.
convex_hull()	Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.
covers(other)	Возвращает True, если геометрия охватывает геометрию other.
crosses(other)	Возвращает True, если при пересечении геометрий объекты частично пересекаются.
difference(other)	Возвращает область первой геометрии, которая не пересечена второй геометрией.
disjoint(other)	Возвращает True, если геометрии не пересекаются и не соприкасаются.
distance_by_points(start, end[, cs])	Производит расчет расстояния между двумя точками и азимут от первой до второй точки.
envelope()	Возвращает полигон, описывающий заданную геометрию.
equals(other)	Производит сравнение с другой геометрией.
from_geojson(json[, cs])	Возвращает геометрию из ее „GeoJSON“ представления.
from_wkb(wkb[, coordsystem])	Создает геометрический объект из строки формата WKB.
from_wkt(wkt[, coordsystem])	Создает геометрический объект из строки формата WKT.
get_area([u])	Рассчитывает площадь, если объект площадной.
get_distance(other[, u])	Производит расчет расстояния до объекта other.
get_length([u])	Рассчитывает длину геометрии.
get_perimeter([u])	Рассчитывает периметр геометрии.
intersection(other)	Возвращает область пересечения с другой геометрией.
intersects(other)	Возвращает True, если геометрии пересекаются.
overlaps(other)	Возвращает True, если пересечение геометрий отличается от обеих геометрий.
point_by_azimuth(point, azimuth, distance[, cs])	Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.
relate(other)	Проверяет отношения между объектами.
reproject(cs)	Перепроецирует геометрию в другую систему координат.

continues on next page

Таблица 83 – продолжение с предыдущей страницы

rotate(point, angle)	Поворот геометрии относительно заданной точки.
scale(kx, ky)	Масштабирует объект по заданным коэффициентам масштабирования.
shift(dx, dy)	Смещает объект на заданную величину.
symmetric_difference(other)	Возвращает логический XOR областей геометрий (объединение разниц).
to_geojson()	Преобразует геометрию в формат „GeoJSON“
to_linestring()	Пробует геометрию преобразовать в линейный объект.
to_polygon()	Пробует геометрию преобразовать в площадной объект.
touches(other)	Возвращает True, если геометрии соприкасаются.
union(other)	Возвращает результат объединения двух геометрий.
within(other)	Возвращает True, если геометрия находится полностью внутри геометрии other.

Attributes:

bounds	Возвращает минимальный ограничивающий прямоугольник.
center	Центр дуги.
coordsystem	Система Координат (СК) геометрии.
endAngle	Конечный угол дуги.
is_valid	Проверяет геометрию на валидность.
is_valid_reason	Если геометрия неправильная, возвращает краткую аннотацию причины.
name	Возвращает наименование геометрического объекта.
startAngle	Начальный угол дуги.
type	Возвращает тип геометрического элемента.
wkb	Возвращает WKB строку для геометрии.
wkt	Возвращает WKT строку для геометрии.
xRadius	Радиус большой полуоси прямоугольника, в который вписана дуга.
yRadius	Радиус малой полуоси прямоугольника, в который вписана дуга.

affine_transform(trans)

Трансформирует объект исходя из заданных параметров трансформации.

См.также:

Для простых операций типа сдвиг, масштабирование и поворот, рекомендуется использовать `shift()`, `scale()`, `rotate()` соответственно.

Параметры **trans** (`QTransform`) – Матрица трансформации.

Тип результата `Geometry`

almost_equals(other, tolerance)
Производит примерное сравнения с другой геометрией в пределах заданной точности.

Параметры

- **other** (`Geometry`) – Сравниваемый объект.
- **tolerance** (`float`) – Точность сравнения.

Тип результата `bool`

Результат Возвращает True, если геометрии равны в пределах заданного отклонения.

boundary()
Возвращает границы геометрии в виде полилинии.

Тип результата `Geometry`

property bounds
Возвращает минимальный ограничивающий прямоугольник.

Тип результата `Rect`

buffer(distance, resolution=16, capStyle=1, joinStyle=1, mitreLimit=5.0)
Производит построение буфера.

Параметры

- **distance** (`float`) – Ширина буфера.
- **resolution** (`int`) – Количество сегментов на квадрант.
- **capStyle** (`int`) – Стиль окончания.
- **joinStyle** (`int`) – Стиль соединения.
- **mitreLimit** (`float`) – Предел среза.

Тип результата `Geometry`

property center
Центр дуги.

Тип результата `Pnt`

centroid()
Возвращает центроид геометрии.

Тип результата `Geometry`

clone()
Создает копию объекта.

Тип результата `Geometry`

contains(other)
Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.

Тип результата bool

convex_hull()
Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.

Тип результата Geometry

property coordsystem
Система Координат (СК) геометрии.

Тип результата CoordSystem

covers(other)
Возвращает True, если геометрия охватывает геометрию other.

Тип результата bool

crosses(other)
Возвращает True, если при пересечении геометрий объекты частично пересекаются.

Тип результата bool

difference(other)
Возвращает область первой геометрии, которая не пересечена второй геометрией.

Тип результата Geometry

disjoint(other)
Возвращает True, если геометрии не пересекаются и не соприкасаются.

Тип результата bool

static distance_by_points(start, end, cs=None)
Производит расчет расстояния между двумя точками и азимут от первой до второй точки.

См.также:

Обратная функция point_by_azimuth()

Параметры

- **start** (Union[Pnt, Tuple[float, float]]) – Начальная точка. Если задана СК, то координаты в ней.
- **end** (Union[Pnt, Tuple[float, float]]) – Конечная точка. Если задана СК, то координаты в ней.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Tuple

Результат Возвращается пара значений (расстояние в метрах, азимут в градусах)

property endAngle
Конечный угол дуги.

Тип результата float

envelope()
Возвращает полигон, описывающий заданную геометрию.

Тип результата Geometry

equals(other)
Производит сравнение с другой геометрией.

Параметры **other** (Geometry) – Сравниваемая геометрия.

Тип результата bool

Результат Возвращает True, если геометрии равны.

static from_geojson(json, cs=None)
Возвращает геометрию из ее „GeoJSON“ представления.

Параметры

- **json** (str) – json представление в виде строки.
- **cs** (Optional[CoordSystem]) – Система координат.

Тип результата Geometry

static from_wkb(wkb, coordsystem=None)
Создает геометрический объект из строки формата WKB.

Параметры

- **wkb** (bytes) – Строка WKB
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии.

Список 145: Пример.

```
wkb = b'\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00$@\x00\x00\x00\x00\x00\x00
↪$@'
pnt = Geometry.from_wkb(wkb)
```

Тип результата Geometry

static from_wkt(wkt, coordsystem=None)
Создает геометрический объект из строки формата WKT.

Параметры

- **wkt** (str) – Строка WKT. Допустимо задание строки в формате с указанием SRID (EWKT). В данном случае система координат для создаваемой геометрии будет установлено исходя их этого значения.
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии. Если строка задана в виде EWKT и указано значение SRID, игнорируется.

Список 146: Пример.

```
polygon = Geometry.from_wkt('POLYGON ((10 10, 100 100, 100 10, 10 10))')
point = Geometry.from_wkt('SRID=4326;POINT (10 10)')
crs = CoordSystem.from_epsg(4326)
pline = Geometry.from_wkt('LINESTRING (30 10, 10 30, 40 40)', crs)
```

Тип результата Geometry

get_area(u=None)

Рассчитывает площадь, если объект площадной. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в квадратных метрах.

Список 147: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
poly = Polygon([(0,0), (0,2), (2, 2), (2,0)], cs=csMercatorKm)
print('Area Mercator km:', poly.get_area())
print('Area Mercator m:', poly.get_area(Unit.sq_m))
print('Perimeter Mercator km:', poly.get_perimeter())
print('Perimeter Mercator m:', poly.get_perimeter(Unit.m))
# В Широте/Долготе
poly.coordsystem = CoordSystem.from_prj("1, 104")
print('Area LL m:', poly.get_area())
print('Area LL km:', poly.get_area(Unit.sq_km))
print('Perimeter LL m:', poly.get_perimeter())
print('Perimeter LL km:', poly.get_perimeter(Unit.km))
'''
>>> Area Mercator km: 4.017946986632519
>>> Area Mercator m: 4017946.9866325194
>>> Perimeter Mercator km: 8.017972048421552
>>> Perimeter Mercator m: 8017.972048421552
>>> Area LL m: 49452172514.0342
>>> Area LL km: 49452.1725140342
>>> Perimeter LL m: 889423.5067063896
>>> Perimeter LL km: 889.4235067063896
'''
```

Параметры **u** (Optional[AreaUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_distance(other, u=None)

Производит расчет расстояния до объекта other. Результат возвращает в СК текущего объекта.

Параметры

- **other** (Geometry) – Анализируемый объект.
- **u** (Optional[LinearUnit]) – Единицы измерения, в которых требуется получить результат.

Список 148: Пример.

```
# Создадим геометрию без СК
first = Point(0, 0)
second = Point(10, 0)
print('В плане:', first.get_distance(second))
#Установим разные СК для точек
first.coordsystem = CoordSystem.from_prj("10, 104, 7, 0")
second.coordsystem = CoordSystem.from_prj("1, 104")
print('На сфере м:', first.get_distance(second))
print('На сфере км:', first.get_distance(second, Unit.km))
'''
>>> В плане: 10.0
>>> На сфере м: 1111948.7428468117
>>> На сфере км: 1111.9487428468117
'''
```

Тип результата float

get_length(u=None)

Рассчитывает длину геометрии. Метод применим только для линейных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Список 149: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
ls = Line((0,0), (10,0), csMercatorKm)
print('Length Mercator km:', ls.get_length())
print('Lenght Mercator m:', ls.get_length(Unit.m))
# В Широте/Долготе
csLL = CoordSystem.from_prj("1, 104")
ls = Line((0,0), (10,0), csLL)
print('Length LL m:', ls.get_length())
print('Length LL km:', ls.get_length(Unit.km))
'''
>>> Length Mercator km: 9.988805508567783
>>> Lenght Mercator m: 9988.805508567782
>>> Length LL m: 1111948.7428468117
>>> Length LL km: 1111.9487428468117
'''
```

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_perimeter(u=None)

Рассчитывает периметр геометрии. Метод применим только для площадных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Пример см. get_area()

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

intersection(other)
Возвращает область пересечения с другой геометрией.

Тип результата Geometry

intersects(other)
Возвращает True, если геометрии пересекаются.

Тип результата bool

property is_valid
Проверяет геометрию на валидность.

Тип результата bool

property is_valid_reason
Если геометрия неправильная, возвращает краткую аннотацию причины.

Тип результата str

property name
Возвращает наименование геометрического объекта.

Тип результата str

overlaps(other)
Возвращает True, если пересечение геометрий отличается от обеих геометрий.

Тип результата bool

static point_by_azimuth(point, azimuth, distance, cs=None)
Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.

См.также:

Обратная функция distance_by_points()

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, относительно которой производится расчет. Если задана СК, то в ней.
- **azimuth** (float) – Азимут в градусах, указывающий направление.
- **distance** (float) – Расстояние по азимуту. Задается в метрах.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Pnt

Результат Возвращает результирующую точку.

relate(other)
Проверяет отношения между объектами. Подробнее см. DE-9IM

Тип результата str

reproject(cs)
Перепроецирует геометрию в другую систему координат.

Параметры **cs** (CoordSystem) – СК, в которой требуется получить объект.

Тип результата Geometry

rotate(point, angle)
Поворот геометрии относительно заданной точки. Поворот производится против часовой стрелки.

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, вокруг которой необходимо произвести поворот.
- **angle** (float) – Угол поворота в градусах.

Тип результата Geometry

scale(kx, ky)
Масштабирует объект по заданным коэффициентам масштабирования. После выполнения операции центр результирующего объекта остается прежним.

Параметры

- **kx** (float) – Масштабирование по координате X.
- **ky** (float) – Масштабирование по координате Y.

Тип результата Geometry

shift(dx, dy)
Смещает объект на заданную величину. Значения задаются в текущей проекции.

Параметры

- **dx** (float) – Сдвиг по координате X.
- **dy** (float) – Сдвиг по координате Y.

Тип результата Geometry

property startAngle
Начальный угол дуги.

Тип результата float

symmetric_difference(other)
Возвращает логический XOR областей геометрий (объединение разниц).

Параметры **other** (Geometry) – Геометрия для анализа.

Тип результата Geometry

to_geojson()
Преобразует геометрию в формат „GeoJSON“

Тип результата str

to_linestring()
Пробует геометрию преобразовать в линейный объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

to_polygon()
Пробует геометрию преобразовать в площадной объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

touches(other)
Возвращает True, если геометрии соприкасаются.

Тип результата bool

property type
Возвращает тип геометрического элемента.

Таблица 85: Возможные значения

Значение	Наименование
Unknown	Не определен
Point	Точка
Line	Линия
LineString	Полилиния
Polygon	Полигон
MultiPoint	Коллекция точек
MultiLineString	Коллекция полилиний
MultiPolygon	Коллекция полигонов
GeometryCollection	Смешанная коллекция
Arc	Дуга
Ellipse	Эллипс
Rectangle	Прямоугольник
RoundedRectangle	Скругленный прямоугольник
Text	Текст

Список 150: Пример.

```
point = Point(10,10)
if point.type == GeometryType.Point:
    print('Это точка')
```

Тип результата GeometryType

union(other)
Возвращает результат объединения двух геометрий.

Тип результата Geometry

within(other)
Возвращает True, если геометрия находится полностью внутри геометрии other.

Тип результата bool

property wkb
Возвращает WKB строку для геометрии.

Тип результата bytes

property wkt
Возвращает WKT строку для геометрии.

Тип результата str

property xRadius
Радиус большой полуоси прямоугольника, в который вписана дуга.

Тип результата float

property yRadius
Радиус малой полуоси прямоугольника, в который вписана дуга.

Тип результата float

Text - Текст

class axipy.mi.**Text**(text, rect, view=None, angle=0, cs=None)

Базовые классы: axipy.da.Geometry

Геометрический объект типа текст.

Предупреждение: Геометрия текста и, в отличие от остальных типов объектов, определяется кроме геометрического представления так же и стилем его оформления axipy.da.TextStyle. Так же стоит заметить, что параметры геометрии текста зависит от того, куда (с карту или отчет) будет добавлен созданный текст. Поэтому созданный объект для карты должен быть добавлен в карту, а для отчета - в отчет.

Примечание: Для создания текстового объекта с указанием его размера в поинтах можно использовать метод create_by_style()

Параметры

- **text** (str) – Строка с текстом. Многострочный текст можно задать, вставив символ «\n» в место переноса.
- **rect** (Union[Rect, QRectF]) – Прямоугольник, в который будет вписан текст. Прямоугольник задается в координатах отчета. Верхней левой точкой является startPoint. В этот прямоугольник будет произведена попытка размещения текста.
- **angle** (float) – Угол поворота текстового объекта. Задается значением в градусах против ЧС по отношению к горизонтали.
- **view** (Union[MapView, ReportView, None]) – Окно карты или отчета.
- **cs** (Optional[CoordSystem]) – Система Координат, в которой создается геометрия.

Список 151: Пример создания текста и вставки его в отчет.

```
report_view = view_manager.create_reportview()
r = Rect(8, 6, 11, 7)
text = Text("Пример\nтекста", rect=r, angle=20, view=report_view)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
geomItem = GeometryReportItem()
geomItem.geometry = text
geomItem.style = Style.for_geometry(text)
report_view.report.items.add(geomItem)
```

Methods:

affine_transform(trans)	Трансформирует объект исходя из заданных параметров трансформации.
almost_equals(other, tolerance)	Производит примерное сравнения с другой геометрией в пределах заданной точности.
boundary()	Возвращает границы геометрии в виде полилинии.
buffer(distance[, resolution, capStyle, …])	Производит построение буфера.
centroid()	Возвращает центроид геометрии.
clone()	Создает копию объекта.
contains(other)	Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.
convex_hull()	Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.
covers(other)	Возвращает True, если геометрия охватывает геометрию other.
create_by_style(text, point, style, view[, …])	Создает текстовый объект по точке привязки и стиля для карты или отчета.
crosses(other)	Возвращает True, если при пересечении геометрий объекты частично пересекаются.
difference(other)	Возвращает область первой геометрии, которая не пересечена второй геометрией.
disjoint(other)	Возвращает True, если геометрии не пересекаются и не соприкасаются.
distance_by_points(start, end[, cs])	Производит расчет расстояния между двумя точками и азимут от первой до второй точки.
envelope()	Возвращает полигон, описывающий заданную геометрию.
equals(other)	Производит сравнение с другой геометрией.
from_geojson(json[, cs])	Возвращает геометрию из ее „GeoJSON“ представления.
from_wkb(wkb[, coordsystem])	Создает геометрический объект из строки формата WKB.
from_wkt(wkt[, coordsystem])	Создает геометрический объект из строки формата WKT.
get_area([u])	Рассчитывает площадь, если объект площадной.

continues on next page

Таблица 86 – продолжение с предыдущей страницы

get_distance(other[, u])	Производит расчет расстояния до объекта other.
get_length([u])	Рассчитывает длину геометрии.
get_perimeter([u])	Рассчитывает периметр геометрии.
intersection(other)	Возвращает область пересечения с другой геометрией.
intersects(other)	Возвращает True, если геометрии пересекаются.
overlaps(other)	Возвращает True, если пересечение геометрий отличается от обеих геометрий.
point_by_azimuth(point, azimuth, distance[, cs])	Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.
relate(other)	Проверяет отношения между объектами.
reproject(cs)	Перепроецирует геометрию в другую систему координат.
rotate(point, angle)	Поворот геометрии относительно заданной точки.
scale(kx, ky)	Масштабирует объект по заданным коэффициентам масштабирования.
shift(dx, dy)	Смещает объект на заданную величину.
symmetric_difference(other)	Возвращает логический XOR областей геометрий (объединение разниц).
to_geojson()	Преобразует геометрию в формат „GeoJSON“
to_linestring()	Пробует геометрию преобразовать в линейный объект.
to_polygon()	Пробует геометрию преобразовать в площадной объект.
touches(other)	Возвращает True, если геометрии соприкасаются.
union(other)	Возвращает результат объединения двух геометрий.
within(other)	Возвращает True, если геометрия находится полностью внутри геометрии other.

Attributes:

angle	Угол поворота текста, отсчитываемый от горизонтали против часовой стрелки
bounds	Возвращает минимальный ограничивающий прямоугольник.
coordsystem	Система Координат (СК) геометрии.
is_valid	Проверяет геометрию на валидность.

continues on next page

Таблица 87 – продолжение с предыдущей страницы

is_valid_reason	Если геометрия неправильная, возвращает краткую аннотацию причины.
name	Возвращает наименование геометрического объекта.
startPoint	Координаты точки привязки.
text	Текст.
type	Возвращает тип геометрического элемента.
wkb	Возвращает WKB строку для геометрии.
wkt	Возвращает WKT строку для геометрии.

affine_transform(trans)
Трансформирует объект исходя из заданных параметров трансформации.

См.также:

Для простых операций типа сдвиг, масштабирование и поворот, рекомендуется использовать shift(), scale(), rotate() соответственно.

Параметры **trans** (QTransform) – Матрица трансформации.

Тип результата Geometry

almost_equals(other, tolerance)
Производит примерное сравнения с другой геометрией в пределах заданной точности.

Параметры

- **other** (Geometry) – Сравниваемый объект.
- **tolerance** (float) – Точность сравнения.

Тип результата bool

Результат Возвращает True, если геометрии равны в пределах заданного отклонения.

property angle
Угол поворота текста, отсчитываемый от горизонтали против часовой стрелки

Тип результата float

boundary()
Возвращает границы геометрии в виде полилинии.

Тип результата Geometry

property bounds
Возвращает минимальный ограничивающий прямоугольник.

Тип результата Rect

buffer(distance, resolution=16, capStyle=1, joinStyle=1, mitreLimit=5.0)
Производит построение буфера.

Параметры

- **distance** (float) – Ширина буфера.
- **resolution** (int) – Количество сегментов на квадрант.
- **capStyle** (int) – Стиль окончания.
- **joinStyle** (int) – Стиль соединения.
- **mitreLimit** (float) – Предел среза.

Тип результата Geometry

centroid()
Возвращает центроид геометрии.

Тип результата Geometry

clone()
Создает копию объекта.

Тип результата Geometry

contains(other)
Возвращает True, если геометрия полностью содержит передаваемую в качестве параметра геометрию.

Тип результата bool

convex_hull()
Возвращает минимальный окаймляющий полигон со всеми выпуклыми углами.

Тип результата Geometry

property coordsystem
Система Координат (СК) геометрии.

Тип результата CoordSystem

covers(other)
Возвращает True, если геометрия охватывает геометрию other.

Тип результата bool

classmethod create_by_style(text, point, style, view, angle=0, cs=None)
Создает текстовый объект по точке привязки и стиля для карты или отчета. Результирующий объект будет корректен только для текущего состояния окна карты или отчета, куда он должен быть добавлен. Это связано с тем, что расчет размера производится на основании текущего установленного масштаба. Поэтому при изменении масштаба перед добавлением необходимо заново создать текстовый объект.

Параметры

- **text** (str) – Текст.
- **point** (Union[Pnt, QPointF]) – Точка привязки startPoint. Этой точкой служит левая верхняя точка.
- **style** (Style) – Стиль оформления текста.
- **view** (Union[MapView, ReportView]) – Окно карты или отчета.
- **angle** (float) – Угол поворота текстового объекта. Задается значением в градусах против ЧС по отношению к горизонтали.

- **cs** (Optional[CoordSystem]) – Система Координат, в которой создается геометрия.

Список 152: Пример.

```
report_view = view_manager.create_reportview()
style_txt = Style.from_mapinfo('Font ("Times New Roman", 0, 23, 16711680)')
text = Text.create_by_style("Пример\nтекста2", (10,10), style=style_txt,␣
↪view=report_view,  angle=20)
# Поменяем угол
text.angle = -20
# Изменим начальную точку
text.startPoint = (11, 11)
```

crosses(other)
Возвращает True, если при пересечении геометрий объекты частично пересекаются.

Тип результата bool

difference(other)
Возвращает область первой геометрии, которая не пересечена второй геометрией.

Тип результата Geometry

disjoint(other)
Возвращает True, если геометрии не пересекаются и не соприкасаются.

Тип результата bool

static distance_by_points(start, end, cs=None)
Производит расчет расстояния между двумя точками и азимут от первой до второй точки.

См.также:

Обратная функция point_by_azimuth()

Параметры

- **start** (Union[Pnt, Tuple[float, float]]) – Начальная точка. Если задана СК, то координаты в ней.
- **end** (Union[Pnt, Tuple[float, float]]) – Конечная точка. Если задана СК, то координаты в ней.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Tuple

Результат Возвращается пара значений (расстояние в метрах, азимут в градусах)

envelope()
Возвращает полигон, описывающий заданную геометрию.

Тип результата Geometry

equals(other)
Производит сравнение с другой геометрией.

Параметры **other** (Geometry) – Сравниваемая геометрия.

Тип результата bool

Результат Возвращает True, если геометрии равны.

static from_geojson(json, cs=None)
Возвращает геометрию из ее „GeoJSON“ представления.

Параметры

- **json** (str) – json представление в виде строки.
- **cs** (Optional[CoordSystem]) – Система координат.

Тип результата Geometry

static from_wkb(wkb, coordsystem=None)
Создает геометрический объект из строки формата WKB.

Параметры

- **wkb** (bytes) – Строка WKB
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии.

Список 153: Пример.

```
wkb = b'\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00$@\x00\x00\x00\x00\x00\x00
↪$@'
pnt = Geometry.from_wkb(wkb)
```

Тип результата Geometry

static from_wkt(wkt, coordsystem=None)
Создает геометрический объект из строки формата WKT.

Параметры

- **wkt** (str) – Строка WKT. Допустимо задание строки в формате с указанием SRID (EWKT). В данном случае система координат для создаваемой геометрии будет установлено исходя их этого значения.
- **coordsystem** (Optional[CoordSystem]) – Система координат, которая будет установлена для геометрии. Если строка задана в виде EWKT и указано значение SRID, игнорируется.

Список 154: Пример.

```
polygon = Geometry.from_wkt('POLYGON ((10 10, 100 100, 100 10, 10 10))')
point = Geometry.from_wkt('SRID=4326;POINT (10 10)')
crs = CoordSystem.from_epsg(4326)
pline = Geometry.from_wkt('LINESTRING (30 10, 10 30, 40 40)', crs)
```

Тип результата Geometry

get_area(u=None)

Рассчитывает площадь, если объект площадной. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в квадратных метрах.

Список 155: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
poly = Polygon([(0,0), (0,2), (2, 2), (2,0)], cs=csMercatorKm)
print('Area Mercator km:', poly.get_area())
print('Area Mercator m:', poly.get_area(Unit.sq_m))
print('Perimeter Mercator km:', poly.get_perimeter())
print('Perimeter Mercator m:', poly.get_perimeter(Unit.m))
# В Широте/Долготе
poly.coordsystem = CoordSystem.from_prj("1, 104")
print('Area LL m:', poly.get_area())
print('Area LL km:', poly.get_area(Unit.sq_km))
print('Perimeter LL m:', poly.get_perimeter())
print('Perimeter LL km:', poly.get_perimeter(Unit.km))
'''
>>> Area Mercator km: 4.017946986632519
>>> Area Mercator m: 4017946.9866325194
>>> Perimeter Mercator km: 8.017972048421552
>>> Perimeter Mercator m: 8017.972048421552
>>> Area LL m: 49452172514.0342
>>> Area LL km: 49452.1725140342
>>> Perimeter LL m: 889423.5067063896
>>> Perimeter LL km: 889.4235067063896
'''
```

Параметры **u** (Optional[AreaUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_distance(other, u=None)

Производит расчет расстояния до объекта other. Результат возвращает в СК текущего объекта.

Параметры

- **other** (Geometry) – Анализируемый объект.
- **u** (Optional[LinearUnit]) – Единицы измерения, в которых требуется получить результат.

Список 156: Пример.

```
# Создадим геометрию без СК
first = Point(0, 0)
second = Point(10, 0)
print('В плане:', first.get_distance(second))
#Установим разные СК для точек
first.coordsystem = CoordSystem.from_prj("10, 104, 7, 0")
second.coordsystem = CoordSystem.from_prj("1, 104")
```

(continues on next page)

(продолжение с предыдущей страницы)

```
print('На сфере м:', first.get_distance(second))
print('На сфере км:', first.get_distance(second, Unit.km))
'''
>>> В плане: 10.0
>>> На сфере м: 1111948.7428468117
>>> На сфере км: 1111.9487428468117
'''
```

Тип результата float

get_length(u=None)

Рассчитывает длину геометрии. Метод применим только для линейных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Список 157: Пример.

```
# В проекции Меркатора:
csMercatorKm = CoordSystem. from_prj("10, 104, 1, 0")
ls = Line((0,0), (10,0), csMercatorKm)
print('Length Mercator km:', ls.get_length())
print('Lenght Mercator m:', ls.get_length(Unit.m))
# В Широте/Долготе
csLL = CoordSystem.from_prj("1, 104")
ls = Line((0,0), (10,0), csLL)
print('Length LL m:', ls.get_length())
print('Length LL km:', ls.get_length(Unit.km))
'''
>>> Length Mercator km: 9.988805508567783
>>> Lenght Mercator m: 9988.805508567782
>>> Length LL m: 1111948.7428468117
>>> Length LL km: 1111.9487428468117
'''
```

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

get_perimeter(u=None)

Рассчитывает периметр геометрии. Метод применим только для площадных объектов. В противном случае возвращает 0. В случае, если СК задана как Широта/Долгота, то расчет производится на сфере в метрах.

Пример см. get_area()

Параметры **u** (Optional[LinearUnit]) – Единица измерения, в которой необходимо получить результат. Если не задана, то используется единица измерения для СК. Если и она не задана, то производится расчет на плоскости.

Тип результата float

intersection(other)
Возвращает область пересечения с другой геометрией.

Тип результата Geometry

intersects(other)
Возвращает True, если геометрии пересекаются.

Тип результата bool

property is_valid
Проверяет геометрию на валидность.

Тип результата bool

property is_valid_reason
Если геометрия неправильная, возвращает краткую аннотацию причины.

Тип результата str

property name
Возвращает наименование геометрического объекта.

Тип результата str

overlaps(other)
Возвращает True, если пересечение геометрий отличается от обеих геометрий.

Тип результата bool

static point_by_azimuth(point, azimuth, distance, cs=None)
Производит расчет координат точки относительно заданной, и находящейся на расстоянии distance по направлению azimuth.

См.также:

Обратная функция distance_by_points()

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, относительно которой производится расчет. Если задана СК, то в ней.
- **azimuth** (float) – Азимут в градусах, указывающий направление.
- **distance** (float) – Расстояние по азимуту. Задается в метрах.
- **cs** (Optional[CoordSystem]) – СК, на базе эллипсоида которой производится расчет. Если не задана, то расчет производится на плоскости.

Тип результата Pnt

Результат Возвращает результирующую точку.

relate(other)
Проверяет отношения между объектами. Подробнее см. DE-9IM

Тип результата str

reproject(cs)
Перепроецирует геометрию в другую систему координат.

Параметры cs (CoordSystem) – СК, в которой требуется получить объект.

Тип результата Geometry

rotate(point, angle)
Поворот геометрии относительно заданной точки. Поворот производится против часовой стрелки.

Параметры

- **point** (Union[Pnt, Tuple[float, float]]) – Точка, вокруг которой необходимо произвести поворот.
- **angle** (float) – Угол поворота в градусах.

Тип результата Geometry

scale(kx, ky)
Масштабирует объект по заданным коэффициентам масштабирования. После выполнения операции центр результирующего объекта остается прежним.

Параметры

- **kx** (float) – Масштабирование по координате X.
- **ky** (float) – Масштабирование по координате Y.

Тип результата Geometry

shift(dx, dy)
Смещает объект на заданную величину. Значения задаются в текущей проекции.

Параметры

- **dx** (float) – Сдвиг по координате X.
- **dy** (float) – Сдвиг по координате Y.

Тип результата Geometry

property startPoint
Координаты точки привязки.

Тип результата Pnt

symmetric_difference(other)
Возвращает логический XOR областей геометрий (объединение разниц).

Параметры **other** (Geometry) – Геометрия для анализа.

Тип результата Geometry

property text
Текст.

Тип результата str

to_geojson()
Преобразует геометрию в формат „GeoJSON“

Тип результата str

to_linestring()
Пробует геометрию преобразовать в линейный объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

to_polygon()
Пробует геометрию преобразовать в площадной объект. В случае неудачи возвращает None.

Тип результата Optional[Geometry]

touches(other)
Возвращает True, если геометрии соприкасаются.

Тип результата bool

property type
Возвращает тип геометрического элемента.

Таблица 88: Возможные значения

Значение	Наименование
Unknown	Не определен
Point	Точка
Line	Линия
LineString	Полилиния
Polygon	Полигон
MultiPoint	Коллекция точек
MultiLineString	Коллекция полилиний
MultiPolygon	Коллекция полигонов
GeometryCollection	Смешанная коллекция
Arc	Дуга
Ellipse	Эллипс
Rectangle	Прямоугольник
RoundedRectangle	Скругленный прямоугольник
Text	Текст

Список 158: Пример.

```
point = Point(10,10)
if point.type == GeometryType.Point:
    print('Это точка')
```

Тип результата GeometryType

union(other)
Возвращает результат объединения двух геометрий.

Тип результата Geometry

within(other)
Возвращает True, если геометрия находится полностью внутри геометрии other.

Тип результата bool

property wkb
Возвращает WKB строку для геометрии.

Тип результата bytes

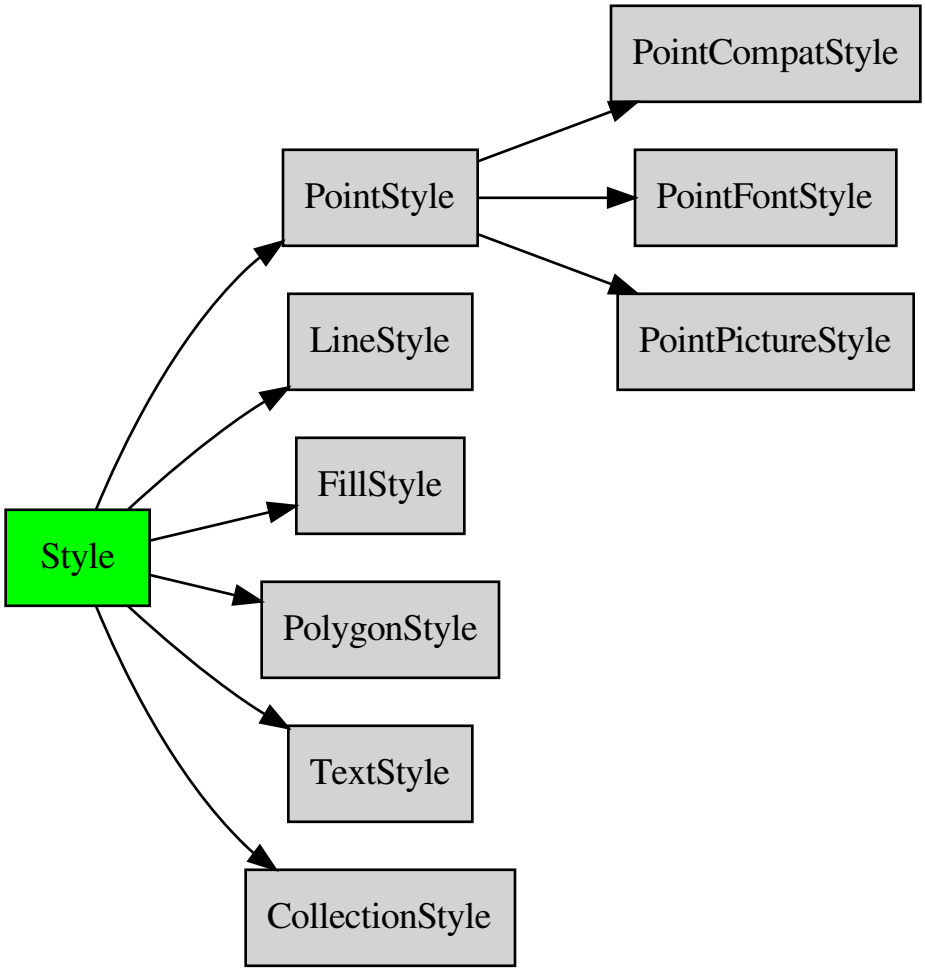
property wkt
Возвращает WKT строку для геометрии.

Тип результата str

16.1.6.16 axipy.da style

Style - Стиль

Иерархия классов стилей геометрических объектов:



class axipy.da.**Style**

Абстрактный класс стиля оформления геометрического объекта.

Определяет как будет отрисован геометрический объект.

Примечание: Для получения текстового представления стиля можно воспользоваться функцией `str`.

---

clone()

Создаёт копию объекта стиля

Тип результата Style

draw(geometry, painter)
Рисует геометрический объект с текущим стилем в произвольном контексте вывода. Это может быть востребовано при желании отрисовать геометрию со стилем на форме или диалоге.

Параметры

- **geometry** (Geometry) – Геометрия. Должна соответствовать стилю. Т.е. если объект полигон, а стиль для рисования точечных объектов, то ничего нарисовано не будет.
- **painter** (QPainter) – Контекст вывода.

Список 159: Пример отрисовки в растре и сохранение результата в файле.

```
image = QImage(100, 100, QImage.Format_ARGB32_Premultiplied)
image.fill(Qt.white)
painter = QPainter(image)
point = Point(50,50)
style = PointStyle.create_mi_font(42, Qt.red, 24)
style.draw(point, painter)
image.save(filename)
```

classmethod for_geometry(geom)
Возвращает стиль по умолчанию для переданного объекта.

Параметры geom (Geometry) – Геометрический объект, для которого необходимо получить соответствующий ему стиль.

Тип результата Style

classmethod from_mapinfo(mapbasic_string)
Получает стиль из строки формата MapBasic.

Параметры mapbasic_string (str) – Строка в формате MapBasic.

```
style = Style.from_mapinfo("Pen (1, 2, 0) Brush (8, 255)")
```

Тип результата Style

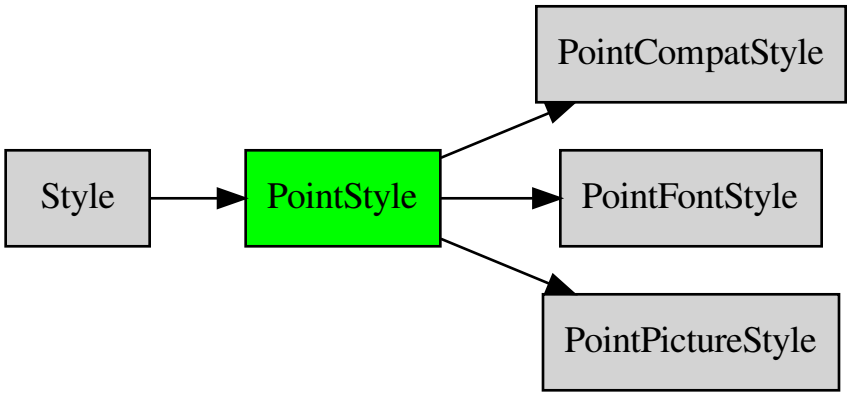
to_mapinfo()
Возвращает строковое представление в формате MapBasic.

```
print(style.to_mapinfo())
'''
>>> Pen (1, 2, 0) Brush (8, 255)
'''
```

Тип результата str

PointStyle - Стиль точек

Иерархия классов стилей геометрических объектов:



class axipy.da.**PointStyle**

Базовые классы: axipy.da.Style

Стиль оформления точечных объектов.

По умолчанию создается стиль на базе шрифта True Type, а параметры аналогичны значениям по умолчанию в методе create_mi_font().

Поддерживается 3 вида оформления:

- Совместимое с MapInfo версии 3. Для создания такого стиля необходимо использовать create_mi_compat().
- На базе шрифтов True Type. Задано по умолчанию. Стиль создается посредством create_mi_font().
- На базе растрового файла. Стиль можно создать с помощью create_mi_picture().

Список 160: Пример.

```
style_1 = PointStyle.create_mi_compat(color = Qt.blue)
style_2 = PointStyle.create_mi_font(42, Qt.red, 24)
style_3 = PointStyle.create_mi_picture('AMBU-64.bmp')
```

Methods:

clone()	Создаёт копию объекта стиля
create_mi_compat([symbol, color, pointSize])	Создание стиля в виде совместимого с MapInfo 3 PointCompatStyle.
create_mi_font([symbol, color, size, ...])	Создание стиля на базе шрифта True Type PointFontStyle.

continues on next page

Таблица 89 – продолжение с предыдущей страницы

create_mi_picture(filename[, color, size, ...])	Создание стиля с ссылкой на растровый файл PointPictureStyle.
draw(geometry, painter)	Рисует геометрический объект с текущим стилем в произвольном контексте вывода.
for_geometry(geom)	Возвращает стиль по умолчанию для переданного объекта.
from_mapinfo(mapbasic_string)	Получает стиль из строки формата MapBasic.
to_mapinfo()	Возвращает строковое представление в формате MapBasic.

Attributes:

color	Цвет символа.

clone()
Создаёт копию объекта стиля

Тип результата Style

property color
Цвет символа.

Тип результата QColor

static create_mi_compat(symbol=35, color=PySide2.QtCore.Qt.GlobalColor.red, pointSize=8)
Создание стиля в виде совместимого с MapInfo 3 PointCompatStyle.

Параметры

- **symbol** (int) – Номер символа, который будет отображен при отрисовке. Для создания невидимого символа используйте значение 31. Стандартный набор условных знаков включает символы от 31 до 67,
- **color** (QColor) – Цвет символа
- **pointSize** (int) – Целое число, размер символа в пунктах от 1 до 48.

Тип результата PointCompatStyle

static create_mi_font(symbol=36, color=PySide2.QtCore.Qt.GlobalColor.red, size=8, fontname='Axioma MI MapSymbols', fontstyle=0, rotation=0.0)
Создание стиля на базе шрифта True Type PointFontStyle.

Параметры

- **symbol** (int) – Целое, имеющее значение 31 или больше, определяющее, какой используется символ из шрифтов TrueType. Для создания невидимого символа используйте значение 31.
- **color** (QColor) – Цвет символа
- **pointSize** – Целое число, размер символа в пунктах от 1 до 48;

- **fontname** (str) – Строка с именем шрифта TrueType (например, значение по умолчание „Axioma MI MapSymbols“)
- **fontstyle** (int) – Стиль дополнительного оформления, например, курсивный текст. Возможные параметры см. в таблице ниже. Для указания нескольких параметров их суммируют между собой.
- **rotation** (float) – Угол поворота символа в градусах.

Тип результата PointFontStyle

static create_mi_picture(filename, color=PySide2.QtCore.Qt.GlobalColor.black, size=12, customstyle=0)

Создание стиля с ссылкой на растровый файл PointPictureStyle.

Параметры

- **filename** (str) – Наименование растрового файла. Строка до 31 символа длиной. Данный файл должен находится в каталоге CustSymb с ресурсами. Например, “Arrow.BMP”.
- **color** (QColor) – Цвет символа.
- **size** (int) – Размер символа в в пунктах от 1 до 48.
- **customstyle** (int) – Задание дополнительных параметров стиля оформления.

Тип результата PointPictureStyle

draw(geometry, painter)

Рисует геометрический объект с текущим стилем в произвольном контексте вывода. Это может быть востребовано при желании отрисовать геометрию со стилем на форме или диалоге.

Параметры

- **geometry** (Geometry) – Геометрия. Должна соответствовать стилю. Т.е. если объект полигон, а стиль для рисования точечных объектов, то ничего нарисовано не будет.
- **painter** (QPainter) – Контекст вывода.

Список 161: Пример отрисовки в растре и сохранение результата в файле.

```
image = QImage(100, 100, QImage.Format_ARGB32_Premultiplied)
image.fill(Qt.white)
painter = QPainter(image)
point = Point(50,50)
style = PointStyle.create_mi_font(42, Qt.red, 24)
style.draw(point, painter)
image.save(filename)
```

classmethod for_geometry(geom)

Возвращает стиль по умолчанию для переданного объекта.

Параметры geom (Geometry) – Геометрический объект, для которого необходимо получить соответствующий ему стиль.

Тип результата Style

classmethod from_mapinfo(mapbasic_string)
Получает стиль из строки формата MapBasic.

Параметры mapbasic_string (str) – Строка в формате MapBasic.

```
style = Style.from_mapinfo("Pen (1, 2, 0) Brush (8, 255)")
```

Тип результата Style

to_mapinfo()
Возвращает строковое представление в формате MapBasic.

```
print(style.to_mapinfo())
'''
>>> Pen (1, 2, 0) Brush (8, 255)
'''
```

Тип результата str

PointCompatStyle - Стиль, совместимый с MapInfo 3

class axipy.da.**PointCompatStyle**(symbol=35, color=PySide2.QtCore.Qt.GlobalColor.red, pointSize=8)

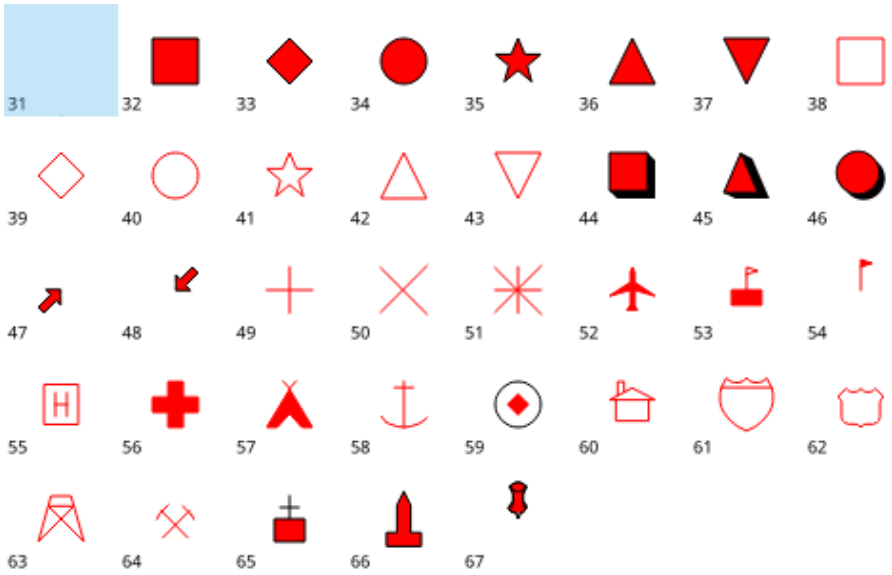
Базовые классы: axipy.da.PointStyle

Класс стиля, совместимого с MapInfo 3.

Параметры

- **symbol** (int) – Номер символа, который будет отображен при отрисовке. Для создания невидимого символа используйте значение 31. Стандартный набор условных знаков включает символы от 31 до 67,
- **color** (QColor) – Цвет символа
- **pointSize** (int) – Целое число, размер символа в пунктах от 1 до 48.

В системе доступны следующие стили:



Список 162: Пример.

```
style = PointCompatStyle()
style.color = Qt.blue
style.symbol = 43
```

Methods:

clone()	Создаёт копию объекта стиля
create_mi_compat([symbol, color, pointSize])	Создание стиля в виде совместимого с MapInfo 3 PointCompatStyle.
create_mi_font([symbol, color, size, ...])	Создание стиля на базе шрифта True Type PointFontStyle.
create_mi_picture(filename[, color, size, ...])	Создание стиля с ссылкой на растровый файл PointPictureStyle.
draw(geometry, painter)	Рисует геометрический объект с текущим стилем в произвольном контексте вывода.
for_geometry(geom)	Возвращает стиль по умолчанию для переданного объекта.
from_mapinfo(mapbasic_string)	Получает стиль из строки формата MapBasic.
to_mapinfo()	Возвращает строковое представление в формате MapBasic.

Attributes:

color	Цвет символа.
size	Размер символа в пунктах.
symbol	Номер символа.

clone()
Создаёт копию объекта стиля

Тип результата Style

property color
Цвет символа.

Тип результата QColor

static create_mi_compat(symbol=35, color=PySide2.QtCore.Qt.GlobalColor.red, pointSize=8)
Создание стиля в виде совместимого с MapInfo 3 PointCompatStyle.

Параметры

- **symbol** (int) – Номер символа, который будет отображен при отрисовке. Для создания невидимого символа используйте значение 31. Стандартный набор условных знаков включает символы от 31 до 67,
- **color** (QColor) – Цвет символа
- **pointSize** (int) – Целое число, размер символа в пунктах от 1 до 48.

Тип результата PointCompatStyle

static create_mi_font(symbol=36, color=PySide2.QtCore.Qt.GlobalColor.red, size=8, fontname='Axioma MI MapSymbols', fontstyle=0, rotation=0.0)
Создание стиля на базе шрифта True Type PointFontStyle.

Параметры

- **symbol** (int) – Целое, имеющее значение 31 или больше, определяющее, какой используется символ из шрифтов TrueType. Для создания невидимого символа используйте значение 31.
- **color** (QColor) – Цвет символа
- **pointSize** – Целое число, размер символа в пунктах от 1 до 48;
- **fontname** (str) – Строка с именем шрифта TrueType (например, значение по умолчание „Axioma MI MapSymbols“)
- **fontstyle** (int) – Стиль дополнительного оформления, например, курсивный текст. Возможные параметры см. в таблице ниже. Для указания нескольких параметров их суммируют между собой.
- **rotation** (float) – Угол поворота символа в градусах.

Тип результата PointFontStyle

static create_mi_picture(filename, color=PySide2.QtCore.Qt.GlobalColor.black, size=12, customstyle=0)
Создание стиля с ссылкой на растровый файл PointPictureStyle.

Параметры

- **filename** (str) – Наименование растрового файла. Строка до 31 символа длиной. Данный файл должен находится в каталоге CustSymb с ресурсами. Например, “Arrow.BMP”.
- **color** (QColor) – Цвет символа.
- **size** (int) – Размер символа в в пунктах от 1 до 48.

- **customstyle** (int) – Задание дополнительных параметров стиля оформления.

Тип результата PointPictureStyle

draw(geometry, painter)
Рисует геометрический объект с текущим стилем в произвольном контексте вывода. Это может быть востребовано при желании отрисовать геометрию со стилем на форме или диалоге.

Параметры

- **geometry** (Geometry) – Геометрия. Должна соответствовать стилю. Т.е. если объект полигон, а стиль для рисования точечных объектов, то ничего нарисовано не будет.
- **painter** (QPainter) – Контекст вывода.

Список 163: Пример отрисовки в растре и сохранение результата в файле.

```
image = QImage(100, 100, QImage.Format_ARGB32_Premultiplied)
image.fill(Qt.white)
painter = QPainter(image)
point = Point(50,50)
style = PointStyle.create_mi_font(42, Qt.red, 24)
style.draw(point, painter)
image.save(filename)
```

classmethod for_geometry(geom)
Возвращает стиль по умолчанию для переданного объекта.

Параметры geom (Geometry) – Геометрический объект, для которого необходимо получить соответствующий ему стиль.

Тип результата Style

classmethod from_mapinfo(mapbasic_string)
Получает стиль из строки формата MapBasic.

Параметры mapbasic_string (str) – Строка в формате MapBasic.

```
style = Style.from_mapinfo("Pen (1, 2, 0) Brush (8, 255)")
```

Тип результата Style

property size
Размер символа в пунктах.

Тип результата int

property symbol
Номер символа.

Тип результата int

to_mapinfo()
Возвращает строковое представление в формате MapBasic.

```
print(style.to_mapinfo())
'''
>>> Pen (1, 2, 0) Brush (8, 255)
'''
```

Тип результата str

PointFontStyle - Стиль на базе шрифта True Type

class axipy.da.**PointFontStyle**(symbol=36, color=PySide2.QtCore.Qt.GlobalColor.red, size=8, fontname='Axioma MI MapSymbols', fontstyle=0, rotation=0.0)

Базовые классы: axipy.da.PointStyle

Стиль на базе шрифта True Type.

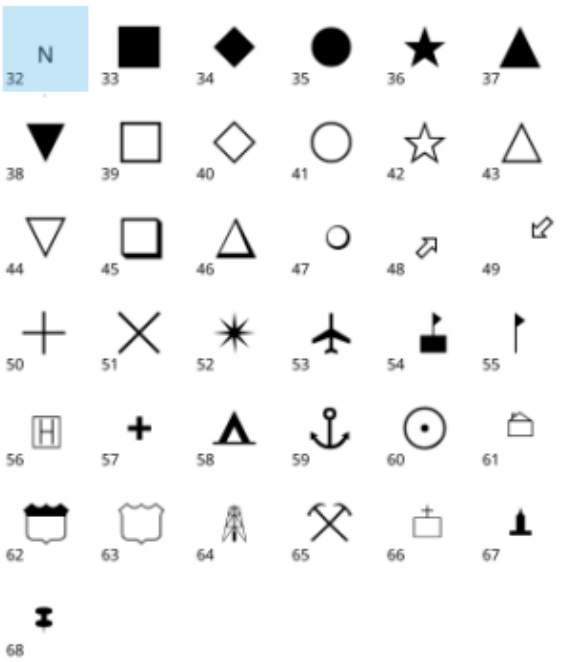
Параметры

- **symbol** (int) – Целое, имеющее значение 31 или больше, определяющее, какой используется символ из шрифтов TrueType. Для создания невидимого символа используйте значение 31.
- **color** (QColor) – Цвет символа
- **pointSize** – Целое число, размер символа в пунктах от 1 до 48;
- **fontname** (str) – Строка с именем шрифта TrueType (например, значение по умолчание „Axioma MI MapSymbols“)
- **fontstyle** (int) – Стиль дополнительного оформления, например, курсивный текст. Возможные параметры см. в таблице ниже. Для указания нескольких параметров их суммируют между собой.
- **rotation** (float) – Угол поворота символа в градусах.

Таблица 93: Возможные значения параметра fontstyle:

Значение	Наименование
0	Обычный текст
1	Жирный текст
16	Черная кайма вокруг символа
32	Тень
256	Белая кайма вокруг символа

В системе доступны следующие стили:



Список 164: Пример.

```
fs = PointFontStyle()
fs.font_name = 'Axioma MI Oil&Gas'
fs.has_shadow = True
fs.rotation = 30
fs.bold = True
fs.size = 44
fs.symbol = 45
fs.black_border = True
fs.white_border = True
```

Attributes:

black_border	Темная окантовка.
bold	Жирный шрифт..
color	Цвет символа.
font_name	Наименование шрифта.
has_shadow	Признак тени.
rotation	Угол поворота.
size	Размер символа в пунктах.
symbol	Номер символа.
white_border	Светлая окантовка.

Methods:

clone()	Создаёт копию объекта стиля
create_mi_compat([symbol, color, pointSize])	Создание стиля в виде совместимого с MapInfo 3 PointCompatStyle.

continues on next page

Таблица 95 – продолжение с предыдущей страницы

create_mi_font([symbol, color, size, ...])	Создание стиля на базе шрифта True Type PointFontStyle.
create_mi_picture(filename[, color, size, ...])	Создание стиля с ссылкой на растровый файл PointPictureStyle.
draw(geometry, painter)	Рисует геометрический объект с текущим стилем в произвольном контексте вывода.
for_geometry(geom)	Возвращает стиль по умолчанию для переданного объекта.
from_mapinfo(mapbasic_string)	Получает стиль из строки формата MapBasic.
to_mapinfo()	Возвращает строковое представление в формате MapBasic.

property black_border
Темная окантовка.

Тип результата bool

property bold
Жирный шрифт..

Тип результата bool

clone()
Создаёт копию объекта стиля

Тип результата Style

property color
Цвет символа.

Тип результата QColor

static create_mi_compat(symbol=35, color=PySide2.QtCore.Qt.GlobalColor.red, pointSize=8)
Создание стиля в виде совместимого с MapInfo 3 PointCompatStyle.

Параметры

- **symbol** (int) – Номер символа, который будет отображен при отрисовке. Для создания невидимого символа используйте значение 31. Стандартный набор условных знаков включает символы от 31 до 67,
- **color** (QColor) – Цвет символа
- **pointSize** (int) – Целое число, размер символа в пунктах от 1 до 48.

Тип результата PointCompatStyle

static create_mi_font(symbol=36, color=PySide2.QtCore.Qt.GlobalColor.red, size=8, fontname='Axioma MI MapSymbols', fontstyle=0, rotation=0.0)
Создание стиля на базе шрифта True Type PointFontStyle.

Параметры

- **symbol** (int) – Целое, имеющее значение 31 или больше, определяющее, какой используется символ из шрифтов TrueType. Для создания невидимого символа используйте значение 31.
- **color** (QColor) – Цвет символа
- **pointSize** – Целое число, размер символа в пунктах от 1 до 48;
- **fontname** (str) – Строка с именем шрифта TrueType (например, значение по умолчание „Axioma MI MapSymbols“)
- **fontstyle** (int) – Стиль дополнительного оформления, например, курсивный текст. Возможные параметры см. в таблице ниже. Для указания нескольких параметров их суммируют между собой.
- **rotation** (float) – Угол поворота символа в градусах.

Тип результата PointFontStyle

static create_mi_picture(filename, color=PySide2.QtCore.Qt.GlobalColor.black, size=12, customstyle=0)

Создание стиля с ссылкой на растровый файл PointPictureStyle.

Параметры

- **filename** (str) – Наименование растрового файла. Строка до 31 символа длиной. Данный файл должен находится в каталоге CustSymb с ресурсами. Например, “Arrow.BMP”.
- **color** (QColor) – Цвет символа.
- **size** (int) – Размер символа в в пунктах от 1 до 48.
- **customstyle** (int) – Задание дополнительных параметров стиля оформления.

Тип результата PointPictureStyle

draw(geometry, painter)

Рисует геометрический объект с текущим стилем в произвольном контексте вывода. Это может быть востребовано при желании отрисовать геометрию со стилем на форме или диалоге.

Параметры

- **geometry** (Geometry) – Геометрия. Должна соответствовать стилю. Т.е. если объект полигон, а стиль для рисования точечных объектов, то ничего нарисовано не будет.
- **painter** (QPainter) – Контекст вывода.

Список 165: Пример отрисовки в растре и сохранение результата в файле.

```
image = QImage(100, 100, QImage.Format_ARGB32_Premultiplied)
image.fill(Qt.white)
painter = QPainter(image)
point = Point(50,50)
style = PointStyle.create_mi_font(42, Qt.red, 24)
style.draw(point, painter)
image.save(filename)
```

property font_name
Наименование шрифта.

Тип результата str

classmethod for_geometry(geom)
Возвращает стиль по умолчанию для переданного объекта.

Параметры geom (Geometry) – Геометрический объект, для которого необходимо получить соответствующий ему стиль.

Тип результата Style

classmethod from_mapinfo(mapbasic_string)
Получает стиль из строки формата MapBasic.

Параметры mapbasic_string (str) – Строка в формате MapBasic.

```
style = Style.from_mapinfo("Pen (1, 2, 0) Brush (8, 255)")
```

Тип результата Style

property has_shadow
Признак тени.

Тип результата bool

property rotation
Угол поворота.

Тип результата float

property size
Размер символа в пунктах.

Тип результата int

property symbol
Номер символа.

Тип результата int

to_mapinfo()
Возвращает строковое представление в формате MapBasic.

```
print(style.to_mapinfo())
'''
>>> Pen (1, 2, 0) Brush (8, 255)
'''
```

Тип результата str

property white_border
Светлая окантовка.

Тип результата bool

PointPictureStyle - Стиль со ссылкой на растровый файл

class axipy.da.**PointPictureStyle**(filename, color=PySide2.QtCore.Qt.GlobalColor.black, size=12, customstyle=0)

Базовые классы: axipy.da.PointStyle

Стиль со ссылкой на растровый файл.

Параметры

- **filename** (str) – Наименование растрового файла. Строка до 31 символа длиной. Данный файл должен находится в каталоге CustSymb с ресурсами. Например, “Arrow.BMP”.
- **color** (QColor) – Цвет символа.
- **size** (int) – Размер символа в в пунктах от 1 до 48.
- **customstyle** (int) – Задание дополнительных параметров стиля оформления.

Таблица 96: Возможные значения параметра customstyle:

Значение	Наименование
0	Флажки Фон и Покрасить одним цветом не установлены. Символ показывается стандартно. Все белые точки изображения становятся прозрачными и под ними видны объекты Карты.
1	Установлен флажок Фон; все белые точки изображения становятся непрозрачными.
2	Установлен флажок Покрасить одним цветом все не белые точки изображения красятся в цвет символа.
3	Установлены флажки Фон и Покрасить одним цветом.

Список 166: Пример.

```
fs = PointPictureStyle('AMBU1-32.bmp')
fs.color = Qt.red
fs.apply_color = True
fs.actual_size = True
fs.show_background = True
```

Attributes:

actual_size	Реальный размер.
apply_color	Применить цвет.
color	Цвет символа.
filename	Наименование файла растра.
show_background	Непрозрачный фон.
size	Размер символа в пунктах.

Methods:

clone()	Создаёт копию объекта стиля

continues on next page

Таблица 98 – продолжение с предыдущей страницы

create_mi_compat([symbol, color, pointSize])	Создание стиля в виде совместимого с MapInfo 3 PointCompatStyle.
create_mi_font([symbol, color, size, ...])	Создание стиля на базе шрифта True Type PointFontStyle.
create_mi_picture(filename[, color, size, ...])	Создание стиля с ссылкой на растровый файл PointPictureStyle.
draw(geometry, painter)	Рисует геометрический объект с текущим стилем в произвольном контексте вывода.
for_geometry(geom)	Возвращает стиль по умолчанию для переданного объекта.
from_mapinfo(mapbasic_string)	Получает стиль из строки формата MapBasic.
to_mapinfo()	Возвращает строковое представление в формате MapBasic.

property actual_size
Реальный размер.

Тип результата bool

property apply_color
Применить цвет.

Тип результата bool

clone()
Создаёт копию объекта стиля

Тип результата Style

property color
Цвет символа.

Тип результата QColor

static create_mi_compat(symbol=35, color=PySide2.QtCore.Qt.GlobalColor.red, pointSize=8)
Создание стиля в виде совместимого с MapInfo 3 PointCompatStyle.

Параметры

- **symbol** (int) – Номер символа, который будет отображен при отрисовке. Для создания невидимого символа используйте значение 31. Стандартный набор условных знаков включает символы от 31 до 67,
- **color** (QColor) – Цвет символа
- **pointSize** (int) – Целое число, размер символа в пунктах от 1 до 48.

Тип результата PointCompatStyle

static create_mi_font(symbol=36, color=PySide2.QtCore.Qt.GlobalColor.red, size=8, fontname='Axioma MI MapSymbols', fontstyle=0, rotation=0.0)
Создание стиля на базе шрифта True Type PointFontStyle.

Параметры

- **symbol** (int) – Целое, имеющее значение 31 или больше, определяющее, какой используется символ из шрифтов TrueType. Для создания невидимого символа используйте значение 31.
- **color** (QColor) – Цвет символа
- **pointSize** – Целое число, размер символа в пунктах от 1 до 48;
- **fontname** (str) – Строка с именем шрифта TrueType (например, значение по умолчание „Axioma MI MapSymbols“)
- **fontstyle** (int) – Стиль дополнительного оформления, например, курсивный текст. Возможные параметры см. в таблице ниже. Для указания нескольких параметров их суммируют между собой.
- **rotation** (float) – Угол поворота символа в градусах.

Тип результата PointFontStyle

static create_mi_picture(filename, color=PySide2.QtCore.Qt.GlobalColor.black, size=12, customstyle=0)

Создание стиля с ссылкой на растровый файл PointPictureStyle.

Параметры

- **filename** (str) – Наименование растрового файла. Строка до 31 символа длиной. Данный файл должен находится в каталоге CustSymb с ресурсами. Например, “Arrow.BMP”.
- **color** (QColor) – Цвет символа.
- **size** (int) – Размер символа в в пунктах от 1 до 48.
- **customstyle** (int) – Задание дополнительных параметров стиля оформления.

Тип результата PointPictureStyle

draw(geometry, painter)

Рисует геометрический объект с текущим стилем в произвольном контексте вывода. Это может быть востребовано при желании отрисовать геометрию со стилем на форме или диалоге.

Параметры

- **geometry** (Geometry) – Геометрия. Должна соответствовать стилю. Т.е. если объект полигон, а стиль для рисования точечных объектов, то ничего нарисовано не будет.
- **painter** (QPainter) – Контекст вывода.

Список 167: Пример отрисовки в растре и сохранение результата в файле.

```
image = QImage(100, 100, QImage.Format_ARGB32_Premultiplied)
image.fill(Qt.white)
painter = QPainter(image)
point = Point(50,50)
style = PointStyle.create_mi_font(42, Qt.red, 24)
style.draw(point, painter)
image.save(filename)
```

property filename
Наименование файла растра.

Тип результата str

classmethod for_geometry(geom)
Возвращает стиль по умолчанию для переданного объекта.

Параметры geom (Geometry) – Геометрический объект, для которого необходимо получить соответствующий ему стиль.

Тип результата Style

classmethod from_mapinfo(mapbasic_string)
Получает стиль из строки формата MapBasic.

Параметры mapbasic_string (str) – Строка в формате MapBasic.

```
style = Style.from_mapinfo("Pen (1, 2, 0) Brush (8, 255)")
```

Тип результата Style

property show_background
Непрозрачный фон.

Тип результата bool

property size
Размер символа в пунктах.

Тип результата int

to_mapinfo()
Возвращает строковое представление в формате MapBasic.

```
print(style.to_mapinfo())
'''
>>> Pen (1, 2, 0) Brush (8, 255)
'''
```

Тип результата str

LineStyle - Стиль линий

class axipy.da.**LineStyle**(pattern=2, color=PySide2.QtCore.Qt.GlobalColor.black, width=1)

Базовые классы: axipy.da.Style

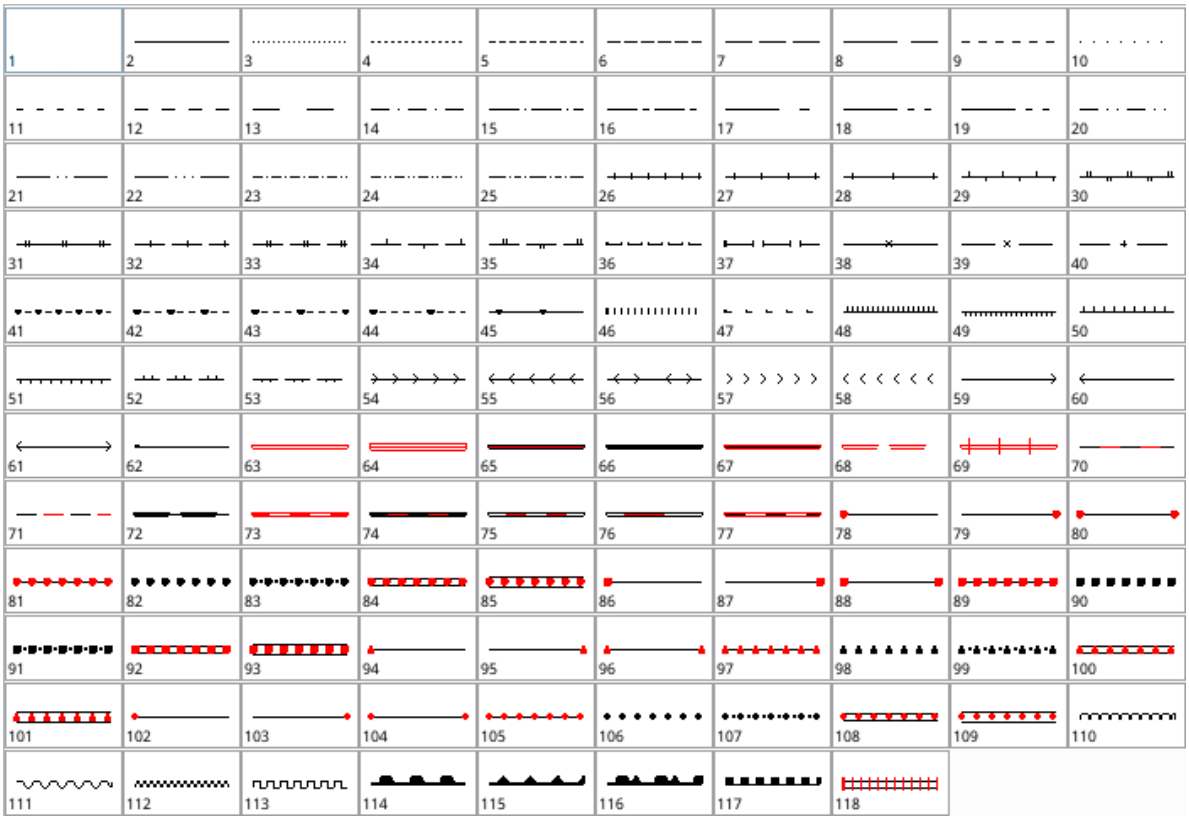
Стиль линейного объекта, совместимый с MapInfo.

Параметры

- **pattern** (int) – Тип линии. Типы линий обозначаются кодами от 1 до 118. Тип 1 представляет собой невидимую линию.
- **color** (QColor) – Цвет линии
- **width** (int) – Толщина линии. Задается числом от 0 до 7, при этом линия нулевой ширины невидима на экране. 11-2047 - это значения, которые могут быть преобразованы в пункты: ширина линии =

(число пунктов * 10) + 10 Значение 0 допустимо только для типа линии 1 или невидимых линий.

В системе доступны следующие стили линии:



Список 168: Пример.

```
style = LineStyle(3, Qt.red)
```

Methods:

clone()	Создаёт копию объекта стиля
draw(geometry, painter)	Рисует геометрический объект с текущим стилем в произвольном контексте вывода.
for_geometry(geom)	Возвращает стиль по умолчанию для переданного объекта.
from_mapinfo(mapbasic_string)	Получает стиль из строки формата MapBasic.
to_mapinfo()	Возвращает строковое представление в формате MapBasic.

Attributes:

color	Цвет линии.
pattern	Номер стиля линии.
width	Толщина линии.

clone()
Создаёт копию объекта стиля

Тип результата Style

property color
Цвет линии.

Тип результата QColor

draw(geometry, painter)
Рисует геометрический объект с текущим стилем в произвольном контексте вывода. Это может быть востребовано при желании отрисовать геометрию со стилем на форме или диалоге.

Параметры

- **geometry** (Geometry) – Геометрия. Должна соответствовать стилю. Т.е. если объект полигон, а стиль для рисования точечных объектов, то ничего нарисовано не будет.
- **painter** (QPainter) – Контекст вывода.

Список 169: Пример отрисовки в растре и сохранение результата в файле.

```
image = QImage(100, 100, QImage.Format_ARGB32_Premultiplied)
image.fill(Qt.white)
painter = QPainter(image)
point = Point(50,50)
style = PointStyle.create_mi_font(42, Qt.red, 24)
style.draw(point, painter)
image.save(filename)
```

classmethod for_geometry(geom)
Возвращает стиль по умолчанию для переданного объекта.

Параметры geom (Geometry) – Геометрический объект, для которого необходимо получить соответствующий ему стиль.

Тип результата Style

classmethod from_mapinfo(mapbasic_string)
Получает стиль из строки формата MapBasic.

Параметры mapbasic_string (str) – Строка в формате MapBasic.

```
style = Style.from_mapinfo("Pen (1, 2, 0) Brush (8, 255)")
```

Тип результата Style

property pattern
Номер стиля линии.

Тип результата int

to_mapinfo()
Возвращает строковое представление в формате MapBasic.

```
print(style.to_mapinfo())
'''
>>> Pen (1, 2, 0) Brush (8, 255)
'''
```

Тип результата str

property width
Толщина линии.

Тип результата int

FillStyle - Стиль заливки полигона

class axipy.da.**FillStyle**(pattern=1, color=PySide2.QtCore.Qt.GlobalColor.transparent)

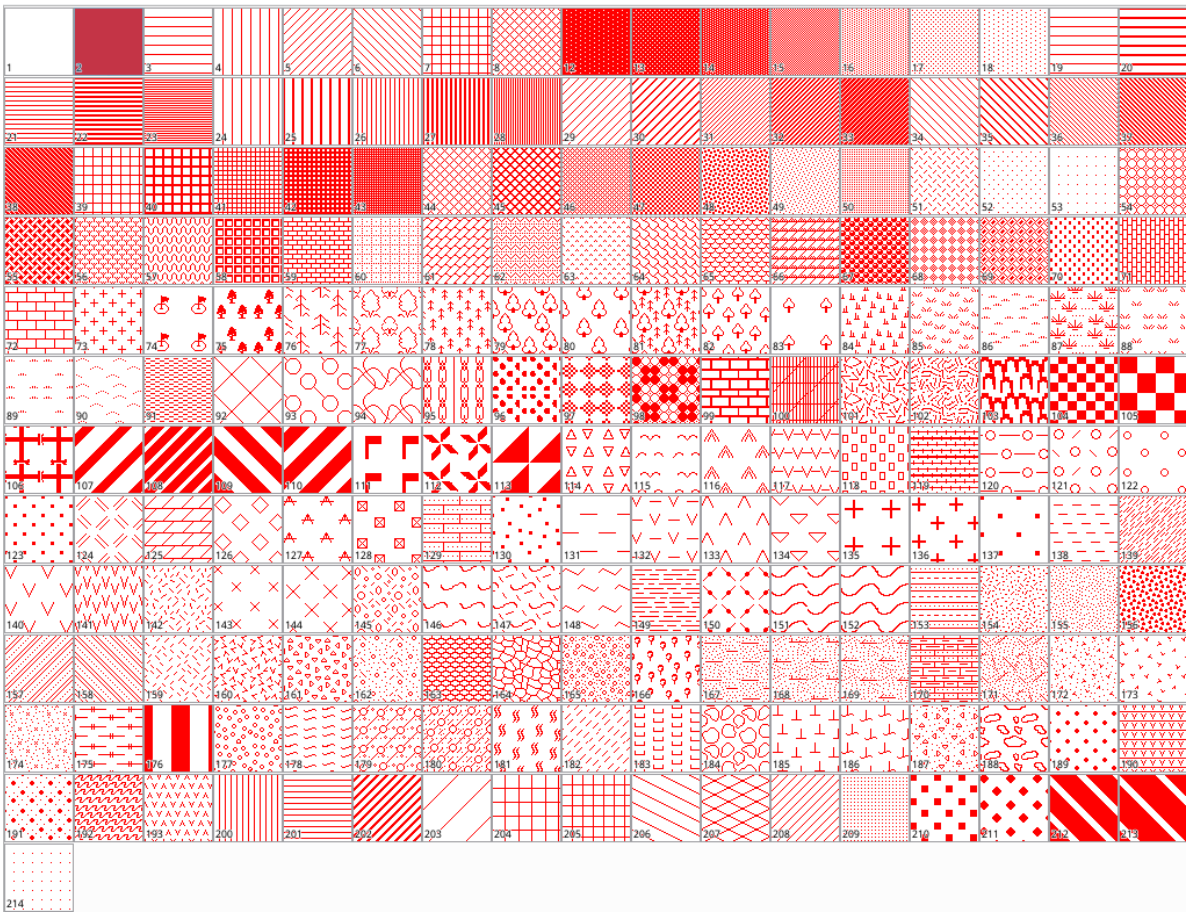
Базовые классы: axipy.da.Style

Стиль заливки полигонов PolygonStyle.

Параметры

- **pattern** (int) – Номер стиля заливки.
- **color** (QColor) – Цвет основной заливки.

В системе доступны следующие стили заливки:



Attributes:

bg_color	Цвет фона.
color	Цвет линии.
pattern	Номер стиля заливки.

Methods:

clone()	Создаёт копию объекта стиля
draw(geometry, painter)	Рисует геометрический объект с текущим стилем в произвольном контексте вывода.
for_geometry(geom)	Возвращает стиль по умолчанию для переданного объекта.
from_mapinfo(mapbasic_string)	Получает стиль из строки формата MapBasic.
to_mapinfo()	Возвращает строковое представление в формате MapBasic.

property bg_color
Цвет фона.

Тип результата QColor

clone()
Создаёт копию объекта стиля

Тип результата Style

property color
Цвет линии.

Тип результата QColor

draw(geometry, painter)
Рисует геометрический объект с текущим стилем в произвольном контексте вывода. Это может быть востребовано при желании отрисовать геометрию со стилем на форме или диалоге.

Параметры

- **geometry** (Geometry) – Геометрия. Должна соответствовать стилю. Т.е. если объект полигон, а стиль для рисования точечных объектов, то ничего нарисовано не будет.
- **painter** (QPainter) – Контекст вывода.

Список 170: Пример отрисовки в растре и сохранение результата в файле.

```
image = QImage(100, 100, QImage.Format_ARGB32_Premultiplied)
image.fill(Qt.white)
painter = QPainter(image)
point = Point(50,50)
style = PointStyle.create_mi_font(42, Qt.red, 24)
style.draw(point, painter)
image.save(filename)
```

classmethod for_geometry(geom)
Возвращает стиль по умолчанию для переданного объекта.

Параметры geom (Geometry) – Геометрический объект, для которого необходимо получить соответствующий ему стиль.

Тип результата Style

classmethod from_mapinfo(mapbasic_string)
Получает стиль из строки формата MapBasic.

Параметры mapbasic_string (str) – Строка в формате MapBasic.

```
style = Style.from_mapinfo("Pen (1, 2, 0) Brush (8, 255)")
```

Тип результата Style

property pattern
Номер стиля заливки.

Тип результата int

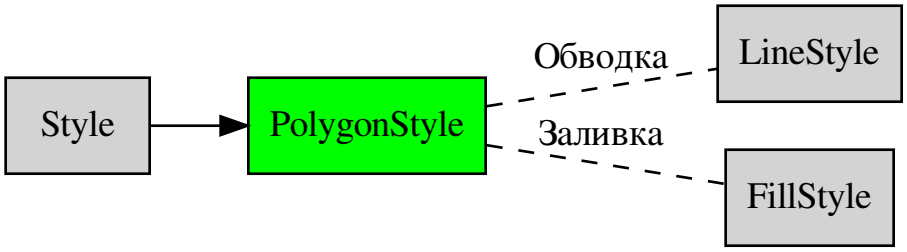
to_mapinfo()
Возвращает строковое представление в формате MapBasic.

```
print(style.to_mapinfo())
'''
>>> Pen (1, 2, 0) Brush (8, 255)
'''
```

Тип результата str

PolygonStyle - Стиль полигонов

Зависимости стиля площадного объекта:



class axipy.da.**PolygonStyle**(pattern=1, color=PySide2.QtCore.Qt.GlobalColor.white, pattern_pen=1)

Базовые классы: axipy.da.Style

Стиль площадного объекта. По умолчанию создается прозрачный стиль FillStyle с черной окантовкой LineStyle.

Список 171: Пример.

```
# Создадим стиль по умолчанию
plstyle = PolygonStyle()
print(plstyle.to_mapinfo())
# Назначим цвет заливки и фона заливки
plstyle.set_brush(color=Qt.green, bgColor=Qt.blue)
print(plstyle.to_mapinfo())
# Установим обводку
plstyle.set_pen(color=Qt.black)
print(plstyle.to_mapinfo())
# Установим параметры заливки через свойства
plstyle.fill.pattern = 5
plstyle.fill.color = Qt.blue
# Установим параметры обводки через свойства
plstyle.border.width = 4
plstyle.border.color = Qt.blue
'''
>>> Brush (1, 16777215)
>>> Brush (1, 65280, 255)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
>>> Pen (1, 2, 0)  Brush (1, 65280, 255)
'''
```

Параметры

- **pattern** (int) – Номер стиля заливки.
- **color** (QColor) – Цвет основной заливки.
- **pattern_pen** (int) – Цвет обводки. По умолчанию обводка отсутствует.

Attributes:

border	Стиль окантовки.
fill	Стиль заливки.

Methods:

clone()	Создаёт копию объекта стиля
draw(geometry, painter)	Рисует геометрический объект с текущим стилем в произвольном контексте вывода.
for_geometry(geom)	Возвращает стиль по умолчанию для переданного объекта.
from_mapinfo(mapbasic_string)	Получает стиль из строки формата MapBasic.
set_brush([pattern, color, bgColor])	Задание стиля заливки площадного объекта.
set_pen([pattern, color, width])	Задание стиля обводки.
to_mapinfo()	Возвращает строковое представление в формате MapBasic.

property border
Стиль окантовки. Может отсутствовать. Для переопределения можно также использовать set_pen()

Тип результата LineStyle

clone()
Создаёт копию объекта стиля

Тип результата Style

draw(geometry, painter)
Рисует геометрический объект с текущим стилем в произвольном контексте вывода. Это может быть востребовано при желании отрисовать геометрию со стилем на форме или диалоге.

Параметры

- **geometry** (Geometry) – Геометрия. Должна соответствовать стилю. Т.е. если объект полигон, а стиль для рисования точечных объектов, то ничего нарисовано не будет.
- **painter** (QPainter) – Контекст вывода.

Список 172: Пример отрисовки в растре и сохранение результата в файле.

```
image = QImage(100, 100, QImage.Format_ARGB32_Premultiplied)
image.fill(Qt.white)
painter = QPainter(image)
point = Point(50,50)
style = PointStyle.create_mi_font(42, Qt.red, 24)
style.draw(point, painter)
image.save(filename)
```

property fill
Стиль заливки.

Тип результата FillStyle

classmethod for_geometry(geom)
Возвращает стиль по умолчанию для переданного объекта.

Параметры geom (Geometry) – Геометрический объект, для которого необходимо получить соответствующий ему стиль.

Тип результата Style

classmethod from_mapinfo(mapbasic_string)
Получает стиль из строки формата MapBasic.

Параметры mapbasic_string (str) – Строка в формате MapBasic.

```
style = Style.from_mapinfo("Pen (1, 2, 0) Brush (8, 255)")
```

Тип результата Style

set_brush(pattern=1, color=PySide2.QtCore.Qt.GlobalColor.white, bgColor=PySide2.QtCore.Qt.GlobalColor.transparent)
Задание стиля заливки площадного объекта.

Параметры

- **pattern** (int) – Номер стиля заливки. Шаблон задается числом от 1 до 71, при этом в шаблоне с номером 1 оба цвета отсутствуют, а в шаблоне 2 отсутствует цвет фона. Шаблоны с кодами 9-11 зарезервированы для внутренних целей.
- **color** (QColor) – Цвет основной заливки.
- **bgColor** (QColor) – Цвет заднего фона, если заливка неполная.

Доступные стили заливки см. FillStyle:

set_pen(pattern=2, color=PySide2.QtCore.Qt.GlobalColor.black, width=1)
Задание стиля обводки. Параметры аналогичны при задании стиля линии LineStyle()

Параметры

- **pattern** (int) – Номер стиля линии.
- **color** (QColor) – Цвет линии
- **width** (int) – Толщина линии.

to_mapinfo()

Возвращает строковое представление в формате MapBasic.

```
print(style.to_mapinfo())
'''
>>> Pen (1, 2, 0) Brush (8, 255)
'''
```

Тип результата str

TextStyle - Стиль текста

class axipy.da.**TextStyle**(fontname, size, style=0, forecolor=PySide2.QtCore.Qt.GlobalColor.black, backcolor=PySide2.QtCore.Qt.GlobalColor.transparent)

Базовые классы: axipy.da.Style

Стиль текстового объекта.

Параметры

- **fontname** (str) – Наименование шрифта.
- **size** (int) – Размер шрифта в пунктах. Может принимать значение 0 для подписей в окне карты, так как они являются атрибутами карты и их размер определяется динамически.
- **style** (int) – Дополнительные параметры стиля. Подробнее см. в таблице ниже.
- **color** – Цвет шрифта
- **backcolor** (QColor) – Цвет заднего фона, если он задан.

Таблица 105: Возможные значения параметра style

Значение	Наименование
0	Обычный
1	Жирный
2	Курсив
4	Подчеркнутый
16	Контур (только для Macintosh)
32	Тень
256	Кайма
512	Капитель
1024	Разрядка

Attributes:

bg_color	Цвет фона текста
bg_type	Тип отрисовки фона текста.
color	Цвет текста
effects	Эффекты применяемые к текстовому объекту.
fontname	Шрифт

Methods:

clone()	Создаёт копию объекта стиля
draw(geometry, painter)	Рисует геометрический объект с текущим стилем в произвольном контексте вывода.
for_geometry(geom)	Возвращает стиль по умолчанию для переданного объекта.
from_mapinfo(mapbasic_string)	Получает стиль из строки формата MapBasic.
to_mapinfo()	Возвращает строковое представление в формате MapBasic.

property bg_color
Цвет фона текста

Тип результата QColor

property bg_type
Тип отрисовки фона текста.

Тип результата TextBackgroundType

clone()
Создаёт копию объекта стиля

Тип результата Style

property color
Цвет текста

Тип результата QColor

draw(geometry, painter)
Рисует геометрический объект с текущим стилем в произвольном контексте вывода. Это может быть востребовано при желании отрисовать геометрию со стилем на форме или диалоге.

Параметры

- **geometry** (Geometry) – Геометрия. Должна соответствовать стилю. Т.е. если объект полигон, а стиль для рисования точечных объектов, то ничего нарисовано не будет.
- **painter** (QPainter) – Контекст вывода.

Список 173: Пример отрисовки в растре и сохранение результата в файле.

```
image = QImage(100, 100, QImage.Format_ARGB32_Premultiplied)
image.fill(Qt.white)
painter = QPainter(image)
point = Point(50,50)
style = PointStyle.create_mi_font(42, Qt.red, 24)
style.draw(point, painter)
image.save(filename)
```

property effects
Эффекты применяемые к текстовому объекту.

Тип результата TextStyleEffects

property fontname
Шрифт

Тип результата str

classmethod for_geometry(geom)
Возвращает стиль по умолчанию для переданного объекта.

Параметры geom (Geometry) – Геометрический объект, для которого необходимо получить соответствующий ему стиль.

Тип результата Style

classmethod from_mapinfo(mapbasic_string)
Получает стиль из строки формата MapBasic.

Параметры mapbasic_string (str) – Строка в формате MapBasic.

```
style = Style.from_mapinfo("Pen (1, 2, 0) Brush (8, 255)")
```

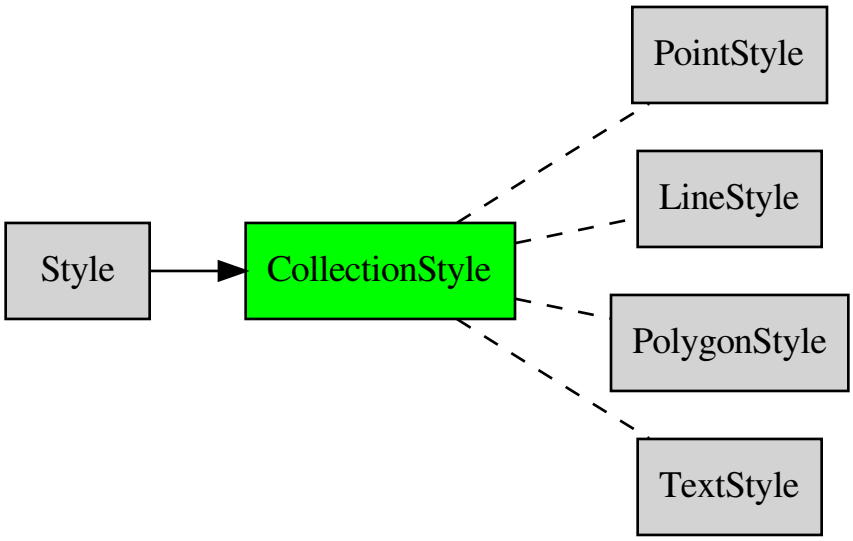
Тип результата Style

to_mapinfo()
Возвращает строковое представление в формате MapBasic.

```
print(style.to_mapinfo())
'''
>>> Pen (1, 2, 0) Brush (8, 255)
'''
```

Тип результата str

CollectionStyle - Стиль коллекций



class axipy.da.**CollectionStyle**

Базовые классы: axipy.da.Style

Смешанный стиль для разнородного типа объектов.

Данный стиль представляет собой контейнер стилей. может применяться в купе с геометрическим объектом типа разнородная коллекция axipy.da.GeometryCollection. Для задания или переопределения стилей простейших объектов, необходимо вызывать соответствующие методы для необходимых типов объектов.

Примечание: Объекты стилей, полученные через методы line(), polygon() и т.д. будут удалены сразу же после удаления объекта стиля коллекции. Если их нужно сохранить, воспользуйтесь операцией clone().

Methods:

clone()	Создаёт копию объекта стиля
draw(geometry, painter)	Рисует геометрический объект с текущим стилем в произвольном контексте вывода.
find_style(geom)	Пытаемся найти стиль подходящий для переданной геометрии
for_geometry(geom)	Возвращает стиль по умолчанию для переданного объекта.

continues on next page

Таблица 108 – продолжение с предыдущей страницы

for_line(style)	Задание стиля для линейных объектов LineStyle.
for_point(style)	Задание стиля для точечных объектов PointStyle.
for_polygon(style)	Задание стиля для полигональных объектов PolygonStyle.
for_text(style)	Задание стиля для текстовых объектов TextStyle.
from_mapinfo(mapbasic_string)	Получает стиль из строки формата MapBasic.
to_mapinfo()	Возвращает строковое представление в формате MapBasic.

Attributes:

line	Стиль для линейных объектов LineStyle.
point	Стиль для точечных объектов PointStyle.
polygon	Стиль для полигональных объектов PolygonStyle.
text	Стиль для текстовых объектов TextStyle.

clone()
Создаёт копию объекта стиля

Тип результата Style

draw(geometry, painter)
Рисует геометрический объект с текущим стилем в произвольном контексте вывода. Это может быть востребовано при желании отрисовать геометрию со стилем на форме или диалоге.

Параметры

- **geometry** (Geometry) – Геометрия. Должна соответствовать стилю. Т.е. если объект полигон, а стиль для рисования точечных объектов, то ничего нарисовано не будет.
- **painter** (QPainter) – Контекст вывода.

Список 174: Пример отрисовки в растре и сохранение результата в файле.

```
image = QImage(100, 100, QImage.Format_ARGB32_Premultiplied)
image.fill(Qt.white)
painter = QPainter(image)
point = Point(50,50)
style = PointStyle.create_mi_font(42, Qt.red, 24)
style.draw(point, painter)
image.save(filename)
```

find_style(geom)
Пытаемся найти стиль подходящий для переданной геометрии

Тип результата Style

classmethod for_geometry(geom)
Возвращает стиль по умолчанию для переданного объекта.

Параметры geom (Geometry) – Геометрический объект, для которого необходимо получить соответствующий ему стиль.

Тип результата Style

for_line(style)
Задание стиля для линейных объектов LineStyle.

for_point(style)
Задание стиля для точечных объектов PointStyle.

for_polygon(style)
Задание стиля для полигональных объектов PolygonStyle.

for_text(style)
Задание стиля для текстовых объектов TextStyle.

classmethod from_mapinfo(mapbasic_string)
Получает стиль из строки формата MapBasic.

Параметры mapbasic_string (str) – Строка в формате MapBasic.

```
style = Style.from_mapinfo("Pen (1, 2, 0) Brush (8, 255)")
```

Тип результата Style

property line
Стиль для линейных объектов LineStyle.

Тип результата Optional[LineStyle]

property point
Стиль для точечных объектов PointStyle.

Тип результата Optional[PointStyle]

property polygon
Стиль для полигональных объектов PolygonStyle.

Тип результата Optional[PolygonStyle]

property text
Стиль для текстовых объектов TextStyle.

Тип результата Optional[TextStyle]

to_mapinfo()
Возвращает строковое представление в формате MapBasic.

```
print(style.to_mapinfo())
'''
>>> Pen (1, 2, 0) Brush (8, 255)
'''
```

Тип результата str

16.1.6.17 axipy.da

Модуль источников данных.

В данном модуле содержатся классы и методы для работы с источниками данных.

axipy.da.**provider_manager**
Готовый экземпляр открытия/создания объектов данных.

Type axipy.da.ProviderManager

axipy.da.**state_manager**
Готовый экземпляр наблюдателей за состоянием.

Type axipy.da.StateManager

axipy.da.**data_manager**
Хранилище объектов приложения.

Это то же хранилище, которое отображается в панели «Открытые данные».

Примечание: При открытии объектов данных `axipy.da.ProviderManager.openfile()` они автоматически попадают в каталог.

Type axipy.da.DataManager

class axipy.da.**DefaultKeys**
Идентификаторы наблюдателей по умолчанию.

Таблица 110: Атрибуты

Значение	Тип	Наименование
Selection	bool	Есть выборка
Editable	bool	Активная карта имеет редактируемый слой
SelectionEditable	bool	Карта имеет редактируемый слой и есть выделенные объекты на одном из слоев карты
SelectionEditableIsSame	bool	Карта имеет редактируемый слой и выборку на этом слое
Widget	bool	Есть активное окно
MapView	bool	Есть активное окно карты
TableView	bool	Есть активное окно таблицы
HasTables	bool	Открыта хотя бы одна таблица

class axipy.da.**ValueObserver**
Наблюдатель за одним значением.

value()
Возвращает значение.

```
# Эквивалентно
v = obs.value()
v = obs()
```

Тип результата typing.Any

setValue(value)
Устанавливает значение.

При изменении значения испускается сигнал changed.

Параметры **value** (typing.Any) – Новое значение.

property **changed**(value)
Сигнал об изменении значения.

Параметры **value** (typing.Any) – Новое значение.

Тип результата PySide2.QtCore.Signal

```
def print_func(value):
    print(value)

observer.changed.connect(print_func)
```

ProviderManager - Объект открытия/создания данных

class axipy.da.**ProviderManager**
Класс открытия/создания объектов данных DataProvider.

Примечание: Используйте готовый экземпляр этого класса axipy.da.provider_manager.

Примечание: Для удобного задания параметров используйте экземпляры провайдеров: tab, shp, csv, mif, excel, sqlite, postgre, oracle, mssql, ogr, svg, gdal, rest, tms, wms, wmts.

Примечание: Открытые данные автоматически попадают в хранилище данных axipy.da.DataManager.

Пример открытия локальной таблицы:

```
table = provider_manager.openfile('../path/to/datadir/table.tab')
```

Methods:

create(definition)	Создает и открывает данные из описания.
create_open(definition)	Создает и открывает данные из описания.
createfile(filepath, schema, *args, **kwargs)	Создает таблицу.
loaded_providers()	Возвращает список всех загруженных провайдеров данных.
open(definition)	Открывает данные по описанию.
open_hidden(definition)	Открывает данные по описанию.
openfile(filepath, *args, **kwargs)	Открывает данные из файла.
providers()	Возвращает список всех загруженных провайдеров данных.

continues on next page

Таблица 111 – продолжение с предыдущей страницы

query(query_text, *tables)	Выполняет SQL-запрос к перечисленным таблицам.
read_contents(definition)	Читает содержимое источника данных.

Attributes:

csv	Файловый провайдер - Текст с разделителями.
excel	Провайдер чтения файлов Excel.
gdal	Растровый провайдер GDAL.
mif	Провайдер данных MIF-MID.
mssql	Провайдер для базы данных MSSQLServer.
ogr	Векторный провайдер OGR.
oracle	Провайдер для базы данных Oracle.
postgre	Провайдер для базы данных PostgreSQL.
rest	Провайдер REST.
shp	Векторный провайдер SHP.
sqlite	Векторный провайдер sqlite.
svg	Провайдер для SVG.
tab	Провайдер MapInfo.
tms	Тайловый провайдер.
wms	Web Map Service.
wmts	Web Map Tile Service.

create(definition)
Создает и открывает данные из описания.

Параметры definition (dict) – Описание объекта данных.

Псевдоним create_open().

Тип результата DataObject

create_open(definition)
Создает и открывает данные из описания.

Возможные параметры:

- src - Строка, определяющая местоположение источника данных. Это может быть либо путь к файлу с расширением TAB, либо пустая строка (для таблицы, размещаемой в памяти).
- schema - Схема таблицы. Задается массивом объектов, содержащих атрибуты.
- hidden - Если указано True, то созданный объект не будет зарегистрирован в каталоге. См. также open_hidden()

Параметры definition (dict) – Описание объекта данных.

Пример:

```
definition = {
    'src': '../path/to/datadir/edit/table.tab',
    'schema': attr.schema(
        attr.string('field1'),
        attr.integer('field2'),
    ),
}
table = provider_manager.create(definition)
```

Тип результата DataObject

createfile(filepath, schema, *args, **kwargs)
Создает таблицу.

create() выполняет ту же функцию, но в более обобщенном виде.

Параметры

- **filepath** (str) – Путь к создаваемой таблице.
- **schema** – Схема таблицы.

Тип результата DataObject

property csv
Файловый провайдер - Текст с разделителями.

Тип результата CsvDataProvider

property excel
Провайдер чтения файлов Excel.

Тип результата ExcelDataProvider

property gdal
Растровый провайдер GDAL.

Тип результата GdalDataProvider

loaded_providers()
Возвращает список всех загруженных провайдеров данных.

Тип результата dict

Результат Провайдеры в виде пар (Идентификатор : Описание).

property mif
Провайдер данных MIF-MID.

Тип результата MifMidDataProvider

property mssql
Провайдер для базы данных MSSQLServer.

Тип результата MsSqlDataProvider

property ogr
Векторный провайдер OGR.

Тип результата OgrDataProvider

open(definition)
Открывает данные по описанию.

Формат описания объектов данных (набор и тип параметров) индивидуален для каждого провайдера данных, однако многие элементы используются для всех провайдеров данных. В нижеприведенной таблице можно определить какие параметр можно указывать при открытии того или иного источника. Т.е. допустимые параметры для конкретного провайдера указаны в соответствующем методе open для этого провайдера.

Таблица 113: Доступные провайдеры данных и ссылки на дополнительные параметры:

Провайдер	Краткое описание	Ссылка
tab	Провайдер MapInfo	axipy.da.TabDataProvider.open()
csv	Текст с разделителями	axipy.da.CsvDataProvider.open()
ogr	Векторный провайдер OGR	axipy.da.OgrDataProvider.open()
excel	Провайдер чтения файлов Excel	axipy.da.ExcelDataProvider.open()
shp	Векторный провайдер SHP	axipy.da.ShapeDataProvider.open()
sqlite	Векторный провайдер sqlite	axipy.da.SqliteDataProvider.open()
svg	Провайдер для SVG	axipy.da.SvgDataProvider.open()
gdal	Растровый провайдер GDAL	axipy.da.GdalDataProvider.open()
postgre	Провайдер для базы данных PostgreSQL	axipy.da.PostgreDataProvider.open()
oracle	Провайдер для базы данных Oracle	axipy.da.OracleDataProvider.open()
mssql	Провайдер для базы данных MSSQLServer	axipy.da.MsSqlDataProvider.open()
rest	Провайдер REST	axipy.da.RestDataProvider.open()
tms	Тайловый провайдер	axipy.da.TmsDataProvider.open()
wms	Web Map Service	axipy.da.WmsDataProvider.open()
wmts	Web Map Tile Service	axipy.da.WmtsDataProvider.open()

Также существуют параметры, которые допустимы независимо от типа провайдера

Таблица 114: Доступные провайдеры данных и ссылки на дополнительные параметры:

Параметр	Краткое описание
provider	Используемый провайдер. Допустимые значения можно получить `loaded_providers()`. Если не задан, то система пытается его определить самостоятельно.
src	Ссылка на источник. Как правило, это имя файла. Для конкретного провайдера может дублироваться под другим именем.
dataobject	Если источник содержит несколько таблиц, то имя конкретного указывается через данный параметр
alias	Псевдоним для открываемого источника данных. В системе открытыйобъект будет доступен по этому имени

Параметры **definition** (`dict`) – Описание объекта данных.

Пример открытия файла (аналогичен `openfile()`):

```
json = {'src':'../path/to/datadir/world.tab'}
table_world = provider_manager.open(json)
```

или, что тоже самое:

```
json = {'filepath':'../path/to/datadir/world.tab'}
table_world = provider_manager.open(json)
```

Пример открытия файла с несколькими таблицами:

```
# Пример открытия GPKG файла::
definition = { 'src': '../path/to/datadir/example.gpkg',
               'dataobject': 'tablename' }
table = provider_manager.open(definition)
```

Пример открытия таблицы базы данных:

```
definition = {"host": "localhost",
              "db": "sample",
              "user": "postgres",
              "password": "postgres",
              "dataobject": "public.world",
              "provider": "PgDataProvider"}
table = provider_manager.open(definition)
```

Тип результата DataObject

open_hidden(definition)
Открывает данные по описанию. Аналогична функции open() за исключением того, что когда данный объект добавляется в каталог, он не учитывается в общем списке и от него из этого каталога не приходят события.

Примечание: См. также open()

Параметры **definition** (dict) – Описание объекта данных.

Пример:

```
table = axipy.provider_manager.open_hidden({'src':'world.tab'})
print(len(axipy.data_manager), axipy.data_manager.exists(table))
axipy.data_manager.remove(table)
>>> 0 True
```

Тип результата DataObject

openfile(filepath, *args, **kwargs)
Открывает данные из файла.

Параметры

- **filepath** (str) – Путь к открываемому файлу.
- ****kwargs** – Именованные аргументы. Возможные варианты от провайдера. Подробнее см. open()

Пример:

```
table = provider_manager.openfile('../path/to/datadir/example.gpkg')
```

Тип результата DataObject

property oracle
Провайдер для базы данных Oracle.

Тип результата OracleDataProvider

property postgre
Провайдер для базы данных PostgreSQL.

Тип результата PostgreDataProvider

providers()
Возвращает список всех загруженных провайдеров данных.

Тип результата List[DataProvider]

Результат Список провайдеров.

query(query_text, *tables)
Выполняет SQL-запрос к перечисленным таблицам.

Предупреждение: Используйте axipy.da.DataManager.query().

Параметры

- **query_text** (str) – Текст запроса.
- ***tables** – Список таблиц, к которым выполняется запрос.

Тип результата Table

Результат Таблица, если результатом запроса является таблица.

Пример:

```
query_text = "SELECT * FROM world, caps WHERE world.capital = caps.capital"
joined = provider_manager.query(query_text, world, caps)
```

read_contents(definition)

Читает содержимое источника данных.

Обычно используется для источников, способных содержать несколько объектов данных.

Параметры definition (Union[dict, str]) – Описание источника данных.

Тип результата List[str]

Результат Имена объектов данных.

Пример:

```
contents = axipy.provider_manager.read_contents('../path/to/datadir/example.
↪gpkg')
print(contents)
>>> ['world', 'worldcap']

world = axipy.provider_manager.openfile('../path/to/datadir/example.gpkg',␣
↪dataobject='world')
```

property rest

Провайдер REST.

Тип результата RestDataProvider

property shp

Векторный провайдер SHP.

Тип результата ShapeDataProvider

property sqlite

Векторный провайдер sqlite.

Тип результата SqliteDataProvider

property svg

Провайдер для SVG.

Тип результата SvgDataProvider

property tab

Провайдер MapInfo.

Тип результата TabDataProvider

property tms
Тайловый провайдер.

Тип результата TmsDataProvider

property wms
Web Map Service.

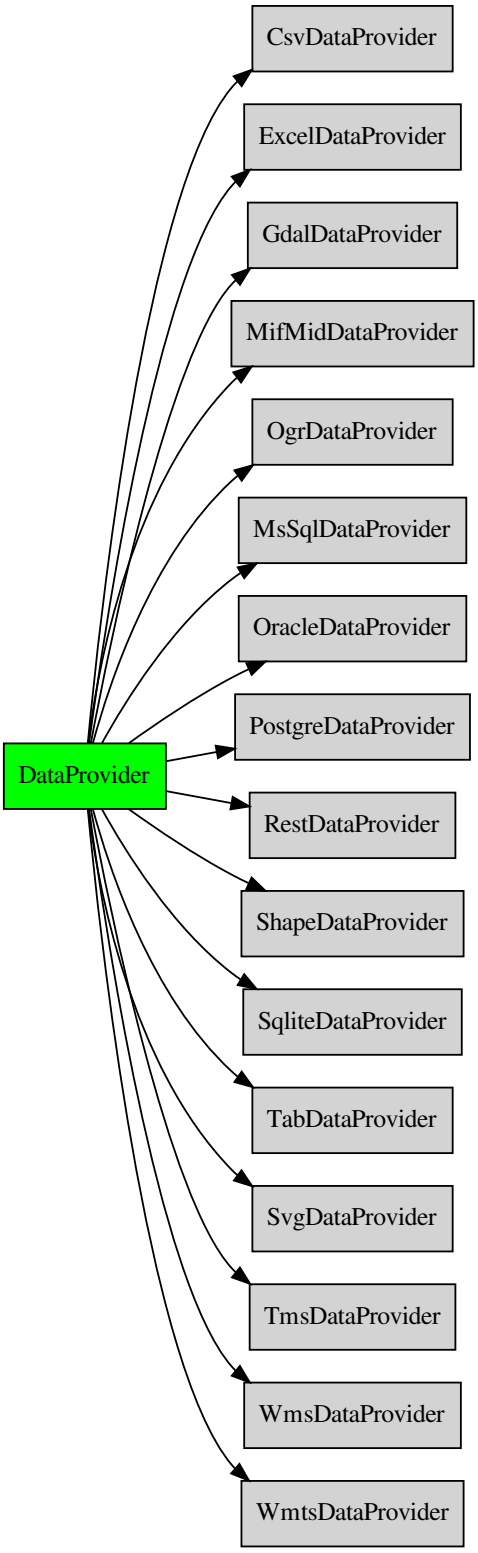
Тип результата WmsDataProvider

property wmts
Web Map Tile Service.

Тип результата WmtsDataProvider

DataProvider - Провайдер данных

Иерархия классов провайдера данных:



class axipy.da.**DataProvider**(info)
Абстрактный провайдер данных.

Methods:

create_open(*args, **kwargs)	Создает и открывает объект данных.
file_extensions()	Список поддерживаемых расширений файлов.
get_destination()	Создает назначение объекта данных.
get_source()	Создает источник данных.
open(*args, **kwargs)	Открывает объект данных.

Attributes:

id	Идентификатор провайдера.

create_open(*args, **kwargs)
Создает и открывает объект данных.

Пример:

```
provider.create_open(...)
```

Что эквивалентно:

```
provider.get_destiantion(...).create_open()
```

См.также:

DataProvider.destination().

file_extensions()
Список поддерживаемых расширений файлов.

Тип результата List[str]

Результат Пустой список для не файловых провайдеров.

get_destination()
Создает назначение объекта данных.

Исключение **NotImplementedError** – Если провайдер не поддерживает создание назначений.

Тип результата Destination

get_source()
Создает источник данных.

Исключение **NotImplementedError** – Если провайдер не поддерживает создание источников.

Тип результата Source

property id
Идентификатор провайдера.

Тип результата str

open(*args, **kwargs)
Открывает объект данных.

Пример:

```
provider.open(...)
```

Что эквивалентно:

```
provider.get_source(...).open()
```

См.также:

`DataProvider.source()`.

Source - Источник данных

class axipy.da.**Source**(*args)

Источник данных.

Используется для открытия данных или для указания источника при конвертации.

Пример открытия:

```
table = source.open()
```

Пример конвертации:

```
destination.export_from(source)
```

Примечание: Не все провайдеры поддерживают открытие и конвертацию. См. описание конкретного провайдера данных.

Methods:

open()	Открывает объект данных.

open()

Открывает объект данных.

Тип результата DataObject

Destination - Назначение объекта данных

class axipy.da.**Destination**(schema, *args)

Назначение объекта данных.

Используется для создания данных или для указания назначения при конвертации.

Пример создания:

```
table = destination.create_open()
```

Пример конвертации:

```
destination.export_from(source)
```

Примечание: Не все провайдеры поддерживают создание и конвертацию. См. описание конкретного провайдера данных.

Methods:

create_open()	Создает и открывает объект данных.
export(features)	Создает объект данных и экспортирует в него записи.
export_from(source[, copy_schema])	Создает объект данных и экспортирует в него записи из источника данных.
export_from_table(table[, copy_schema])	Создает объект данных и экспортирует в него записи из таблицы.

create_open()
Создает и открывает объект данных.

Тип результата DataObject

export(features)
Создает объект данных и экспортирует в него записи.

Параметры **features** (Iterator[Feature]) – Записи.

Список 175: Пример экспорта данных

```
# Определяем схему будущей таблицы
schema = Schema(Attribute.string('name', 30), coordsystem="prj:1,104")
# Формируем данные для вставки. В нашем случае одна точка
features = [Feature(name='hello', geometry=Point(10,10))]
# Имя выходного файла
filepath = './path/to/world_out.tab'
# Создаем таблиwe по определенной ранее информации
dest = provider_manager.tab.get_destination(filepath, schema)
# Непосредственно производим экспорт
dest.export(features)
```

export_from(source, copy_schema=False)
Создает объект данных и экспортирует в него записи из источника данных.

Параметры

- **source** (Source) – Источник данных.
- **copy_schema** (bool) – Копировать схему источника без изменений.

export_from_table(table, copy_schema=False)
Создает объект данных и экспортирует в него записи из таблицы.

Параметры

- **table** (Table) – Таблица.
- **copy_schema** (bool) – Копировать схему источника без изменений.

Список 176: Пример экспорта таблицы

```
# Открываем исходную таблицу
table_src = provider_manager.openfile(input_filepath)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
# Формируем целевую и производим экспорт
destination = provider_manager.csv.get_destination(output_filepath, Schema())
destination.export_from_table(table_src, copy_schema=True)
```

CsvDataProvider - Текст с разделителями

class axipy.da.**CsvDataProvider**(info)

Базовые классы: axipy.da.DataProvider

Файловый провайдер: Текст с разделителями.

Примечание: Ссылку на провайдер можно получить через глобальную переменную `axipy.da.provider_manager.csv`.

Methods:

create_open(filepath, schema[, with_header, ...])	Создает и открывает объект данных.
file_extensions()	Список поддерживаемых расширений файлов.
get_destination(filepath, schema[, ...])	Создает назначение объекта данных.
get_source(filepath[, with_header, ...])	Создает источник данных.
open(filepath[, with_header, delimiter, ...])	Открывает объект данных.

Attributes:

id	Идентификатор провайдера.

create_open(filepath, schema, with_header=True, delimiter=';', encoding='utf8')

Создает и открывает объект данных.

Параметры

- **filepath** (str) – Путь к файлу.
- **schema** (Schema) – Схема таблицы.
- **with_header** (bool) – Признак того, что в первой строке содержатся имена атрибутов таблицы.
- **delimiter** (str) – Разделитель полей.
- **encoding** (str) – Кодировка.

Тип результата Table

file_extensions()

Список поддерживаемых расширений файлов.

Тип результата List[str]

Результат Пустой список для не файловых провайдеров.

get_destination(filepath, schema, with_header=True, delimiter=';', encoding='utf8')
Создает назначение объекта данных.

Параметры

- **filepath** (str) – Путь к файлу.
- **schema** (Schema) – Схема таблицы.
- **with_header** (bool) – Признак того, что в первой строке содержатся имена атрибутов таблицы.
- **delimiter** (str) – Разделитель полей.
- **encoding** (str) – Кодировка.

Тип результата Destination

get_source(filepath, with_header=True, delimiter=',', encoding='utf8', alias=None)
Создает источник данных.

Параметры

- **filepath** (str) – Путь к файлу.
- **with_header** (bool) – Признак того, что в первой строке содержатся имена атрибутов таблицы.
- **delimiter** (str) – Разделитель полей.
- **encoding** (str) – Кодировка.

Тип результата Source

property id
Идентификатор провайдера.

Тип результата str

open(filepath, with_header=True, delimiter=',', encoding='utf8', alias=None)
Открывает объект данных.

Параметры

- **filepath** (str) – Путь к файлу.
- **with_header** (bool) – Признак того, что в первой строке содержатся имена атрибутов таблицы.
- **delimiter** (str) – Разделитель полей.
- **encoding** (str) – Кодировка.
- **alias** (Optional[str]) – Псевдоним для открываемой таблицы.

Тип результата Table

ExcelDataProvider - Провайдер чтения файлов Excel

class axipy.da.**ExcelDataProvider**(info)

Базовые классы: axipy.da.DataProvider

Провайдер чтения файлов Excel.

Примечание: Ссылку на провайдер можно получить через глобальную переменную `axipy.da.provider_manager.excel`.

Methods:

create_open(filepath, schema)	Создает и открывает объект данных.
file_extensions()	Список поддерживаемых расширений файлов.
get_destination(filepath, schema)	Создает назначение объекта данных.
get_source(filepath[, page, with_header, ...])	Создает источник данных.
open(filepath[, page, with_header, ...])	Открывает объект данных.

Attributes:

id	Идентификатор провайдера.

create_open(filepath, schema)
Создает и открывает объект данных.

Параметры

- **filepath** (str) – Путь к файлу.
- **schema** (Schema) – Схема таблицы.

Тип результата Table

file_extensions()
Список поддерживаемых расширений файлов.

Тип результата List[str]

Результат Пустой список для не файловых провайдеров.

get_destination(filepath, schema)
Создает назначение объекта данных.

Параметры

- **filepath** (str) – Путь к файлу.
- **schema** (Schema) – Схема таблицы.

Тип результата Destination

get_source(filepath, page=None, with_header=False, encoding='utf8', alias=None)
Создает источник данных.

Параметры

- **filepath** (str) – Путь к файлу.
- **page** (Optional[str]) – Имя страницы. Если не указана, то берется первая.
- **with_header** (bool) – Признак того, что в первой строке содержатся имена атрибутов таблицы.
- **encoding** (str) – Кодировка.

Тип результата Source

property id
Идентификатор провайдера.

Тип результата str

open(filepath, page=None, with_header=False, encoding='utf8', alias=None)
Открывает объект данных.

Параметры

- **filepath** (str) – Путь к файлу.
- **page** (Optional[str]) – Имя страницы.
- **with_header** (bool) – Признак того, что в первой строке содержатся имена атрибутов таблицы.
- **encoding** (str) – Кодировка.
- **alias** (Optional[str]) – Псевдоним для открываемой таблицы.

Тип результата Table

MifMidDataProvider -

class axipy.da.**MifMidDataProvider**(info)

Базовые классы: axipy.da.DataProvider

Провайдер данных MIF-MID.

Примечание: Поддерживает экспорт только в TAB. См. convert_to_tab().

Примечание: Ссылку на провайдер можно получить через глобальную переменную axipy.da.provider_manager.mif.

Methods:

convert_to_tab(mif_filepath, tab_filepath)	Конвертирует из MIF в TAB.

continues on next page

Таблица 123 – продолжение с предыдущей страницы

create_open()	**Внимание:** Не поддерживается.
file_extensions()	Список поддерживаемых расширений файлов.
get_destination(filepath, schema)	Создает назначение объекта данных.
get_source()	**Внимание:** Не поддерживается.
open()	**Внимание:** Не поддерживается.

Attributes:

id	Идентификатор провайдера.

convert_to_tab(mif_filepath, tab_filepath)

Конвертирует из MIF в TAB.

Список 177: Пример экспорта

```
# Исходный файл MIF
mif_filepath = './path/to/world.mif'
# Целевой файл TAB
tab_filepath = './path/to/world_out.tab'
# Преобразвание MIF в TAB
provider_manager.mif.convert_to_tab(mif_filepath, tab_filepath)
```

Параметры

- **mif_filepath** (str) – Путь к исходному файлу.
- **tab_filepath** (str) – Путь к выходному файлу.

Исключение **Exception** – Если при конвертации произошла ошибка.

create_open()

> **Внимание:** Не поддерживается.

Исключение NotImplementedError –

file_extensions()
Список поддерживаемых расширений файлов.

Тип результата List[str]

Результат Пустой список для не файловых провайдеров.

get_destination(filepath, schema)
Создает назначение объекта данных.

Параметры

- **filepath** (str) – Путь к файлу.
- **schema** (Schema) – Схема таблицы.

Тип результата Destination

get_source()

> **Внимание:** Не поддерживается.

Исключение NotImplementedError –

Тип результата Source

property id
Идентификатор провайдера.

Тип результата str

open()

> **Внимание:** Не поддерживается.

Исключение NotImplementedError –

ShapeDataProvider - Векторный провайдер SHP

class axipy.da.**ShapeDataProvider**(info)

Базовые классы: axipy.da.DataProvider

Векторный провайдер SHP.

Примечание: Ссылку на провайдер можно получить через глобальную переменную `axipy.da.provider_manager.shp`.

Methods:

create_open(filepath, schema[, encoding])	Создает и открывает объект данных.
file_extensions()	Список поддерживаемых расширений файлов.
get_destination(filepath, schema[, encoding])	Создает назначение объекта данных.
get_source(filepath[, encoding, prj, alias])	Создает источник данных.
open(filepath[, encoding, prj, alias])	Открывает объект данных.
open_temporary(schema)	Создает и открывает временную таблицу.

Attributes:

id	Идентификатор провайдера.

create_open(filepath, schema, encoding='utf8')
Создает и открывает объект данных.

Параметры

- **filepath** (str) – Путь к файлу.
- **schema** (Schema) – Схема таблицы.
- **encoding** (str) – Кодировка.

Тип результата Table

file_extensions()
Список поддерживаемых расширений файлов.

Тип результата List[str]

Результат Пустой список для не файловых провайдеров.

get_destination(filepath, schema, encoding='utf8')
Создает назначение объекта данных.

Параметры

- **filepath** (str) – Путь к файлу.
- **schema** (Schema) – Схема таблицы.
- **encoding** (str) – Кодировка.

Тип результата Destination

get_source(filepath, encoding='utf8', prj=None, alias=None)
Создает источник данных.

Параметры

- **filepath** (str) – Путь к файлу.
- **encoding** (str) – Кодировка.
- **prj** (Optional[str]) – Строка Системы Координат.

Тип результата Source

property id
Идентификатор провайдера.

Тип результата str

open(filepath, encoding='utf8', prj=None, alias=None)
Открывает объект данных.

Пример:

```
shp = provider_manager.shp.open('world.shp', prj='1, 104')
```

Параметры

- **filepath** (str) – Путь к файлу.
- **encoding** (str) – Кодировка.
- **prj** (Optional[str]) – Строка Системы Координат.
- **alias** (Optional[str]) – Псевдоним для открываемой таблицы.

Тип результата Table

open_temporary(schema)
Создает и открывает временную таблицу.

Параметры **schema** (Schema) – Схема таблицы.

Тип результата Table

SqliteDataProvider - Векторный провайдер sqlite

class axipy.da.**SqliteDataProvider**(info)

Базовые классы: axipy.da.DataProvider

Векторный провайдер sqlite.

Примечание: Ссылку на провайдер можно получить через глобальную переменную axipy.da.provider_manager.sqlite.

Methods:

create_open()	**Внимание:** Не поддерживается.
file_extensions()	Список поддерживаемых расширений файлов.

continues on next page

Таблица 127 – продолжение с предыдущей страницы

get_destination()	**Внимание:** Не поддерживается.
get_source(filepath[, dataobject, sql, prj, ...])	Создает источник данных.
open(filepath[, dataobject, sql, prj, alias])	Открывает объект данных.

Attributes:

id	Идентификатор провайдера.

create_open()

Внимание: Не поддерживается.

Исключение NotImplementedError –

file_extensions()
Список поддерживаемых расширений файлов.

Тип результата List[str]

Результат Пустой список для не файловых провайдеров.

get_destination()

Внимание: Не поддерживается.

Исключение NotImplementedError –

get_source(filepath, dataobject=None, sql=None, prj=None, alias=None)
Создает источник данных. В качестве объекта может быть указана либо таблица, либо текст запроса. Если указан sql, то он имеет более высокий приоритет по отношению к значению dataobject. Если оба параметра опущены, будет возвращен None.

Параметры

- **filepath** (str) – Путь к файлу.
- **dataobject** (Optional[str]) – Имя таблицы.
- **sql** (Optional[str]) – SQL-запрос.
- **prj** (Optional[str]) – Строка Системы Координат.

Пример с таблицей:

```
table = provider_manager.openfile('world.sqlite', dataobject='world')
```

Пример с запросом и переопределенной СК:

```
table = provider_manager.openfile('world.sqlite', sql="select * from world␣
↪where Страна like 'P%'", prj='12, 104, "m", 0')
```

Тип результата Source

property id
Идентификатор провайдера.

Тип результата str

open(filepath, dataobject=None, sql=None, prj=None, alias=None)
Открывает объект данных.

В качестве объекта может быть указана либо таблица, либо текст запроса. Если указан sql, то он имеет более высокий приоритет по отношению к значению dataobject. Если оба параметра опущены, будет возвращен None.

Параметры

- **filepath** (str) – Путь к файлу.
- **dataobject** (Optional[str]) – Имя таблицы.
- **sql** (Optional[str]) – SQL-запрос.
- **prj** (Optional[str]) – Строка Системы Координат.
- **alias** (Optional[str]) – Псевдоним для открываемой таблицы.

Тип результата Table

TabDataProvider - Провайдер MapInfo

class axipy.da.**TabDataProvider**(info)

Базовые классы: axipy.da.DataProvider

Провайдер MapInfo.

Примечание: Ссылку на провайдер можно получить через глобальную переменную axipy.da.provider_manager.tab.

Methods:

change_coordsystem(filepath, coordsystem)	Изменяет координатную систему в TAB файле без изменения самих данных.
copy_table_files(src_filepath, dest_filepath)	Копирует все связанные файлы с данным файлом в файловой системе под новым именем.
create_open(filepath, schema)	Создает и открывает объект данных.
file_extensions()	Список поддерживаемых расширений файлов.

continues on next page

Таблица 129 – продолжение с предыдущей страницы

get_destination(filepath, schema)	Создает назначение объекта данных.
get_source(filepath[, alias])	Создает источник данных.
open(filepath[, alias])	Открывает объект данных.
remove_table_files(filepath)	Удаляет все связанные файлы с данным файлом в файловой системе.
rename_table_files(src_filepath, dest_filepath)	Переименовывает файл и все связанные файлы с ним.

Attributes:

id	Идентификатор провайдера.

change_coordsystem(filepath, coordsystem)

Изменяет координатную систему в TAB файле без изменения самих данных. Меняется непосредственно сам файл, так что рекомендуется сделать копию.

Параметры

- **filepath** (str) – Путь к файлу TAB (имя файла).
- **coordsystem** (CoordSystem) – Новое значение СК

Исключение RuntimeError – При возникновении ошибки

Список 178: Пример использования

```
in_filepath = 'path/to/input_filename.tab'
cs = CoordSystem.from_prj('10, 104, 7, 0')
provider_manager.tab.change_coordsystem(in_filepath, cs)
```

copy_table_files(src_filepath, dest_filepath)

Копирует все связанные файлы с данным файлом в файловой системе под новым именем.

Параметры

- **src_filepath** (str) – Путь к исходному файлу TAB (имя файла).
- **dest_filepath** (str) – Путь к выходному файлу TAB (имя файла).

Исключение RuntimeError – При возникновении ошибки

Список 179: Пример использования

```
src_filepath = 'path/to/input_filename.tab'
dest_filepath = 'path/to/output_filename.tab'
provider_manager.tab.copy_table_files(src_filepath, dest_filepath)
```

create_open(filepath, schema)

Создает и открывает объект данных.

Параметры

- **filepath** (str) – Путь к файлу.
- **schema** (Schema) – Схема таблицы.

Тип результата Destination

file_extensions()
Список поддерживаемых расширений файлов.

Тип результата List[str]

Результат Пустой список для не файловых провайдеров.

get_destination(filepath, schema)
Создает назначение объекта данных.

Параметры

- **filepath** (str) – Путь к файлу.
- **schema** (Schema) – Схема таблицы.

Тип результата Destination

get_source(filepath, alias=None)
Создает источник данных.

Параметры **filepath** (str) – Путь к файлу.

Тип результата Source

property id
Идентификатор провайдера.

Тип результата str

open(filepath, alias=None)
Открывает объект данных.

Параметры

- **filepath** (str) – Путь к файлу.
- **alias** (Optional[str]) – Псевдоним для открываемой таблицы.

Тип результата Table

remove_table_files(filepath)
Удаляет все связанные файлы с данным файлом в файловой системе.

Параметры **filepath** (str) – Путь к файлу TAB (имя файла).

Исключение RuntimeError – При возникновении ошибки

Список 180: Пример использования

```
filepath = 'path/to/input_filename.tab'
provider_manager.tab.remove_table_files(filepath)
```

rename_table_files(src_filepath, dest_filepath)
Переименовывает файл и все связанные файлы с ним.

Параметры

- **src_filepath** (str) – Путь к исходному файлу TAB (имя файла).
- **dest_filepath** (str) – Путь к новому имени файла TAB (имя файла). Файл не должен существовать.

Исключение RuntimeError – При возникновении ошибки

Список 181: Пример использования

```
src_filepath = 'path/to/old_filename.tab'
dest_filepath = 'path/to/new_filename.tab'
provider_manager.tab.rename_table_files(src_filepath, dest_filepath)
```

SvgDataProvider - Провайдер для SVG

class axipy.da.**SvgDataProvider**(info)

Базовые классы: axipy.da.DataProvider

Провайдер для SVG.

Примечание: Ссылку на провайдер можно получить через глобальную переменную `axipy.da.provider_manager.excel`.

Methods:

create_open()	**Внимание:** Не поддерживается.
file_extensions()	Список поддерживаемых расширений файлов.
get_destination()	**Внимание:** Не поддерживается.
get_source(data[, alias])	Создает источник данных.
open(data[, alias])	Открывает объект данных.

Attributes:

id	Идентификатор провайдера.

create_open()

Внимание: Не поддерживается.

Исключение NotImplementedError –

file_extensions()
Список поддерживаемых расширений файлов.

Тип результата List[str]

Результат Пустой список для не файловых провайдеров.

get_destination()

Внимание: Не поддерживается.

Исключение NotImplementedError –

get_source(data, alias=None)
Создает источник данных.

Параметры **data** (str) – Имя файла или описание источника данных.

Тип результата Source

property id
Идентификатор провайдера.

Тип результата str

open(data, alias=None)
Открывает объект данных.

Параметры

- **data** (str) – Имя файла или описание источника данных.
- **alias** (Optional[str]) – Псевдоним для открываемой таблицы.

Тип результата DataObject

PostgreDataProvider - Провайдер для базы данных PostgreSQL

class axipy.da.**PostgreDataProvider**(info)

Базовые классы: axipy.da.DataProvider

Провайдер для Базы Данных PostgreSQL.

Примечание: Ссылку на провайдер можно получить через глобальную переменную axipy.da.provider_manager.postgre.

Methods:

create_open(schema, db_name, ...)	dataobject,	Создает и открывает объект данных.
file_extensions()		Список поддерживаемых расширений файлов.
get_destination(schema, db_name, ...)	dataobject,	Создает назначение объекта данных.

continues on next page

Таблица 133 – продолжение с предыдущей страницы

get_source(host, db_name, user, password[, ...])	Создает описательную структуру для источника данных.
open(host, db_name, user, password[, port, ...])	Открывает объект данных.

Attributes:

id	Идентификатор провайдера.

create_open(schema, dataobject, db_name, host, user, password, port=5432)
Создает и открывает объект данных.

Параметры

- **schema** (Schema) – Схема таблицы.
- **dataobject** (str) – Имя таблицы.
- **db_name** (str) – Имя базы данных.
- **host** (str) – Адрес сервера.
- **user** (str) – Имя пользователя.
- **password** (str) – Пароль.
- **port** (int) – Порт.

Тип результата Table

file_extensions()
Список поддерживаемых расширений файлов.

Тип результата List[str]

Результат Пустой список для не файловых провайдеров.

get_destination(schema, dataobject, db_name, host, user, password, port=5432)
Создает назначение объекта данных.

Параметры

- **schema** (Schema) – Схема таблицы.
- **dataobject** (str) – Имя таблицы.
- **db_name** (str) – Имя базы данных.
- **host** (str) – Адрес сервера.
- **user** (str) – Имя пользователя.
- **password** (str) – Пароль.
- **port** (int) – Порт.

Тип результата Destination

get_source(host, db_name, user, password, port=5432, dataobject=None, sql=None, prj=None, alias=None)
Создает описательную структуру для источника данных. Она в дальнейшем может быть использована при открытии данных ProviderManager.open().

В качестве таблицы можно указать либо ее наименование dataobject либо текст запроса sql.

Параметры

- **host** (str) – Адрес сервера.
- **db_name** (str) – Имя базы данных.
- **user** (str) – Имя пользователя.
- **password** (str) – Пароль.
- **port** (int) – Порт.
- **dataobject** (Optional[str]) – Имя таблицы.
- **sql** (Optional[str]) – SQL-запрос. Если указан, то он имеет более высокий приоритет по отношению к значению dataobject.
- **prj** (Optional[str]) – Строка Системы Координат.

Пример с указанием имени таблицы:

```
definition = provider_manager.postgre.get_source('localhost', 'test',
↪'postgres', 'postgres', dataobject='world')
table = provider_manager.open(definition)
```

Пример с указанием текста запроса:

```
definition = provider_manager.postgre.get_source('localhost', 'test',
↪'postgres', 'postgres', sql="select * from world where Страна like 'P%'")
table = provider_manager.open(definition)
```

Тип результата Source

property id

Идентификатор провайдера.

Тип результата str

open(host, db_name, user, password, port=5432, dataobject=None, sql=None, prj=None, alias=None)

Открывает объект данных.

В качестве таблицы можно указать либо ее наименование dataobject либо текст запроса sql.

Параметры

- **host** (str) – Адрес сервера.
- **db_name** (str) – Имя базы данных.
- **user** (str) – Имя пользователя.
- **password** (str) – Пароль.
- **port** (int) – Порт.
- **dataobject** (Optional[str]) – Имя таблицы.
- **sql** (Optional[str]) – SQL-запрос. Если указан, то он имеет более высокий приоритет по отношению к значению dataobject.

- **prj** (Optional[str]) – Строка Системы Координат.
- **alias** (Optional[str]) – Псевдоним для открываемой таблицы.

Тип результата Table

OracleDataProvider - Провайдер для базы данных Oracle

class axipy.da.**OracleDataProvider**(info)

Базовые классы: axipy.da.DataProvider

Провайдер для Базы Данных Oracle.

Примечание: Для подключения к БД Oracle необходимо настроить `Oracle Instant Client`.

См. Руководство по установке и активации.

Примечание: Ссылку на провайдер можно получить через глобальную переменную `axipy.da.provider_manager.oracle`.

Methods:

create_open(schema, dataobject, db_name, ...)	Создает и открывает объект данных.
file_extensions()	Список поддерживаемых расширений файлов.
get_destination(schema, dataobject, db_name, ...)	Создает назначение объекта данных.
get_source(host, db_name, user, password[, ...])	Создает описательную структуру для источника данных.
open(host, db_name, user, password[, port, ...])	Открывает объект данных.

Attributes:

id	Идентификатор провайдера.

create_open(schema, dataobject, db_name, host, user, password, port=1521)

Создает и открывает объект данных.

Параметры

- **schema** (Schema) – Схема таблицы.
- **dataobject** (str) – Имя таблицы.
- **db_name** (str) – Имя базы данных.
- **host** (str) – Адрес сервера.
- **user** (str) – Имя пользователя.
- **password** (str) – Пароль.

- **port** (int) – Порт.

Тип результата Table

file_extensions()
Список поддерживаемых расширений файлов.

Тип результата List[str]

Результат Пустой список для не файловых провайдеров.

get_destination(schema, dataobject, db_name, host, user, password, port=1521)
Создает назначение объекта данных.

Параметры

- **schema** (Schema) – Схема таблицы.
- **dataobject** (str) – Имя таблицы.
- **db_name** (str) – Имя базы данных.
- **host** (str) – Адрес сервера.
- **user** (str) – Имя пользователя.
- **password** (str) – Пароль.
- **port** (int) – Порт.

Тип результата Destination

get_source(host, db_name, user, password, port=1521, dataobject=None, sql=None, alias=None)
Создает описательную структуру для источника данных. Она в дальнейшем может быть использована при открытии данных ProviderManager.open().

В качестве таблицы можно указать либо ее наименование dataobject либо текст запроса sql.

Параметры

- **host** (str) – Адрес сервера.
- **db_name** (str) – Имя базы данных.
- **user** (str) – Имя пользователя.
- **password** (str) – Пароль.
- **port** (int) – Порт.
- **dataobject** (Optional[str]) – Имя таблицы.
- **sql** (Optional[str]) – SQL-запрос. Если указан, то он имеет более высокий приоритет по отношению к значению dataobject.

Пример с указанием имени таблицы:

```
definition = provider_manager.oracle.get_source('localhost', 'test', 'oracle',
↪ 'oracle', dataobject='world')
table = provider_manager.open(definition)
```

Пример с указанием текста запроса:

```
definition = provider_manager.oracle.get_source('localhost', 'test', 'oracle',
↪ 'oracle', sql="select * from world where Страна like 'P%'")
table = provider_manager.open(definition)
```

Тип результата Source

property id
Идентификатор провайдера.

Тип результата str

open(host, db_name, user, password, port=1521, dataobject=None, sql=None, alias=None)
Открывает объект данных.

В качестве таблицы можно указать либо ее наименование dataobject либо текст запроса sql.

Параметры

- **host** (str) – Адрес сервера.
- **db_name** (str) – Имя базы данных.
- **user** (str) – Имя пользователя.
- **password** (str) – Пароль.
- **port** (int) – Порт.
- **dataobject** (Optional[str]) – Имя таблицы.
- **sql** (Optional[str]) – SQL-запрос. Если указан, то он имеет более высокий приоритет по отношению к значению dataobject.
- **alias** (Optional[str]) – Псевдоним для открываемой таблицы.

Тип результата Table

MsSqlDataProvider - Провайдер для базы данных MSSQLServer

class axipy.da.**MsSqlDataProvider**(info)

Базовые классы: axipy.da.DataProvider

Провайдер для Базы Данных MSSQLServer.

Примечание: Для работы с СУБД Microsoft SQL Server необходимо скачать и установить Microsoft SQL Server Native Client.

См. Руководство по установке и активации.

Примечание: Ссылку на провайдер можно получить через глобальную переменную axipy.da.provider_manager.mssql.

Methods:

create_open(*args, **kwargs)	Создает и открывает объект данных.
file_extensions()	Список поддерживаемых расширений файлов.
get_destination()	Создает назначение объекта данных.
get_source(host, db_name, user, password[, ...])	Создает описательную структуру для источника данных.
open(host, db_name, user, password[, port, ...])	Открывает объект данных.

Attributes:

id	Идентификатор провайдера.

create_open(*args, **kwargs)

Создает и открывает объект данных.

Пример:

```
provider.create_open(...)
```

Что эквивалентно:

```
provider.get_destiantion(...).create_open()
```

См.также:

DataProvider.destination().

file_extensions()

Список поддерживаемых расширений файлов.

Тип результата List[str]

Результат Пустой список для не файловых провайдеров.

get_destination()

Создает назначение объекта данных.

Исключение **NotImplementedError** – Если провайдер не поддерживает создание назначений.

Тип результата Destination

get_source(host, db_name, user, password, port=1433, dataobject=None, sql=None, alias=None)

Создает описательную структуру для источника данных. Она в дальнейшем может быть использована при открытии данных ProviderManager.open().

Примечание: В качестве таблицы можно указать либо ее наименование dataobject либо текст запроса sql.

Примечание: Ссылку на провайдер можно получить через глобальную переменную axipy.provider_manager.mssql.

Параметры

- **host** (str) – Адрес сервера.
- **db_name** (str) – Имя базы данных.
- **user** (str) – Имя пользователя.
- **password** (str) – Пароль.
- **port** (int) – Порт.
- **dataobject** (Optional[str]) – Имя таблицы.
- **sql** (Optional[str]) – SQL-запрос. Если указан, то он имеет более высокий приоритет по отношению к значению dataobject.

Пример с указанием имени таблицы:

```
definition = provider_manager.mssql.get_source('localhost', 'test', 'sa', 'sa
↪', dataobject='world')
table = provider_manager.open(definition)
```

Пример с указанием текста запроса:

```
definition = provider_manager.mssql.get_source('localhost', 'test', 'sa', 'sa
↪', sql="select * from world where Страна like 'P%'")
table = provider_manager.open(definition)
```

Тип результата Source

property id
Идентификатор провайдера.

Тип результата str

open(host, db_name, user, password, port=1433, dataobject=None, sql=None, alias=None)
Открывает объект данных.

Примечание: В качестве таблицы можно указать либо ее наименование dataobject либо текст запроса sql.

Параметры

- **host** (str) – Адрес сервера.
- **db_name** (str) – Имя базы данных.
- **user** (str) – Имя пользователя.
- **password** (str) – Пароль.
- **port** (int) – Порт.
- **dataobject** (Optional[str]) – Имя таблицы.
- **sql** (Optional[str]) – SQL-запрос. Если указан, то он имеет более высокий приоритет по отношению к значению dataobject.
- **alias** (Optional[str]) – Псевдоним для открываемой таблицы.

Тип результата Table

TmsDataProvider - Тайловый провайдер

class axipy.da.**TmsDataProvider**(info)

Базовые классы: axipy.da.DataProvider

Провайдер для тайловых серверов.

Примечание: Ссылку на провайдер можно получить через глобальную переменную `axipy.da.provider_manager.tms`.

Methods:

create_open()	**Внимание:** Не поддерживается.
file_extensions()	Список поддерживаемых расширений файлов.
get_destination()	**Внимание:** Не поддерживается.
get_source(templateUrl[, minLevel, ...])	Создает источник данных.
open(templateUrl[, minLevel, maxLevel, ...])	Открывает объект данных.

Attributes:

id	Идентификатор провайдера.

create_open()

Внимание: Не поддерживается.

Исключение NotImplementedError –

file_extensions()

Список поддерживаемых расширений файлов.

Тип результата List[str]

Результат Пустой список для не файловых провайдеров.

get_destination()

Внимание: Не поддерживается.

Исключение NotImplementedError –

get_source(templateUrl, minLevel=0, maxLevel=19, size=(256, 256), type_address='xyz', watermark='', watermark_style='', prj=None, live_time=0, alias=None)

Создает источник данных.

Параметры

- **templateUrl** (str) – Шаблон для запроса данных. Например, https://maps.axioma-gis.ru/osm/{LEVEL}/{ROW}/{COL}.png
- **minLevel** (int) – Минимальный уровень показа
- **maxLevel** (int) – Максимальный уровень показа
- **size** (tuple) – Размер тайлов
- **type_address** (str) – Тип адресации к тайлам. Поддерживается два значения: xyz и quadkey
- **watermark** (str) – Ссылка на правообладателя
- **watermark_style** (str) – Стиль оформления текста, с которым на карте будут отображаться данные о правообладателе.
- **prj** (Optional[str]) – Строка с Системой Координат. Если None, то используется значение по умолчанию (CoordSys Earth Projection 10, 157, „m“)
- **live_time** (int) – время жизни тайла в секундах. Если равно 0, то значение не учитывается.

Тип результата Source

property id

Идентификатор провайдера.

Тип результата str

open(templateUrl, minLevel=0, maxLevel=19, size=(256, 256), type_address='xyz', watermark='', watermark_style='', prj=None, live_time=0, alias=None)

Открывает объект данных.

Параметры

- **templateUrl** (str) – Шаблон для запроса данных. Например, https://maps.axioma-gis.ru/osm/{LEVEL}/{ROW}/{COL}.png
- **minLevel** (int) – Минимальный уровень показа
- **maxLevel** (int) – Максимальный уровень показа
- **size** (tuple) – Размер тайлов

- **type_address** (str) – Тип адресации к тайлам. Поддерживается два значения: xyz и quadkey
- **watermark** (str) – Ссылка на правообладателя
- **watermark_style** (str) – Стиль оформления текста, с которым на карте будут отображаться данные о правообладателе.
- **prj** (Optional[str]) – Строка с Системой Координат. Если None, то используется значение по умолчанию (CoordSys Earth Projection 10, 157, „m“)
- **live_time** (int) – время жизни тайла в секундах. Если равно 0, то значение не учитывается.
- **alias** (Optional[str]) – Псевдоним для открываемого источника данных.

Пример открытия источника:

```
prj_mercator = 'CoordSys Earth Projection 10, 104, "m", 0 Bounds (-20037508.
↪34, -20037508.34) (20037508.34, 20037508.34)'
osm_raster = provider_manager.tms.open('http://maps.axioma-gis.ru/osm/{LEVEL}/
↪{ROW}/{COL}.png', prj=prj_mercator)
osm_layer = Layer.create(osm_raster)
map = Map([ osm_layer ])
view_manager.create_mapview(map)
```

Тип результата DataObject

RestDataProvider - Провайдер REST

class axipy.da.**RestDataProvider**(info)

Базовые классы: axipy.da.DataProvider

Провайдер для ArcGIS REST.

Примечание: Ссылку на провайдер можно получить через глобальную переменную axipy.da.provider_manager.rest.

Methods:

create_open()	**Внимание:** Не поддерживается.
file_extensions()	Список поддерживаемых расширений файлов.

continues on next page

Таблица 141 – продолжение с предыдущей страницы

get_destination()	**Внимание:** Не поддерживается.
get_source(url[, fmt, imageSR, size, dpi, ...])	Создает источник данных.
open(url[, fmt, imageSR, size, dpi, ...])	Открывает объект данных.

Attributes:

id	Идентификатор провайдера.

create_open()

Внимание: Не поддерживается.

Исключение NotImplementedError –

file_extensions()
Список поддерживаемых расширений файлов.

Тип результата List[str]

Результат Пустой список для не файловых провайдеров.

get_destination()

Внимание: Не поддерживается.

Исключение NotImplementedError –

get_source(url, fmt='png32', imageSR='imageSR', size='1024*1024', dpi=96, transparent='true', layers='', alias=None)
Создает источник данных.

Параметры

- **url** (str) – Базовый URL.
- **fmt** (str) – Формат выходного растра.
- **imageSR** (str) – Код EPSG для выходного растра.
- **size** (str) – Размер тайлов.
- **dpi** (int) – DPI.
- **transparent** (str) – Прозрачность выходного растра.
- **layers** (str) – Перечень слоев.

Тип результата Source

property id
Идентификатор провайдера.

Тип результата str

open(url, fmt='png32', imageSR='imageSR', size='1024*1024', dpi=96, transparent='true', layers='', alias=None)
Открывает объект данных.

Параметры

- **url** (str) – Базовый URL.
- **fmt** (str) – Формат выходного растра.
- **imageSR** (str) – Код EPSG для выходного растра.
- **size** (str) – Размер тайлов.
- **dpi** (int) – DPI.
- **transparent** (str) – Прозрачность выходного растра.
- **layers** (str) – Перечень слоев.
- **alias** (Optional[str]) – Псевдоним для открываемой таблицы.

Тип результата DataObject

WmsDataProvider - Web Map Service

class axipy.da.**WmsDataProvider**(info)

Базовые классы: axipy.da.DataProvider

Провайдер для Web Map Service.

Примечание: Ссылку на провайдер можно получить через глобальную переменную `axipy.da.provider_manager.wms`.

Methods:

create_open()	**Внимание:** Не поддерживается.
file_extensions()	Список поддерживаемых расширений файлов.

continues on next page

Таблица 143 – продолжение с предыдущей страницы

get_destination()	**Внимание:** Не поддерживается.
get_source(url_capabilities, layers[, ...])	Создает источник данных.
open(url_capabilities, layers[, ...])	Открывает объект данных.

Attributes:

id	Идентификатор провайдера.

create_open()

Внимание: Не поддерживается.

Исключение NotImplementedError –

file_extensions()
Список поддерживаемых расширений файлов.

Тип результата List[str]

Результат Пустой список для не файловых провайдеров.

get_destination()

Внимание: Не поддерживается.

Исключение NotImplementedError –

get_source(url_capabilities, layers, image_format='image/png', prj=None, style=None, alias=None)
Создает источник данных.

Параметры

- **url_capabilities** (str) – URL с метаданными capabilities.
- **layers** (List[str]) – Перечень слоев в виде списка.
- **prj** (Optional[str]) – Строка Системы Координат
- **image_format** (str) – Формат выходного растра.
- **style** (Optional[str]) – Наименование стиля оформления.
- **alias** (Optional[str]) – Псевдоним для открываемого источника данных.

Тип результата Source

property id
Идентификатор провайдера.

Тип результата str

open(url_capabilities, layers, image_format='image/png', prj=None, style=None, alias=None)
Открывает объект данных.

Параметры

- **url_capabilities** (str) – URL с метаданными capabilities.
- **layers** (List[str]) – Перечень слоев в виде списка.
- **prj** (Optional[str]) – Строка Системы Координат
- **image_format** (str) – Формат выходного растра.
- **style** (Optional[str]) – Наименование стиля оформления.
- **alias** (Optional[str]) – Псевдоним для открываемого источника данных.

Пример:

```
wms_raster = provider_manager.wms.open('http://www.mapinfo.com/miwms', ['World
↪'], prj='EPSG:4326', style='AreaStyleGreen')
```

Тип результата DataObject

WmtsDataProvider - Web Map Tile Service

class axipy.da.**WmtsDataProvider**(info)

Базовые классы: axipy.da.DataProvider

Провайдер для тайловых серверов.

Примечание: Ссылку на провайдер можно получить через глобальную переменную axipy.da.provider_manager.wmts.

Methods:

create_open()	**Внимание:** Не поддерживается.
file_extensions()	Список поддерживаемых расширений файлов.

continues on next page

Таблица 145 – продолжение с предыдущей страницы

get_destination()	**Внимание:** Не поддерживается.
get_source(capabilitiesUrl, dataObject[, alias])	Создает источник данных.
open(capabilitiesUrl, dataObject[, alias])	Открывает объект данных.

Attributes:

id	Идентификатор провайдера.

create_open()

Внимание: Не поддерживается.

Исключение NotImplementedError –

file_extensions()
Список поддерживаемых расширений файлов.

Тип результата List[str]

Результат Пустой список для не файловых провайдеров.

get_destination()

Внимание: Не поддерживается.

Исключение NotImplementedError –

get_source(capabilitiesUrl, dataObject, alias=None)
Создает источник данных.

Параметры

- **capabilitiesUrl** (str) – URL запроса метаданных.
- **dataObject** (str) – Наименование слоя.

Тип результата Source

property id
Идентификатор провайдера.

Тип результата str

open(capabilitiesUrl, dataObject, alias=None)
Открывает объект данных.

Параметры

- **capabilitiesUrl** (str) – URL запроса метаданных.
- **dataObject** (str) – Наименование слоя.
- **alias** (Optional[str]) – Псевдоним для открываемого источника данных.

Тип результата DataObject

GdalDataProvider - Растровый провайдер GDAL

class axipy.da.**GdalDataProvider**(info)

Базовые классы: axipy.da.DataProvider

Провайдер для растров.

Примечание: Ссылку на провайдер можно получить через глобальную переменную `axipy.da.provider_manager.gdal`.

Methods:

create_open()	**Внимание:** Не поддерживается.
file_extensions()	Список поддерживаемых расширений файлов.
get_destination()	**Внимание:** Не поддерживается.
get_source(data[, alias])	Создает источник данных.
open(data[, alias])	Открывает объект данных.

Attributes:

id	Идентификатор провайдера.

create_open()

Внимание: Не поддерживается.

Исключение NotImplementedError –

file_extensions()
Список поддерживаемых расширений файлов.

Тип результата List[str]

Результат Пустой список для не файловых провайдеров.

get_destination()

Внимание: Не поддерживается.

Исключение NotImplementedError –

get_source(data, alias=None)
Создает источник данных.

Параметры **data** (str) – Имя файла или описание источника данных.

Тип результата Source

property id
Идентификатор провайдера.

Тип результата str

open(data, alias=None)
Открывает объект данных.

Параметры

- **data** (str) – Имя файла или описание источника данных.
- **alias** (Optional[str]) – Псевдоним для открываемого растра.

Тип результата Table

OgrDataProvider - Векторный провайдер OGR

class axipy.da.**OgrDataProvider**(info)

Базовые классы: axipy.da.DataProvider

Провайдер для векторных данных OGR.

Примечание: Ссылку на провайдер можно получить через глобальную переменную axipy.da.provider_manager.ogr.

Methods:

create_open()	**Внимание:** Не поддерживается.
file_extensions()	Список поддерживаемых расширений файлов.
get_destination()	**Внимание:** Не поддерживается.
get_source(data, dataobject[, alias])	Создает источник данных.
open(data, dataobject[, alias])	Открывает объект данных.

Attributes:

id	Идентификатор провайдера.

create_open()

Внимание: Не поддерживается.

Исключение NotImplementedError –

file_extensions()
Список поддерживаемых расширений файлов.

Тип результата List[str]

Результат Пустой список для не файловых провайдеров.

get_destination()

Внимание: Не поддерживается.

Исключение NotImplementedError –

get_source(data, dataobject, alias=None)
Создает источник данных.

Параметры

- **data** (str) – Источник данных или имя файла.
- **dataobject** (str) – Наименование таблицы

Тип результата Source

property id
Идентификатор провайдера.

Тип результата str

open(data, dataobject, alias=None)
Открывает объект данных.

Параметры

- **data** (str) – Источник данных или имя файла.
- **dataobject** (str) – Наименование таблицы
- **alias** (Optional[str]) – Псевдоним для открываемой таблицы.

Тип результата DataObject

16.1.6.18 axipy.da.raster

Модуль операций с растрами.

class axipy.da.raster.**GCP**(device: Union[Tuple[float, float], PySide2.QtCore.QPointF], scene: Union[Tuple[float, float], PySide2.QtCore.QPointF], label: str = '')
Точка привязки (Ground Control Point).

device
Точка на изображении (пиксели).

Type Union[Tuple[float, float], PySide2.QtCore.QPointF]

scene
Точка на карте.

Type Union[Tuple[float, float], PySide2.QtCore.QPointF]

label
Идентификатор.

Type str

class axipy.da.raster.**Algorithm**(value)
Алгоритм трансформации.

Таблица 151: Значения

Значение	Наименование
POLYNOM1	Аффинитет
POLYNOM2	Полиномиальный второго порядка
POLYNOM3	Полиномиальный третьего порядка
SPLINE	Сплайновый

class axipy.da.raster.**Resample**(value)
Метод интерполяции.

Таблица 152: Значения

Значение	Наименование
NearestNeighbour	Ближайший
Bilinear	Билинейный
Cubic	Кубический
CubicSpline	Кубический сплайн
Lanczos	Ланцоша
Average	Средний
Mode	Самый встречающийся
Max	Максимальный
Min	Минимальный
Med	Медианный
Q1	Первый квартиль
Q3	Третий квартиль
Sum	Сумма

class axipy.da.raster.**Format**(value)
Формат изображения.

Таблица 153: Значения

Значение
JPEG
PNG
BMP
GTiff

class axipy.da.raster.**Compression**(value)
Сжатие.

Примечание: Сжатие можно использовать только для файлов формата GeoTIFF.

Таблица 154: Значения

Значение
NONE
PACKBITS
LZW
DEFLATE

axipy.da.raster.**register**(filepath, bindings, coordsystem)
Регистрирует растр.

Добавляет изображению пространственную привязку.

Параметры

- **filepath** (str) – Файл с изображением.
- **bindings** (Union[List[GCP], QTransform]) – Привязка в виде точек или матрицы преобразования.
- **coordsystem** (CoordSystem) – Координатная система.

Список 182: Пример использования

```
from axipy import CoordSystem, Unit
from axipy.da.raster import register, GCP
from PySide2.QtGui import QTransform

matrix = QTransform()
coordsystem = CoordSystem.from_units(Unit.m)
register(imagefile, matrix, coordsystem)
```

axipy.da.raster.**transform**(inputfile, outputfile, points, coordsystem, algorithm=<Algorithm.SPLINE: 4>, resample=<Resample.NearestNeighbour: 0>, output_format=<Format.GTiff: 4>, compression=<Compression.NONE: 1>)

Трансформирует растр.

Растру, имеющему пространственную привязку, задает новую привязку. На выходе получается новый растр.

Параметры

- **inputfile** (str) – Входной файл с растром.
- **outputfile** (str) – Выходной файл с растром.
- **points** (List[GCP]) – Точки привязки.
- **coordsystem** (CoordSystem) – Координатная система.
- **algorithm** (Algorithm) – Алгоритм трансформации.
- **resample** (Resample) – Метод интерполяции.
- **output_format** (Format) – Выходной формат.
- **compression** (Compression) – Метод сжатия.

Список 183: Пример использования

```
coordsystem = CoordSystem.from_epsg(4326)
gcps = [
    GCP((0, 0), (0, 0)),
    GCP((200, 0), (30, 30)),
    GCP((200, 200), (60, 0)),
]
transform(rasterfile, outputfile, gcps, coordsystem)
```

16.1.7 axipy.render

Модуль отрисовки.

Данный модуль содержит инструменты, предназначенные для отрисовки геопространственных и прочих данных.

16.1.7.1 Map - Карта

class axipy.render.**Map**(layers=[])

Класс карты. Рассматривается как группа слоев, объединенная в единую сущность. Вне зависимости от СК входящих в карту слоев, карта отображает все слои в одной СК. Найти наиболее подходящую для этого можно с помощью `get_best_coordsystem()` или же установить другую.

Единицы измерения расстояний `distanceUnit` и площадей `areaUnit` берутся из настроек по умолчанию.

Параметры layers (List[Layer]) – Список слоев, с которым будет создана карта.

Исключение ValueError – Если один и тот же слой был передан несколько раз.

Список 184: Пример.

```
table_world = provider_manager.openfile(filepath)
world = Layer.create(table_world)
map = Map([ world ])
print('СК:', map.get_best_coordsystem().prj)
print('Охват:', map.get_best_rect())
print('Единицы измерения расстояний:', map.distanceUnit.description)
map.distanceUnit = Unit.mi
print('Единицы измерения расстояний (изменено):', map.distanceUnit.description)
'''
>>> СК: Earth Projection 12, 62, "m", 0
>>> Охват: (-16194966.287183324 -8621185.324024437) (16789976.633236416 8326222.
↪646170927)
>>> Единицы измерения расстояний: километры
>>> Единицы измерения расстояний (изменено): мили
'''
```

Attributes:

areaUnit	Единицы измерения площадей карты.
cosmetic	Косметический слой карты.
custom_labels	Пользовательские метки
distanceUnit	Единицы измерения расстояний на карте.
editable_layer	Слой, установленный для текущего редактирования в карте.
layers	Список слоев и групп слоев.
need_redraw	Signal[] Сигнал о необходимости перерисовки карты.

Methods:

draw(context)	Рисует карту в контексте.
get_best_coordsystem()	Определяет координатную системы карты, наиболее подходящую исходя из содержимого перечня слоев.

continues on next page

Таблица 156 – продолжение с предыдущей страницы

get_best_rect([coordsystem])	Определяет ограничивающий прямоугольник карты.
to_image(width, height[, coordsystem, bbox])	Рисует карту в изображение.

property areaUnit
Единицы измерения площадей карты.

Тип результата AreaUnit

property cosmetic
Косметический слой карты.

Тип результата CosmeticLayer

property custom_labels
Пользовательские метки

Список 185: Пример.

```
table_world = provider_manager.openfile(filepath)
world_layer = Layer.create(table_world)
map_ = Map([ world_layer ])
# Определим свойства
p = CustomLabelProperties()
# Переназначим выражение
p.expression = 'Новое выражение'
# Угол поворота
p.angle = 30
# Выноска будет в виде стрелки
p.endType = CustomLabelEndType.Arrow
# Положение выноски
p.position = Point(100, 200)
# Установим свойства
map_.custom_labels.set(map_.layers[0], 1, p)
# Запрос свойств
props = map_.custom_labels.get(map_.layers[0], 1)
```

Тип результата CustomLabels

property distanceUnit
Единицы измерения расстояний на карте.

Тип результата LinearUnit

draw(context)
Рисует карту в контексте.

Параметры context (Context) – Контекст рисования.

Список 186: Пример.

```
# Пример получения карты как растра
map = Map([ world ])
image = QImage(1600, 800, QImage.Format_ARGB32_Premultiplied)
image.fill(Qt.white)
painter = QPainter(image)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
context = Context(painter)
map.draw(context)
```

property editable_layer
Слой, установленный для текущего редактирования в карте.

Исключение **ValueError** – При попытке установить слой, не принадлежащий этой карте.

Тип результата VectorLayer

get_best_coordsystem()
Определяет координатную системы карты, наиболее подходящую исходя из содержимого перечня слоев.

Тип результата CoordSystem

get_best_rect(coordsystem=None)
Определяет ограничивающий прямоугольник карты.

Параметры **coordsystem** (Optional[CoordSystem]) – Координатная система, в которой необходимо получить результат. Если отсутствует, будет выдан результат для наиболее подходящей координатной системы.

Тип результата Rect

property layers
Список слоев и групп слоев.

Примечание: Не содержит косметический слой cosmetic.

Список 187: Примеры доступа.

```
# Создадим карту с тремя слоями
map = Map([ world, worldcap, russia ])
print(len(map.layers))
'''
>>> 3
'''
print(map.layers[0].title)
'''
>>> world
'''
# Группировка первый двух слоев с именем "Мир"
map.layers.group([0,1], 'Мир')
# Перечень элементов
for l in map.layers:
    if isinstance(l, Layer):
        print('Слой:', l.title)
    elif isinstance(l, ListLayers):
        print('Группа:', l.title)
'''
>>> Группа: Мир
>>> Слой: russia
'''
# Изменение позиции
```

(continues on next page)

(продолжение с предыдущей страницы)

```
map.layers.move(0,1)
# Управление видимостью группы слоев
map.layers[1].visible = False
# Разгруппировка ранее созданной группы
map.layers.ungroup(1)
# Удаление слоя из карты
map.layers.remove(1)
# Добавление пустой группы
map.layers.add_group('Новая группа')
```

Тип результата ListLayers

property need_redraw
Signal[] Сигнал о необходимости перерисовки карты. Возникает при изменении контента одного или нескольких слоев карты. Это может быть обусловлено изменением данных таблиц.

Список 188: Пример.

```
# Смотрим активное окно.
if isinstance(view_manager.active, MapView):
    # Если это карта, подключимся к событию обновления окна этой карты.
    map_view = view_manager.active
    map_view.map.need_redraw.connect(lambda : print('Update map'))
```

Тип результата Signal

to_image(width, height, coordsystem=None, bbox=None)
Рисует карту в изображение.

Параметры

- **width** (int) – Ширина выходного изображения.
- **height** (int) – Высота выходного изображения.
- **coordsystem** (Optional[CoordSystem]) – Координатная система. Если не задана, берется наиболее подходящая.
- **bbox** (Optional[Rect]) – Ограничивающий прямоугольник. Если не задан, берется у карты.

Тип результата QImage

Результат Изображение.

16.1.7.2 ListLayers - Список слоев карты

class axipy.render.**ListLayers**

Группа слоев. Может включать в себя как слои axipy.render.Layer так и группы слоев axipy.render.ListLayers. Пример использования см axipy.render.Map.layers

Methods:

add_group(name)	Создает пустую группу.
append(layer)	Добавляет слой в карту.
at(index)	Возвращает слой или группы слоев по их индексу.
group(indexes, name)	Группировка слоев и групп в соответствие со списком их индексов.
move(from_index, to_index)	Перемещает слой или вложенную группу слоев в списке слоев по его индексу.
remove(index)	Удаляет слой по индексу.
ungroup(index)	Разгруппировка группы слоев по его индексу.

Attributes:

count	Количество слоев и групп слоев.
title	Наименование группы.
visible	Управляет видимостью группы.

add_group(name)

Создает пустую группу.

Параметры name (str) – Наименование создаваемой группы.

append(layer)

Добавляет слой в карту. Добавление группы слоев не поддерживается и производится путем группировки существующих элементов посредством метода group().

Параметры layer (Layer) – Добавляемый слой.

Исключение ValueError – Если слой уже содержится в карте.

at(index)

Возвращает слой или группы слоев по их индексу.

Параметры index (int) – Индекс слоя или группы в списке.

Например:

```
layers.at(2)
layers[2]
```

Тип результата Union[Layer, ListLayers]

property count

Количество слоев и групп слоев. Так же допустимо использование функции

len()

Тип результата int

group(indexes, name)
Группировка слоев и групп в соответствие со списком их индексов. При этом создается новая группа и все элементы (слои и группы слоев) помещаются внутрь этой группы.

Параметры

- **indexes** (List[int]) – Список индексов элементов, которые необходимо объединить.
- **name** (str) – Наименование создаваемой группы.

move(from_index, to_index)
Перемещает слой или вложенную группу слоев в списке слоев по его индексу.

Параметры

- **from_index** (int) – Индекс слоя для перемещения.
- **to_index** (int) – Целевой индекс.

remove(index)
Удаляет слой по индексу.

Параметры **index** (int) – Индекс удаляемого слоя.

property title
Наименование группы.

Тип результата str

ungroup(index)
Разгруппировка группы слоев по его индексу. при этом все внутренние элементы переносятся на верхний уровень данного списка. Если по индексу располагается не группа, то будет выброшено исключение.

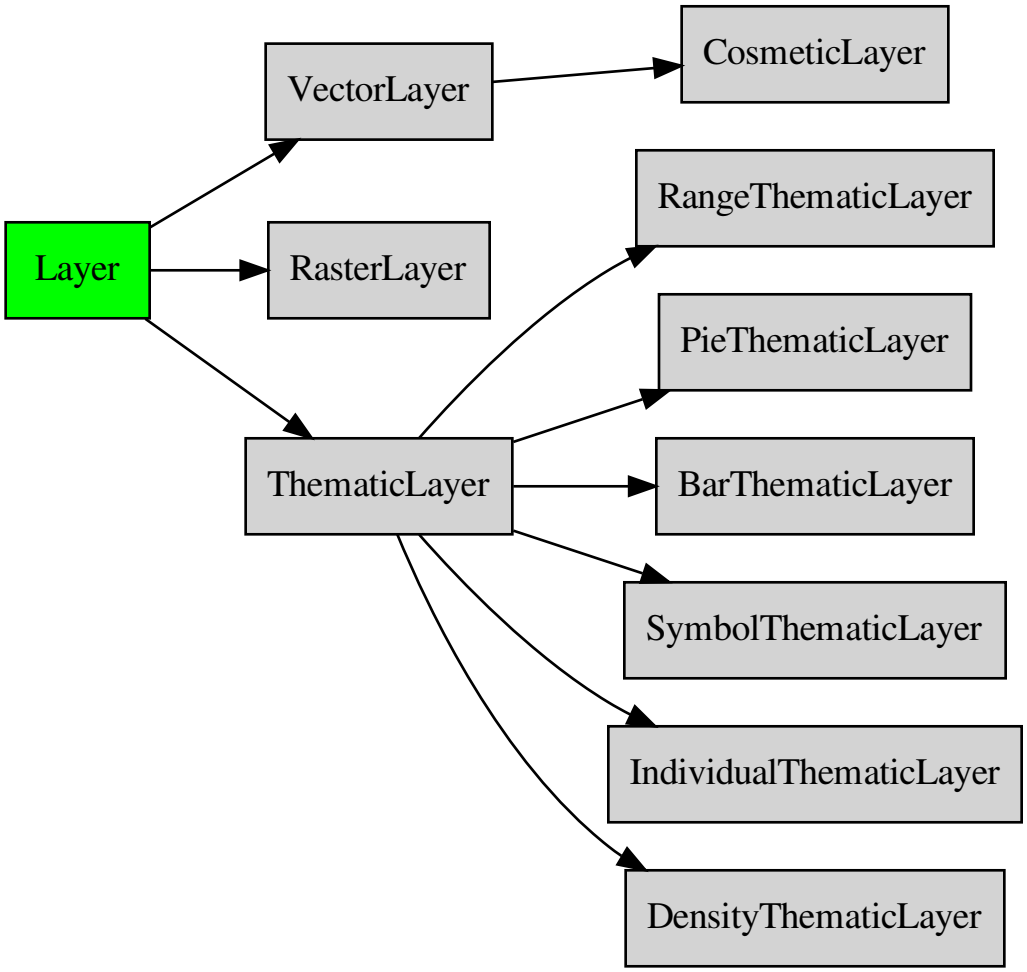
Параметры **index** (int) – Индекс группы слоев.

property visible
Управляет видимостью группы.

16.1.7.3 axipy.render layer

Layer - Слой

Иерархия классов слоев карты:



class axipy.render.**Layer**

Абстрактный базовый класс для слоя карты.

Для создания нового экземпляра для векторного или растрового источника данных необходимо использовать метод `Layer.create()`. Для тематических слоев - использовать соответствующие им конструкторы.

property coordsystem

Координатная система, в которой находятся данные, отображаемые слоем.

Тип результата CoordSystem

classmethod create(dataObject)

Создает слой на базе открытой таблицы или растра.

Параметры dataObject (DataObject) – Таблица или растр. В зависимости от переданного объекта будет создан VectorLayer или RasterLayer.

Список 189: Пример создания слоя на базе файла.

```
# Векторный слой
table = provider_manager.openfile(filepath)
vector_layer = Layer.create(table)
# Подпишемся на обновление контента слоя
vector_layer.need_redraw.connect(lambda: print('Update layer'))
```

Тип результата Layer

property data_changed
Signal[] Сигнал об изменении контента слоя.

Тип результата Signal

property data_object
Источник данных для слоя.

Тип результата DataObject

get_bounds()
Возвращает область, в которую попадают все данные, которые могут быть отображены на слое.

Тип результата Rect

property is_valid
Проверка на валидность объекта. Слой мог быть удален, как пример, в связи с закрытием таблицы

Тип результата bool

property max_zoom
Максимальная ширина окна, при котором слой отображается на карте. Учитывается только при установленном zoom_restrict=True

Тип результата float

property min_zoom
Минимальная ширина окна, при котором слой отображается на карте. Учитывается только при установленном zoom_restrict=True

Тип результата float

property need_redraw
Signal[] Сигнал о необходимости перерисовать слой.

Тип результата Signal

property opacity
Прозрачность слоя в составе карты. Доступные значения от 0 до 100.

Тип результата int

property title
Наименование слоя.

Тип результата str

property visible
Управляет видимостью слоя.

Выключение видимости верхнего слоя для активной карты:

```
if view_manager.active is not None:
    view_manager.active.map.layers[0].visible = False
```

property zoom_restrict
Будет ли использоваться ограничение по отображению. Если установлено True, то для ограничения отображения слоя в зависимости от масштаба используются значения свойств zoom_min и zoom_max

Тип результата bool

RasterLayer - Растровый слой

class axipy.render.**RasterLayer**

Базовые классы: axipy.render.Layer

Класс, который должен использоваться в качестве базового класса для тех слоев, в которых используются свойства отрисовки растрового изображения.

Примечание: Создание слоя производится посредством метода вызова Layer.create()

Список 190: Примеры создания растрового слоя.

```
raster = provider_manager.openfile(filename)
raster_layer = Layer.create(raster)
raster_layer.transparentColor = QColor('#000014')
```

Attributes:

brightness	Яркость.
contrast	Контраст.
coordsystem	Координатная система, в которой находятся данные, отображаемые слоем.
data_changed	Signal[] Сигнал об изменении контента слоя.
data_object	Источник данных для слоя.
grayscale	Является ли данное изображение черно-белым.
is_valid	Проверка на валидность объекта.
max_zoom	Максимальная ширина окна, при котором слой отображается на карте.
min_zoom	Минимальная ширина окна, при котором слой отображается на карте.
need_redraw	Signal[] Сигнал о необходимости перерисовать слой.
opacity	Прозрачность слоя в составе карты.

continues on next page

Таблица 159 – продолжение с предыдущей страницы

title	Наименование слоя.
transparentColor	Цвет растра, который обрабатывается как прозрачный.
visible	Управляет видимостью слоя.
zoom_restrict	Будет ли использоваться ограничение по отображению.

Methods:

create(dataObject)	Создает слой на базе открытой таблицы или растра.
get_bounds()	Возвращает область, в которую попадают все данные, которые могут быть отображены на слое.

property brightness
Яркость. Значение может быть в пределах от -100 до 100.

Тип результата int

property contrast
Контраст. Значение может быть в пределах от -100 до 100.

Тип результата int

property coordsystem
Координатная система, в которой находятся данные, отображаемые слоем.

Тип результата CoordSystem

classmethod create(dataObject)
Создает слой на базе открытой таблицы или растра.

Параметры dataObject (DataObject) – Таблица или растр. В зависимости от переданного объекта будет создан VectorLayer или RasterLayer.

Список 191: Пример создания слоя на базе файла.

```
# Векторный слой
table = provider_manager.openfile(filepath)
vector_layer = Layer.create(table)
# Подпишемся на обновление контента слоя
vector_layer.need_redraw.connect(lambda: print('Update layer'))
```

Тип результата Layer

property data_changed
Signal[] Сигнал об изменении контента слоя.

Тип результата Signal

property data_object
Источник данных для слоя.

Тип результата DataObject

get_bounds()
Возвращает область, в которую попадают все данные, которые могут быть отображены на слое.

Тип результата Rect

property grayscale
Является ли данное изображение черно-белым.

Тип результата bool

property is_valid
Проверка на валидность объекта. Слой мог быть удален, как пример, в связи с закрытием таблицы

Тип результата bool

property max_zoom
Максимальная ширина окна, при котором слой отображается на карте. Учитывается только при установленном zoom_restrict=True

Тип результата float

property min_zoom
Минимальная ширина окна, при котором слой отображается на карте. Учитывается только при установленном zoom_restrict=True

Тип результата float

property need_redraw
Signal[] Сигнал о необходимости перерисовать слой.

Тип результата Signal

property opacity
Прозрачность слоя в составе карты. Доступные значения от 0 до 100.

Тип результата int

property title
Наименование слоя.

Тип результата str

property transparentColor
Цвет растра, который обрабатывается как прозрачный.

Тип результата QColor

property visible
Управляет видимостью слоя.

Выключение видимости верхнего слоя для активной карты:

```
if view_manager.active is not None:
    view_manager.active.map.layers[0].visible = False
```

property zoom_restrict
Будет ли использоваться ограничение по отображению. Если установлено True, то для ограничения отображения слоя в зависимости от масштаба используются значения свойств zoom_min и zoom_max

Тип результата bool

VectorLayer - Векторный слой

class axipy.render.**VectorLayer**

Базовые классы: axipy.render.Layer

Слой, основанный на базе векторных данных.

Примечание: Создание слоя производится посредством метода вызова Layer.create()

Список 192: Примеры работы со свойствами слоя.

```
# Зададим в качестве формулы метки атрибут "Страна" и запретим перекрытие меток␣
↪друг другом:
world.label.text = "Страна"
world.label.placementPolicy = LabelOverlap.DisallowOverlap
# Задание стиля оформления слоя
style_lay = Style.from_mapinfo("Pen (1, 2, 0) Brush (8, 255) Symbol (33,255,14)")
world.overrideStyle = style_lay
# Для сброса переопределения достаточно задать значение None
world.overrideStyle = None
```

Attributes:

coordsystem	Координатная система, в которой находятся данные, отображаемые слоем.
data_changed	Signal[] Сигнал об изменении контента слоя.
data_object	Источник данных для слоя.
is_valid	Проверка на валидность объекта.
label	Метки слоя.
linesDirectionVisibile	Показ направлений линий.
max_zoom	Максимальная ширина окна, при котором слой отображается на карте.
min_zoom	Минимальная ширина окна, при котором слой отображается на карте.
need_redraw	Signal[] Сигнал о необходимости перерисовать слой.
nodesVisible	Показ узлов линий и полигонов.
opacity	Прозрачность слоя в составе карты.
overrideStyle	Переопределяемый стиль слоя.
showCentroid	Показ центроидов на слое.
thematic	Перечень тематик для данного слоя.
title	Наименование слоя.
visible	Управляет видимостью слоя.
zoom_restrict	Будет ли использоваться ограничение по отображению.

Methods:

create(dataObject)	Создает слой на базе открытой таблицы или растра.
get_bounds()	Возвращает область, в которую попадают все данные, которые могут быть отображены на слое.

property coordsystem
Координатная система, в которой находятся данные, отображаемые слоем.

Тип результата CoordSystem

classmethod create(dataObject)
Создает слой на базе открытой таблицы или растра.

Параметры dataObject (DataObject) – Таблица или растр. В зависимости от переданного объекта будет создан VectorLayer или RasterLayer.

Список 193: Пример создания слоя на базе файла.

```
# Векторный слой
table = provider_manager.openfile(filepath)
vector_layer = Layer.create(table)
# Подпишемся на обновление контента слоя
vector_layer.need_redraw.connect(lambda: print('Update layer'))
```

Тип результата Layer

property data_changed
Signal[] Сигнал об изменении контента слоя.

Тип результата Signal

property data_object
Источник данных для слоя.

Тип результата DataObject

get_bounds()
Возвращает область, в которую попадают все данные, которые могут быть отображены на слое.

Тип результата Rect

property is_valid
Проверка на валидность объекта. Слой мог быть удален, как пример, в связи с закрытием таблицы

Тип результата bool

property label
Метки слоя. В качестве формулы может использоваться или наименование поля таблицы или выражение.

Тип результата Label

property linesDirectionVisibile
Показ направлений линий.

Тип результата bool

property max_zoom
Максимальная ширина окна, при котором слой отображается на карте. Учитывается только при установленном zoom_restrict=True

Тип результата float

property min_zoom
Минимальная ширина окна, при котором слой отображается на карте. Учитывается только при установленном zoom_restrict=True

Тип результата float

property need_redraw
Signal[] Сигнал о необходимости перерисовать слой.

Тип результата Signal

property nodesVisible
Показ узлов линий и полигонов.

Тип результата bool

property opacity
Прозрачность слоя в составе карты. Доступные значения от 0 до 100.

Тип результата int

property overrideStyle
Переопределяемый стиль слоя. Если задан как None (по умолчанию), объекты будут отображены на основании оформления источника данных.

Тип результата Style

property showCentroid
Показ центроидов на слое.

Тип результата bool

property thematic
Перечень тематик для данного слоя. Работа с тематическими слоями похожа на работу со списком list.

Список 194: Пример.

```
# Создадим тематический слой
range1 = RangeThematicLayer("Население")
# Добавим в основной слой
world.thematic.append(range1)
# Получим добавленный тематический слой
range1 = world.thematic[0]
# Просмотр всех тематик слоя
for t in world.thematic:
    print('thematic:', t.title)
```

Тип результата ListThematic

property title
Наименование слоя.

Тип результата str

property visible
Управляет видимостью слоя.

Выключение видимости верхнего слоя для активной карты:

```
if view_manager.active is not None:
    view_manager.active.map.layers[0].visible = False
```

property zoom_restrict
Будет ли использоваться ограничение по отображению. Если установлено True, то для ограничения отображения слоя в зависимости от масштаба используются значения свойств zoom_min и zoom_max

Тип результата bool

CosmeticLayer - Косметический слой

class axipy.render.**CosmeticLayer**

Базовые классы: axipy.render.VectorLayer

Косметический слой.

Attributes:

coordsystem	Координатная система, в которой находятся данные, отображаемые слоем.
data_changed	Signal[] Сигнал об изменении контента слоя.
data_object	Источник данных для слоя.
is_valid	Проверка на валидность объекта.
label	Метки слоя.
linesDirectionVisibile	Показ направлений линий.
max_zoom	Максимальная ширина окна, при котором слой отображается на карте.
min_zoom	Минимальная ширина окна, при котором слой отображается на карте.
need_redraw	Signal[] Сигнал о необходимости перерисовать слой.
nodesVisible	Показ узлов линий и полигонов.
opacity	Прозрачность слоя в составе карты.
overrideStyle	Переопределяемый стиль слоя.
showCentroid	Показ центроидов на слое.
thematic	Перечень тематик для данного слоя.
title	Наименование слоя.
visible	Управляет видимостью слоя.
zoom_restrict	Будет ли использоваться ограничение по отображению.

Methods:

create(dataObject)	Создает слой на базе открытой таблицы или растра.
get_bounds()	Возвращает область, в которую попадают все данные, которые могут быть отображены на слое.

property coordsystem
Координатная система, в которой находятся данные, отображаемые слоем.

Тип результата CoordSystem

classmethod create(dataObject)
Создает слой на базе открытой таблицы или растра.

Параметры dataObject (DataObject) – Таблица или растр. В зависимости от переданного объекта будет создан VectorLayer или RasterLayer.

Список 195: Пример создания слоя на базе файла.

```
# Векторный слой
table = provider_manager.openfile(filepath)
vector_layer = Layer.create(table)
# Подпишемся на обновление контента слоя
vector_layer.need_redraw.connect(lambda: print('Update layer'))
```

Тип результата Layer

property data_changed
Signal[] Сигнал об изменении контента слоя.

Тип результата Signal

property data_object
Источник данных для слоя.

Тип результата DataObject

get_bounds()
Возвращает область, в которую попадают все данные, которые могут быть отображены на слое.

Тип результата Rect

property is_valid
Проверка на валидность объекта. Слой мог быть удален, как пример, в связи с закрытием таблицы

Тип результата bool

property label
Метки слоя. В качестве формулы может использоваться или наименование поля таблицы или выражение.

Тип результата Label

property linesDirectionVisibile
Показ направлений линий.

Тип результата bool

property max_zoom
Максимальная ширина окна, при котором слой отображается на карте. Учитывается только при установленном zoom_restrict=True

Тип результата float

property min_zoom
Минимальная ширина окна, при котором слой отображается на карте. Учитывается только при установленном zoom_restrict=True

Тип результата float

property need_redraw
Signal[] Сигнал о необходимости перерисовать слой.

Тип результата Signal

property nodesVisible
Показ узлов линий и полигонов.

Тип результата bool

property opacity
Прозрачность слоя в составе карты. Доступные значения от 0 до 100.

Тип результата int

property overrideStyle
Переопределяемый стиль слоя. Если задан как None (по умолчанию), объекты будут отображены на основании оформления источника данных.

Тип результата Style

property showCentroid
Показ центроидов на слое.

Тип результата bool

property thematic
Перечень тематик для данного слоя. Работа с тематическими слоями похожа на работу со списком list.

Список 196: Пример.

```
# Создадим тематический слой
range1 = RangeThematicLayer("Население")
# Добавим в основной слой
world.thematic.append(range1)
# Получим добавленный тематический слой
range1 = world.thematic[0]
# Просмотр всех тематик слоя
for t in world.thematic:
    print('thematic:', t.title)
```

Тип результата ListThematic

property title
Наименование слоя.

Тип результата str

property visible
Управляет видимостью слоя.

Выключение видимости верхнего слоя для активной карты:

```
if view_manager.active is not None:
    view_manager.active.map.layers[0].visible = False
```

property zoom_restrict
Будет ли использоваться ограничение по отображению. Если установлено True, то для ограничения отображения слоя в зависимости от масштаба используются значения свойств zoom_min и zoom_max

Тип результата bool

ListThematic - Перечень тематик для векторного слоя

class axipy.render.**ListThematic**
Список тематических слоев (тематик) карты.

append(lay)
Добавить тематику.

Параметры lay (ThematicLayer) – Добавляемый тематический слой.

at(idx)
Получение тематики по ее индексу.

Параметры idx (int) – Индекс запрашиваемой тематики.

Тип результата ThematicLayer

property count
Количество тематик слоя.

Тип результата int

move(fromIdx, toIdx)
Поменять тематики местами.

Параметры

- **fromIdx** (int) – Текущий индекс.
- **toIdx** (int) – Новое положение.

remove(idx)
Удалить тематику.

Параметры idx (int) – Индекс удаляемого слоя.

Label - Метка для векторного слоя

class axipy.render.**Label**

Метки слоя. Доступны через свойство векторного слоя VectorLayer.label.

Список 197: Пример использования.

```
# Открываем таблицу
table = provider_manager.openfile(filepath)
# Создаем слой
layer = Layer.create(table)
# Формула метки
layer.label.text  =  'Страна'
# Видимость
layer.label.visible = True
# Если метки перекрывают друг друга, ищем другое положение
layer.label.placementPolicy = LabelOverlap.OtherPosition
# Цвет шрифта
layer.label.color = Qt.blue
# устанавливаем прорачность
layer.label.opacity = 50
# Показываем в пределах (0...3000км)
layer.label.rangeEnabled = True
layer.label.rangeMax = 3000000
# Положение подписей для точечных объектов
p_layout = layer.label.pointLayout
p_layout.position = LabelLayoutPosition.BottomRight
p_layout.visible = True
p_layout.offset = QSize(3,3)
layer.label.pointLayout = p_layout
# Положение подписей для линейных объектов
l_layout = layer.label.lineLayout
l_layout.position = LabelLayoutPosition.Bottom
layer.label.lineLayout = l_layout
# Горизонтальное выравнивание подписей для линий
layer.label.horizontalAlign = LabelHorizontalAlign.Center
# Игнорируем дубликаты
layer.label.supressDuplicates = True
# Свес линии
layer.label.overhang = 67
```

property areaInterior

Режим подписей для областей. По умолчанию LabelAreaInterior.Centroid.

Тип результата LabelAreaInterior

property areaLayout

Положение подписей для областей.

Тип результата LabelLayout

property areaPosition

Режим подписей для областей. По умолчанию LabelAreaPosition.Horizontal.

Тип результата LabelAreaPosition

property backgroundColor

Цвет фона

Тип результата QColor

property backgroundSize
Толщина фона в пунктах.

Тип результата int

property backgroundType
Фон подписи. По умолчанию отсутствует

Тип результата LabelBackgroundType

property color
Цвет шрифта меток.

Тип результата QColor

property font
Шрифт.

Тип результата QFont

property horizontalAlign
Горизонтальное выравнивание подписей. По умолчанию LabelHorizontalAlign.Flat

Тип результата LabelHorizontalAlign

property lineKeepDirection
Направление текста строится вдоль направления линии. По умолчанию False.

Тип результата bool

property lineLayout
Положение подписей для линий.

Тип результата LabelLayout

property linePosition
Режим подписей для линий. По умолчанию LabelLinePosition.FollowPath.

Тип результата LabelLinePosition

property opacity
Прозрачность (0..100). По умолчанию 100 (Непрозрачно).

Тип результата int

property overhang
Максимальный свес для линии (в %).

Тип результата int

property placementPolicy
Принцип наложения меток на слой карты. По умолчанию LabelOverlap.AllowOverlap

Тип результата LabelOverlap

property pointLayout
Положение подписей для точек.

Тип результата LabelLayout

property rangeEnabled
Показывать в пределах. Если True, используются свойства rangeMin и rangeMax. По умолчанию False.

Тип результата bool

property rangeMax
Максимальный предел показа с метрах при включенном свойстве rangeEnabled.

Тип результата float

property rangeMin
Минимальный предел показа с метрах при включенном свойстве rangeEnabled.

Тип результата float

property shadow
Тень. По умолчанию False

Тип результата bool

property spacing
Разрядка. По умолчанию False

Тип результата bool

property supressDuplicates
Запретить повтор подписей. Подписи с одинаковым текстом на этом слое будут отображаться один раз. По умолчанию False.

Тип результата bool

property text
Наименование атрибута таблицы либо выражение для метки, которое может основываться на одном или нескольких атрибутах.

Тип результата str

property useClip
Использовать динамические подписи. По умолчанию False.

Тип результата bool

property visible
Управляет видимостью меток.

Тип результата bool

LabelOverlap - Наложение подписей

class axipy.render.**LabelOverlap**(value)
Стратегия при наложении подписей Label.placementPolicy.

Таблица 165: Значения

Значение	Наименование
AllowOverlap	Допускать перекрытие меток (по умолчанию)
DisallowOverlap	Не допускать перекрытие меток
OtherPosition	Пробовать найти для метки новую позицию

LabelBackgroundType - Фон подписи

class axipy.render.**LabelBackgroundType**(value)
Фон подписи Label.backgroundType.

Таблица 166: Значения

Значение	Наименование
Empty	Отсутствует
Outline	Кайма
Frame	Фломастер

LabelLinePosition - Режим подписей для линий

class axipy.render.**LabelLinePosition**(value)
Режим подписей для линий Label.linePosition.

Таблица 167: Значения

Значение	Наименование
Horizontal	Горизонтально
Parallel	Вдоль сегмента
FollowPath	Вдоль линии

LabelAreaPosition - Режим подписей для областей

class axipy.render.**LabelAreaPosition**(value)
Режим подписей для областей Label.areaPosition.

Таблица 168: Значения

Значение	Наименование
Centroid	У центроида
Horizontal	Горизонтально
Vertical	Вертикально
Automatic	Автоматически

LabelAreaInterior - Выбор участка для областей

class axipy.render.**LabelAreaInterior**(value)
Выбор участка для областей Label.areaInterior.

Таблица 169: Значения

Значение	Наименование
Max	Наибольший
Centroid	Ближайший к центроиду

LabelLayout - Положение подписей

class axipy.render.**LabelLayout**
Положение подписей `Label`.

property offset
Смещение. Интервал значений (0..24)

Тип результата `QSize`

property position
Расположение подписи.

Тип результата `LabelLayoutPosition`

property visible
Видимость подписи для данного типа геометрии. По умолчанию True

Тип результата `bool`

LabelLayoutPosition - Положение подписей

class axipy.render.**LabelLayoutPosition**(value)
Относительное положение подписи `LabelLayout.position`.

Таблица 170: Значения

Значение	Наименование
Center	По центру
TopLeft	Сверху слева
Top	Сверху
TopRight	Сверху справа
Right	Справа
BottomRight	Снизу справа
Bottom	Снизу
BottomLeft	Снизу слева
Left	Слева

LabelHorizontalAlign - Горизонтальное выравнивание подписей

class axipy.render.**LabelHorizontalAlign**(value)
Горизонтальное выравнивание подписи `Label.horizontalAlign`.

Таблица 171: Значения

Значение	Наименование
Flat	По центру пологого участка (можно задать угол на вкладке Дополнительно)
Center	Середина подписи совпадает с центром линии
Begin	Подпись располагается в начале линии
End	Подпись располагается в конце линии

CustomLabelEndType - Тип выноски

class axipy.render.**CustomLabelEndType**(value)
Тип выноски для метки CustomLabelProperties.endType.

Таблица 172: Значения

Значение	Наименование
EndNone	Не отображать (по умолчанию)
Line	Линия
Arrow	Стрелка

CustomLabelProperties - Свойства выносной метки

class axipy.render.**CustomLabelProperties**
Свойства выносной метки. Используется при задании параметров положения метки карты CustomLabels.set() или получения их CustomLabels.get().

property angle
Угол поворота текста в градусах

Тип результата float

property endType
Тип выноски. По умолчанию CustomLabelEndType.EndNone

Тип результата CustomLabelEndType

property expression
Текст. Если не задано, используется базовая формула для слоя.

Тип результата str

property position
Переопределенное положение метки. Если используется по умолчанию, возвращается None. Если при задании не указана система координат, используется СК карты.

Тип результата Point

16.1.7.4 Legend - Легенда слоя

class axipy.render.**Legend**(lay)
Легенда слоя. Позволяет получить информацию об условных обозначениях на слое. Созданная легенда в дальнейшем может быть помещена на лист отчета axipy.render.Report как axipy.render.LegendReportItem или же расположена в отдельном окне легенд слоев для карты axipy.render.Map.

Параметры lay (Layer) – Слой, для которого создается легенда.

Список 198: Пример создания легенды.

```
range1 = RangeThematicLayer("Население")
world.thematic.add(range1)
# Легенда для тематического слоя
legend = Legend(range1)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
legend.columns = 2
# Зададим стиль заднего фона и окантовки
legend.border_style = LineStyle(3, Qt.red)
legend.fill_style = PolygonStyle(49, Qt.yellow)
# Изменим описание для первого элемента
item = legend.items[0]
item.title = 'Описание'
legend.items[0] = item
# Заголовок легенды
legend.caption = 'Легенда для слоя'
# Стиль заголовка
legend.style_caption = Style.from_mapinfo("Font (\"Arial\", 0, 9, 255)")
# Просмотр всех стилей легенды
for it in legend.items:
    print(it.title, it.visible, it.style.to_mapinfo())
# Изменение позиции элемента легенды
legend.items.move(0,1)
# Отрисуем легенду в контексте вместе с картой
image = QImage(800, 600, QImage.Format_ARGB32_Premultiplied)
image.fill(Qt.white)
painter = QPainter(image)
context = Context(painter)
legend.position = (100, 50)
legend.draw(context)
'''
>>> Описание True Pen (1, 2, 8421504) Brush (2, 16776960)
>>> 55419-166640 True Pen (1, 2, 8421504) Brush (2, 12582656)
>>> 166640-631500 True Pen (1, 2, 8421504) Brush (2, 8453888)
...
'''
```

Attributes:

border_style	Стиль используемой окантовки.
caption	Заголовок легенды.
columns	Количество колонок в легенде.
fill_style	Стиль заливки заднего фона.
items	Перечень стилей легенды.
position	Положение легенды в контексте рисования.
style_caption	Стиль заголовка легенды.
style_subcaption	Стиль подзаголовка легенды.
style_text	Стиль текстовых подписей.
subcaption	Подзаголовок легенды.

Methods:

draw(context)	Рисует легенду в контексте.
refresh()	Обновляет стили из источника.
to_image(width, height)	Возвращает легенду в виде растра.

property border_style

Стиль используемой окантовки. Отображается если has_border установлено в True.

Тип результата Style

property caption
Заголовок легенды. Стиль заголовка задается свойством style_caption

Тип результата str

property columns
Количество колонок в легенде. По умолчанию 1.

Тип результата int

draw(context)
Рисует легенду в контексте.

Легенду также можно отрисовать совместно с картой в одном контексте (см. Map.draw()).

Параметры **context** (Context) – Контекст рисования.

property fill_style
Стиль заливки заднего фона.

Тип результата Style

property items
Перечень стилей легенды. Реализован в виде списка. Для изменения какого-либо параметра необходимо сначала получить элемент, затем поменять требуемое свойство, а затем измененный элемент переназначить.

Тип результата ListLegendItems

property position
Положение легенды в контексте рисования.

Тип результата Pnt

refresh()
Обновляет стили из источника.

property style_caption
Стиль заголовка легенды.

Тип результата Style

property style_subcaption
Стиль подзаголовка легенды.

Тип результата Style

property style_text
Стиль текстовых подписей.

Тип результата Style

property subcaption
Подзаголовок легенды. Стиль заголовка задается свойством style_subcaption

Тип результата str

to_image(width, height)
Возвращает легенду в виде растра.

Параметры

- **width** (int) – Ширина выходного растра.

- **height** (int) – Высота выходного растра.

Тип результата QImage

16.1.7.5 LegendItem - Элемент легенды

class axipy.render.**LegendItem**
Элемент легенды.

Attributes:

style	Стиль оформления элемента легенды.
title	Описание элемента легенды.
visible	Видимость элемента легенды.

property style
Стиль оформления элемента легенды.

Тип результата Style

property title
Описание элемента легенды.

Тип результата str

property visible
Видимость элемента легенды.

Тип результата bool

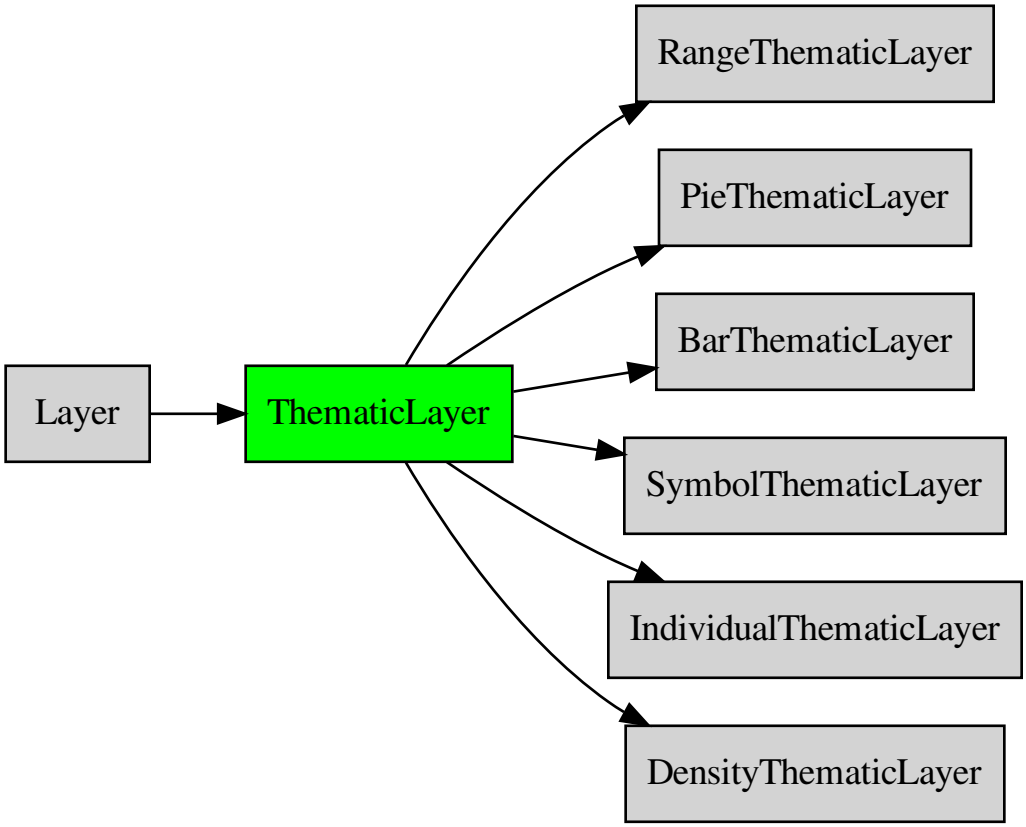
16.1.7.6 ListLegendItems - Список элементов легенды

class axipy.render.**ListLegendItems**
Элементы легенды.

16.1.7.7 axipy.render thematic

ThematicLayer - Тематика

Иерархия классов тематик:



class axipy.render.**ThematicLayer**

Базовые классы: axipy.render.Layer

Абстрактный класс слоя с тематическим оформлением векторного слоя карты на базе атрибутивной информации.

ReallocateThematicColor - Распределение цветов

class axipy.render.**ReallocateThematicColor**

Базовые классы: object

Поддержка различного рода алгоритмов распределения оформления.

Methods:

assign_gray([minV, maxV])	Распределение в виде градации серого.
assign_monotone(color[, minv, maxv])	Монотонная заливка разной яркости (оттенки красного, синего и т.п.).
assign_rainbow([sequential, saturation, value])	Распределение цветов по спектру.
assign_three_colors(colorMin, colorMax, ...)	Цвет, распределенный между тремя заданными цветами (с разрывом).
assign_two_colors(colorMin, colorMax[, useHSV])	Равномерно распределяет оформление по заданным крайним цветам.

assign_gray(minV=20, maxV=80)
Распределение в виде градации серого. Значение задается в интервале (0..100) от черного до белого.

Параметры

- **minV** (int) – Минимальное значение.
- **maxV** (int) – Максимальное значение.

assign_monotone(color, minv=20, maxv=80)
Монотонная заливка разной яркости (оттенки красного, синего и т.п.). Цветовая схема HSL. Максимальное и минимальное значения задаются в интервале (0..100).

Параметры

- **color** (QColor) – Базовый цвет.
- **minV** – Минимальное значение.
- **maxV** – Максимальное значение.

assign_rainbow(sequential=True, saturation=90, value=90)
Распределение цветов по спектру. Цветовая схема HSV.

Параметры

- **sequential** (bool) – Если True, то последовательное распределение цветов. В противном случае распределение случайно.
- **saturation** (float) – Яркость. Задается в интервале (0..100)
- **value** (float) – Насыщенность. Задается в интервале (0..100)

assign_three_colors(colorMin, colorMax, colorBreak, br, useHSV=True)
Цвет, распределенный между тремя заданными цветами (с разрывом).

Параметры

- **colorMin** (QColor) – Цвет нижнего диапазона.
- **colorMax** (QColor) – Цвет верхнего диапазона.

- **colorBreak** (QColor) – Цвет на уровне разрыва.
- **br** (int) – Индекс интервала, на на котором используется цвет разрыва.
- **useHSV** (bool) – Если True, то будет использоваться схема HSV. В противном случае - RGB.

assign_two_colors(colorMin, colorMax, useHSV=False)

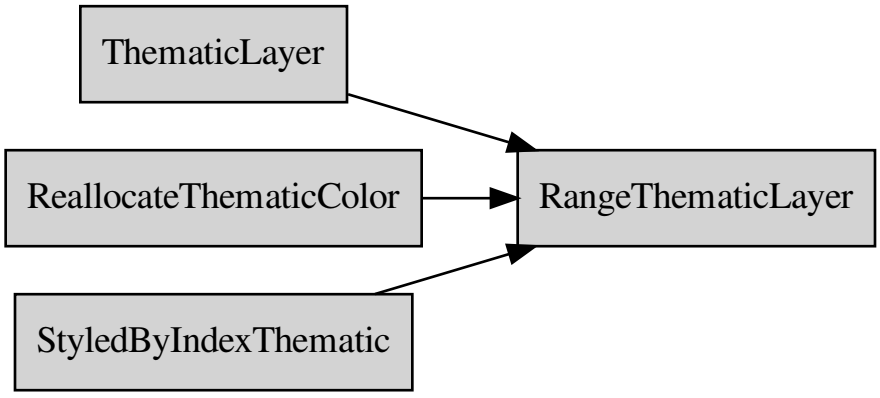
Равномерно распределяет оформление по заданным крайним цветам.

Параметры

- **colorMin** (QColor) – Цвет нижнего диапазона.
- **colorMax** (QColor) – Цвет верхнего диапазона.
- **useHSV** (bool) – Если True, то будет использоваться схема HSV. В противном случае - RGB.

RangeThematicLayer - Интервалы

Иерархия классов:



class axipy.render.**RangeThematicLayer**(expression)

Базовые классы: axipy.render.ThematicLayer, axipy.render.StyledByIndexThematic, axipy.render.ReallocateThematicColor

Тематическое оформление слоя с распределением значений по интервалам. Для распределения цветов по заданным интервалам могут быть использованы функции assign_* класса ReallocateThematicColor в зависимости от требуемых целей.

Параметры expression (str) – Наименование атрибута таблицы или выражение.

Список 199: Пример создания тематики по интервалам.

```
# Пример создания тематики с последующим добавлением ее к базовому слою `world`
range1 = RangeThematicLayer("Население")
range1.ranges = 6
range1.splitType = RangeThematicLayer.EQUAL_COUNT
range1.assign_two_colors(Qt.red, Qt.cyan)
world.thematic.add(range1)
# Пример запроса с последующей заменой::
v = world.thematic[0].get_interval_value(2) # Запрос
v = (999, v[1]) # Заменим минимальное значение для интервала с индексом 2
world.thematic[0].set_interval_value(2, v) # Замена
# Различные виды распределения интервалов тематик по цветам
range1.assign_two_colors(Qt.red, Qt.yellow)
range1.assign_three_colors(Qt.yellow, Qt.cyan, Qt.green, 4)
range1.assign_rainbow()
range1.assign_gray(80, 100)
```

Methods:

assign_gray([minV, maxV])	Распределение в виде градации серого.
assign_monotone(color[, minv, maxv])	Монотонная заливка разной яркости (оттенки красного, синего и т.п.).
assign_rainbow([sequential, saturation, value])	Распределение цветов по спектру.
assign_three_colors(colorMin, colorMax, ...)	Цвет, распределенный между тремя заданными цветами (с разрывом).
assign_two_colors(colorMin, colorMax[, useHSV])	Равномерно распределяет оформление по заданным крайним цветам.
create(dataObject)	Создает слой на базе открытой таблицы или растра.
get_bounds()	Возвращает область, в которую попадают все данные, которые могут быть отображены на слое.
get_interval_value(idx)	Возвращает предельные значения для указанного интервала в виде пары значений.
get_style(idx)	Стиль для указанного выражения.
set_interval_value(idx, v)	Заменяет предельные значения интервала.
set_style(idx, style)	Установка стиля оформления для выражения по его индексу в списке выражений.

Attributes:

coordsystem	Координатная система, в которой находятся данные, отображаемые слоем.
data_changed	Signal[] Сигнал об изменении контента слоя.

continues on next page

Таблица 178 – продолжение с предыдущей страницы

data_object	Источник данных для слоя.
is_valid	Проверка на валидность объекта.
max_zoom	Максимальная ширина окна, при котором слой отображается на карте.
min_zoom	Минимальная ширина окна, при котором слой отображается на карте.
need_redraw	Signal[] Сигнал о необходимости перерисовать слой.
opacity	Прозрачность слоя в составе карты.
ranges	Количество интервалов.
splitType	Тип распределения значений по интервалам.
title	Наименование слоя.
visible	Управляет видимостью слоя.
zoom_restrict	Будет ли использоваться ограничение по отображению.

assign_gray(minV=20, maxV=80)
Распределение в виде градации серого. Значение задается в интервале (0..100) от черного до белого.

Параметры

- **minV** (int) – Минимальное значение.
- **maxV** (int) – Максимальное значение.

assign_monotone(color, minv=20, maxv=80)
Монотонная заливка разной яркости (оттенки красного, синего и т.п.). Цветовая схема HSL. Максимальное и минимальное значения задаются в интервале (0..100).

Параметры

- **color** (QColor) – Базовый цвет.
- **minV** – Минимальное значение.
- **maxV** – Максимальное значение.

assign_rainbow(sequential=True, saturation=90, value=90)
Распределение цветов по спектру. Цветовая схема HSV.

Параметры

- **sequential** (bool) – Если True, то последовательное распределение цветов. В противном случае распределение случайно.
- **saturation** (float) – Яркость. Задается в интервале (0..100)
- **value** (float) – Насыщенность. Задается в интервале (0..100)

assign_three_colors(colorMin, colorMax, colorBreak, br, useHSV=True)
Цвет, распределенный между тремя заданными цветами (с разрывом).

Параметры

- **colorMin** (QColor) – Цвет нижнего диапазона.
- **colorMax** (QColor) – Цвет верхнего диапазона.

- **colorBreak** (QColor) – Цвет на уровне разрыва.
- **br** (int) – Индекс интервала, на на котором используется цвет разрыва.
- **useHSV** (bool) – Если True, то будет использоваться схема HSV. В противном случае - RGB.

assign_two_colors(colorMin, colorMax, useHSV=False)
Равномерно распределяет оформление по заданным крайним цветам.

Параметры

- **colorMin** (QColor) – Цвет нижнего диапазона.
- **colorMax** (QColor) – Цвет верхнего диапазона.
- **useHSV** (bool) – Если True, то будет использоваться схема HSV. В противном случае - RGB.

property coordsystem
Координатная система, в которой находятся данные, отображаемые слоем.

Тип результата CoordSystem

classmethod create(dataObject)
Создает слой на базе открытой таблицы или растра.

Параметры dataObject (DataObject) – Таблица или растр. В зависимости от переданного объекта будет создан VectorLayer или RasterLayer.

Список 200: Пример создания слоя на базе файла.

```
# Векторный слой
table = provider_manager.openfile(filepath)
vector_layer = Layer.create(table)
# Подпишемся на обновление контента слоя
vector_layer.need_redraw.connect(lambda: print('Update layer'))
```

Тип результата Layer

property data_changed
Signal[] Сигнал об изменении контента слоя.

Тип результата Signal

property data_object
Источник данных для слоя.

Тип результата DataObject

get_bounds()
Возвращает область, в которую попадают все данные, которые могут быть отображены на слое.

Тип результата Rect

get_interval_value(idx)
Возвращает предельные значения для указанного интервала в виде пары значений.

Параметры idx (int) – Индекс диапазона.

Тип результата Tuple[float, float]

get_style(idx)
Стиль для указанного выражения.

Параметры **idx** (int) – Порядковый номер выражения.

Тип результата Style

property is_valid
Проверка на валидность объекта. Слой мог быть удален, как пример, в связи с закрытием таблицы

Тип результата bool

property max_zoom
Максимальная ширина окна, при котором слой отображается на карте. Учитывается только при установленном zoom_restrict=True

Тип результата float

property min_zoom
Минимальная ширина окна, при котором слой отображается на карте. Учитывается только при установленном zoom_restrict=True

Тип результата float

property need_redraw
Signal[] Сигнал о необходимости перерисовать слой.

Тип результата Signal

property opacity
Прозрачность слоя в составе карты. Доступные значения от 0 до 100.

Тип результата int

property ranges
Количество интервалов.

Тип результата int

set_interval_value(idx, v)
Заменяет предельные значения интервала.

Параметры

- **idx** (int) – Индекс диапазона.
- **v** (Tuple[float, float]) – Значение в виде пары.

set_style(idx, style)
Установка стиля оформления для выражения по его индексу в списке выражений.

Параметры

- **idx** (int) – Индекс.
- **style** (Style) – Назначаемый стиль.

Список 201: Пример установки стиля для значения с индексом 2 первого тематического слоя.

```
style_new = Style.from_mapinfo("Brush (2, 255, 0)")
world.thematic[0].set_style(2, style_new)
```

property splitType

Тип распределения значений по интервалам.

Таблица 179: Допустимые значения:

Константа	Значение	Описание
EQUAL_INTERVAL	1	Распределение исходя из равномерности интервалов (по умолчанию).
EQUAL_COUNT	2	Распределение исходя их равного количества объектов в каждом интервале.
MANUAL	3	Ручное распределение значений путем задания пределов вручную.

Тип результата int

property title

Наименование слоя.

Тип результата str

property visible

Управляет видимостью слоя.

Выключение видимости верхнего слоя для активной карты:

```
if view_manager.active is not None:
    view_manager.active.map.layers[0].visible = False
```

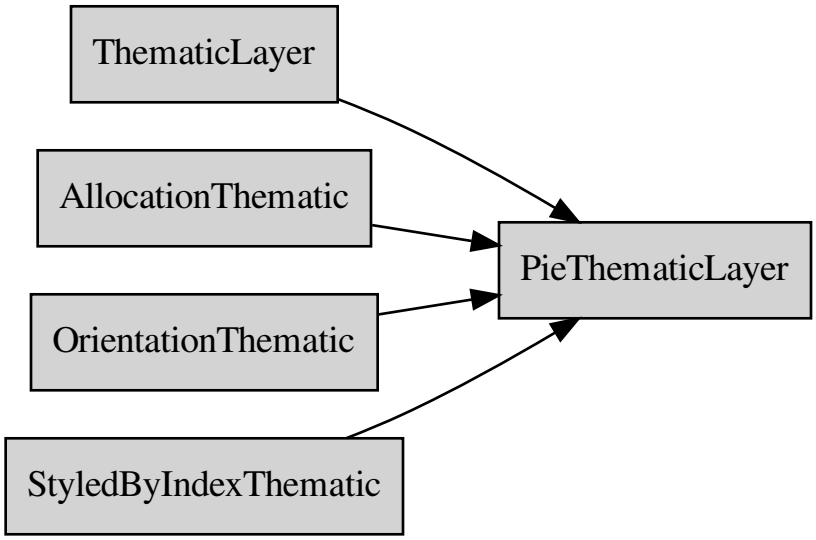
property zoom_restrict

Будет ли использоваться ограничение по отображению. Если установлено True, то для ограничения отображения слоя в зависимости от масштаба используются значения свойств zoom_min и zoom_max

Тип результата bool

PieThematicLayer - Круговые диаграммы

Иерархия классов:



class axipy.render.**PieThematicLayer**(expressions)

Базовые классы: axipy.render.ThematicLayer, axipy.render.AllocationThematic, axipy.render.OrientationThematic, axipy.render.StyledByIndexThematic

Тематика в виде круговых диаграмм.

Параметры expressions (List) – Наименования атрибутов или выражений в виде списка list.

Список 202: Создание тематики с последующим добавлением ее к базовому слою.

```
pie = PieThematicLayer(["Население", "Мужское", "Женское"])
pie.allocationType = PieThematicLayer.SQRT
style_lay_pie = Style.from_mapinfo("Brush (8, 65535, 0)")
# Заменим стиль
pie.set_style (0, style_lay_pie)
# Добавляем к основному слою
world.thematic.add(pie)
```

Attributes:

allocationType	Тип распределения значений.

continues on next page

Таблица 180 – продолжение с предыдущей страницы

coordsystem	Координатная система, в которой находятся данные, отображаемые слоем.
data_changed	Signal[] Сигнал об изменении контента слоя.
data_object	Источник данных для слоя.
is_valid	Проверка на валидность объекта.
max_zoom	Максимальная ширина окна, при котором слой отображается на карте.
min_zoom	Минимальная ширина окна, при котором слой отображается на карте.
need_redraw	Signal[] Сигнал о необходимости перерисовать слой.
opacity	Прозрачность слоя в составе карты.
orientationType	Ориентация относительно центроида.
startAngle	Начальный угол отсчета диаграммы.
title	Наименование слоя.
visible	Управляет видимостью слоя.
zoom_restrict	Будет ли использоваться ограничение по отображению.

Methods:

create(dataObject)	Создает слой на базе открытой таблицы или растра.
get_bounds()	Возвращает область, в которую попадают все данные, которые могут быть отображены на слое.
get_style(idx)	Стиль для указанного выражения.
set_style(idx, style)	Установка стиля оформления для выражения по его индексу в списке выражений.

property allocationType
Тип распределения значений.

Таблица 182: Допустимые значения.

Константа	Значение	Описание
LINEAR	1	Линейное (по умолчанию)
SQRT	2	Квадратичное
LOG10	3	Логарифмическое

Тип результата int

property coordsystem
Координатная система, в которой находятся данные, отображаемые слоем.

Тип результата CoordSystem

classmethod create(dataObject)
Создает слой на базе открытой таблицы или растра.

Параметры dataObject (DataObject) – Таблица или растр. В зависимости от переданного объекта будет создан VectorLayer или RasterLayer.

Список 203: Пример создания слоя на базе файла.

```
# Векторный слой
table = provider_manager.openfile(filepath)
vector_layer = Layer.create(table)
# Подпишемся на обновление контента слоя
vector_layer.need_redraw.connect(lambda: print('Update layer'))
```

Тип результата Layer

property data_changed
Signal[] Сигнал об изменении контента слоя.

Тип результата Signal

property data_object
Источник данных для слоя.

Тип результата DataObject

get_bounds()
Возвращает область, в которую попадают все данные, которые могут быть отображены на слое.

Тип результата Rect

get_style(idx)
Стиль для указанного выражения.

Параметры idx (int) – Порядковый номер выражения.

Тип результата Style

property is_valid
Проверка на валидность объекта. Слой мог быть удален, как пример, в связи с закрытием таблицы

Тип результата bool

property max_zoom
Максимальная ширина окна, при котором слой отображается на карте. Учитывается только при установленном zoom_restrict=True

Тип результата float

property min_zoom
Минимальная ширина окна, при котором слой отображается на карте. Учитывается только при установленном zoom_restrict=True

Тип результата float

property need_redraw
Signal[] Сигнал о необходимости перерисовать слой.

Тип результата Signal

property opacity
Прозрачность слоя в составе карты. Доступные значения от 0 до 100.

Тип результата int

property orientationType

Ориентация относительно центроида.

Таблица 183: Допустимые значения.

Константа	Значение	Описание
CENTER	0	Диаграмма рисуется по центру (по умолчанию)
LEFT_UP	1	Диаграмма выравнивается по левому верхнему краю
UP	2	Диаграмма выравнивается по верхнему краю
RIGHT_UP	3	Диаграмма выравнивается по верхнему правому краю
RIGHT	4	Диаграмма выравнивается по правому краю
RIGHT_DOWN	5	Диаграмма выравнивается по нижнему правому краю
DOWN	6	Диаграмма выравнивается по нижнему краю
LEFT_DOWN	7	Диаграмма выравнивается по нижнему левому краю
LEFT	8	Диаграмма выравнивается по левому краю

Тип результата int

set_style(idx, style)

Установка стиля оформления для выражения по его индексу в списке выражений.

Параметры

- **idx** (int) – Индекс.
- **style** (Style) – Назначаемый стиль.

Список 204: Пример установки стиля для значения с индексом 2 первого тематического слоя.

```
style_new = Style.from_mapinfo("Brush (2, 255, 0)")
world.thematic[0].set_style(2, style_new)
```

property startAngle

Начальный угол отсчета диаграммы.

Тип результата bool

property title

Наименование слоя.

Тип результата str

property visible

Управляет видимостью слоя.

Выключение видимости верхнего слоя для активной карты:

```
if view_manager.active is not None:
    view_manager.active.map.layers[0].visible = False
```

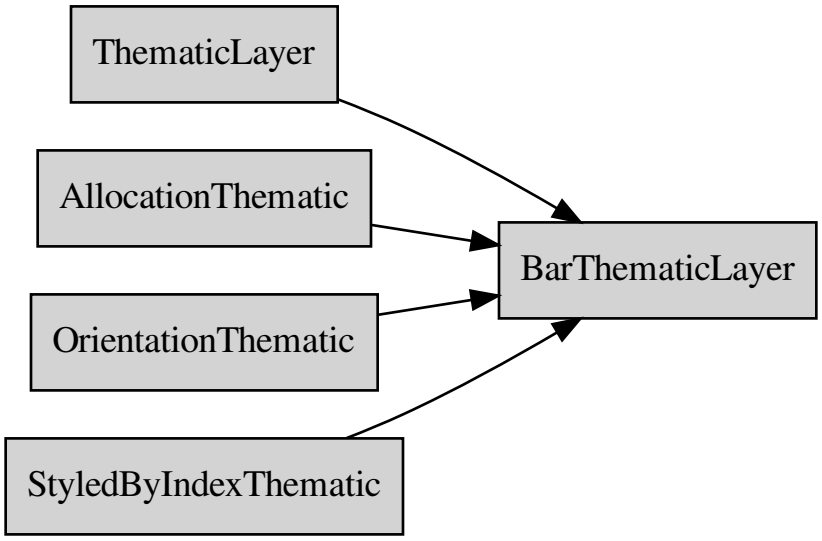
property zoom_restrict

Будет ли использоваться ограничение по отображению. Если установлено True, то для ограничения отображения слоя в зависимости от масштаба используются значения свойств zoom_min и zoom_max

Тип результата bool

BarThematicLayer - Столбчатые диаграммы

Иерархия классов:



class axipy.render.**BarThematicLayer**(expressions)

Базовые классы: axipy.render.ThematicLayer, axipy.render.AllocationThematic, axipy.render.OrientationThematic, axipy.render.StyledByIndexThematic

Тематика в виде столбчатых диаграмм.

Параметры expressions (List) – Наименования атрибутов или выражений в виде списка list.

Список 205: Создание тематики с последующим добавлением ее к базовому слою.

```
bar = BarThematicLayer(["Население", "Мужское", "Женское"])
# Добавляем к основному слою
world.thematic.add(bar)
```

Attributes:

allocationType	Тип распределения значений.
coordsystem	Координатная система, в которой находятся данные, отображаемые слоем.
data_changed	Signal[] Сигнал об изменении контента слоя.
data_object	Источник данных для слоя.
isStacked	Расположение столбчатой диаграммы в виде стопки, если True.
is_valid	Проверка на валидность объекта.
max_zoom	Максимальная ширина окна, при котором слой отображается на карте.
min_zoom	Минимальная ширина окна, при котором слой отображается на карте.
need_redraw	Signal[] Сигнал о необходимости перерисовать слой.
opacity	Прозрачность слоя в составе карты.
orientationType	Ориентация относительно центроида.
title	Наименование слоя.
visible	Управляет видимостью слоя.
zoom_restrict	Будет ли использоваться ограничение по отображению.

Methods:

create(dataObject)	Создает слой на базе открытой таблицы или растра.
get_bounds()	Возвращает область, в которую попадают все данные, которые могут быть отображены на слое.
get_style(idx)	Стиль для указанного выражения.
set_style(idx, style)	Установка стиля оформления для выражения по его индексу в списке выражений.

property allocationType

Тип распределения значений.

Таблица 186: Допустимые значения.

Константа	Значение	Описание
LINEAR	1	Линейное (по умолчанию)
SQRT	2	Квадратичное
LOG10	3	Логарифмическое

Тип результата int

property coordsystem

Координатная система, в которой находятся данные, отображаемые слоем.

Тип результата CoordSystem

classmethod create(dataObject)
Создает слой на базе открытой таблицы или растра.

Параметры dataObject (DataObject) – Таблица или растр. В зависимости от переданного объекта будет создан VectorLayer или RasterLayer.

Список 206: Пример создания слоя на базе файла.

```
# Векторный слой
table = provider_manager.openfile(filepath)
vector_layer = Layer.create(table)
# Подпишемся на обновление контента слоя
vector_layer.need_redraw.connect(lambda: print('Update layer'))
```

Тип результата Layer

property data_changed
Signal[] Сигнал об изменении контента слоя.

Тип результата Signal

property data_object
Источник данных для слоя.

Тип результата DataObject

get_bounds()
Возвращает область, в которую попадают все данные, которые могут быть отображены на слое.

Тип результата Rect

get_style(idx)
Стиль для указанного выражения.

Параметры idx (int) – Порядковый номер выражения.

Тип результата Style

property isStacked
Расположение столбчатой диаграммы в виде стопки, если True.

Тип результата bool

property is_valid
Проверка на валидность объекта. Слой мог быть удален, как пример, в связи с закрытием таблицы

Тип результата bool

property max_zoom
Максимальная ширина окна, при котором слой отображается на карте. Учитывается только при установленном zoom_restrict=True

Тип результата float

property min_zoom
Минимальная ширина окна, при котором слой отображается на карте. Учитывается только при установленном zoom_restrict=True

Тип результата float

property need_redraw
Signal[] Сигнал о необходимости перерисовать слой.

Тип результата Signal

property opacity
Прозрачность слоя в составе карты. Доступные значения от 0 до 100.

Тип результата int

property orientationType
Ориентация относительно центроида.

Таблица 187: Допустимые значения.

Константа	Значение	Описание
CENTER	0	Диаграмма рисуется по центру (по умолчанию)
LEFT_UP	1	Диаграмма выравнивается по левому верхнему краю
UP	2	Диаграмма выравнивается по верхнему краю
RIGHT_UP	3	Диаграмма выравнивается по верхнему правому краю
RIGHT	4	Диаграмма выравнивается по правому краю
RIGHT_DOWN	5	Диаграмма выравнивается по нижнему правому краю
DOWN	6	Диаграмма выравнивается по нижнему краю
LEFT_DOWN	7	Диаграмма выравнивается по нижнему левому краю
LEFT	8	Диаграмма выравнивается по левому краю

Тип результата int

set_style(idx, style)
Установка стиля оформления для выражения по его индексу в списке выражений.

Параметры

- **idx** (int) – Индекс.
- **style** (Style) – Назначаемый стиль.

Список 207: Пример установки стиля для значения с индексом 2 первого тематического слоя.

```
style_new = Style.from_mapinfo("Brush (2, 255, 0)")
world.thematic[0].set_style(2, style_new)
```

property title
Наименование слоя.

Тип результата str

property visible
Управляет видимостью слоя.

Выключение видимости верхнего слоя для активной карты:

```
if view_manager.active is not None:
    view_manager.active.map.layers[0].visible = False
```

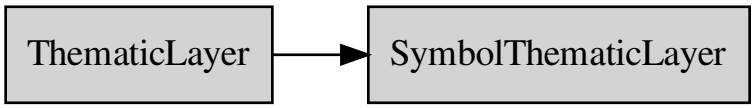
property zoom_restrict

Будет ли использоваться ограничение по отображению. Если установлено True, то для ограничения отображения слоя в зависимости от масштаба используются значения свойств zoom_min и zoom_max

Тип результата bool

SymbolThematicLayer - Знаки

Иерархия классов:



class axipy.render.**SymbolThematicLayer**(expression)

Базовые классы: axipy.render.ThematicLayer

Тематический слой с распределением по интервалам и с градуировкой символа по размеру.

Параметры expression (str) – Наименование атрибута или выражение.

Список 208: Создание тематики с последующим добавлением ее к базовому слою.

```
symbol = SymbolThematicLayer("Население")
symbol.defaultStyle = Style.from_mapinfo("Symbol (33, 255,14)")
symbol.maxHeight = 34
world.thematic.add(symbol)
```

Attributes:

coordsystem	Координатная система, в которой находятся данные, отображаемые слоем.
data_changed	Signal[] Сигнал об изменении контента слоя.
data_object	Источник данных для слоя.
defaultStyle	Стиль по умолчанию для оформления знаков.
is_valid	Проверка на валидность объекта.
maxHeight	Максимальная высота символа.

continues on next page

Таблица 188 – продолжение с предыдущей страницы

max_zoom	Максимальная ширина окна, при котором слой отображается на карте.
minHeight	Минимальная высота символа.
min_zoom	Минимальная ширина окна, при котором слой отображается на карте.
need_redraw	Signal[] Сигнал о необходимости перерисовать слой.
opacity	Прозрачность слоя в составе карты.
title	Наименование слоя.
visible	Управляет видимостью слоя.
zoom_restrict	Будет ли использоваться ограничение по отображению.

Methods:

create(dataObject)	Создает слой на базе открытой таблицы или растра.
get_bounds()	Возвращает область, в которую попадают все данные, которые могут быть отображены на слое.

property coordsystem
Координатная система, в которой находятся данные, отображаемые слоем.

Тип результата CoordSystem

classmethod create(dataObject)
Создает слой на базе открытой таблицы или растра.

Параметры dataObject (DataObject) – Таблица или растр. В зависимости от переданного объекта будет создан VectorLayer или RasterLayer.

Список 209: Пример создания слоя на базе файла.

```
# Векторный слой
table = provider_manager.openfile(filepath)
vector_layer = Layer.create(table)
# Подпишемся на обновление контента слоя
vector_layer.need_redraw.connect(lambda: print('Update layer'))
```

Тип результата Layer

property data_changed
Signal[] Сигнал об изменении контента слоя.

Тип результата Signal

property data_object
Источник данных для слоя.

Тип результата DataObject

property defaultStyle
Стиль по умолчанию для оформления знаков.

Тип результата Style

get_bounds()
Возвращает область, в которую попадают все данные, которые могут быть отображены на слое.

Тип результата Rect

property is_valid
Проверка на валидность объекта. Слой мог быть удален, как пример, в связи с закрытием таблицы

Тип результата bool

property maxHeight
Максимальная высота символа.

Тип результата float

property max_zoom
Максимальная ширина окна, при котором слой отображается на карте. Учитывается только при установленном zoom_restrict=True

Тип результата float

property minHeight
Минимальная высота символа.

Тип результата float

property min_zoom
Минимальная ширина окна, при котором слой отображается на карте. Учитывается только при установленном zoom_restrict=True

Тип результата float

property need_redraw
Signal[] Сигнал о необходимости перерисовать слой.

Тип результата Signal

property opacity
Прозрачность слоя в составе карты. Доступные значения от 0 до 100.

Тип результата int

property title
Наименование слоя.

Тип результата str

property visible
Управляет видимостью слоя.

Выключение видимости верхнего слоя для активной карты:

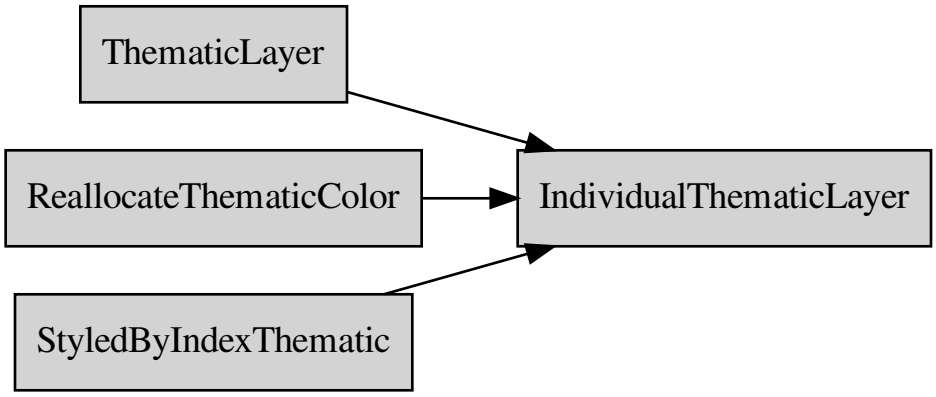
```
if view_manager.active is not None:
    view_manager.active.map.layers[0].visible = False
```

property zoom_restrict
Будет ли использоваться ограничение по отображению. Если установлено True, то для ограничения отображения слоя в зависимости от масштаба используются значения свойств zoom_min и zoom_max

Тип результата bool

IndividualThematicLayer - Индивидуальные значения

Иерархия классов:



class axipy.render.**IndividualThematicLayer**(expression)

Базовые классы: axipy.render.ThematicLayer, axipy.render.StyledByIndexThematic, axipy.render.ReallocateThematicColor

Тематический слой с распределением стилей по индивидуальным значением.

Параметры expression (str) – Наименование атрибута или выражение.

Список 210: Создание тематики с последующим добавлением ее к базовому слою.

```
individual = IndividualThematicLayer("Страна")
individual.assign_rainbow()
world.thematic.add(individual)
# Поменяем стиль оформления
individual.set_style(0, PolygonStyle(45, Qt.blue))
```

Methods:

assign_gray([minV, maxV])	Распределение в виде градации серого.
assign_monotone(color[, minv, maxv])	Монотонная заливка разной яркости (оттенки красного, синего и т.п.).
assign_rainbow([sequential, saturation, value])	Распределение цветов по спектру.
assign_three_colors(colorMin, colorMax, ...)	Цвет, распределенный между тремя заданными цветами (с разрывом).
assign_two_colors(colorMin, colorMax[, useHSV])	Равномерно распределяет оформление по заданным крайним цветам.
create(dataObject)	Создает слой на базе открытой таблицы или растра.

continues on next page

Таблица 190 – продолжение с предыдущей страницы

get_bounds()	Возвращает область, в которую попадают все данные, которые могут быть отображены на слое.
get_style(idx)	Стиль для указанного выражения.
get_value(idx)	Выражение по указанному индексу.
set_style(idx, style)	Установка стиля оформления для выражения по его индексу в списке выражений.

Attributes:

coordsystem	Координатная система, в которой находятся данные, отображаемые слоем.
count	Количество значений в тематике.
data_changed	Signal[] Сигнал об изменении контента слоя.
data_object	Источник данных для слоя.
is_valid	Проверка на валидность объекта.
max_zoom	Максимальная ширина окна, при котором слой отображается на карте.
min_zoom	Минимальная ширина окна, при котором слой отображается на карте.
need_redraw	Signal[] Сигнал о необходимости перерисовать слой.
opacity	Прозрачность слоя в составе карты.
title	Наименование слоя.
visible	Управляет видимостью слоя.
zoom_restrict	Будет ли использоваться ограничение по отображению.

assign_gray(minV=20, maxV=80)
Распределение в виде градации серого. Значение задается в интервале (0..100) от черного до белого.

Параметры

- **minV** (int) – Минимальное значение.
- **maxV** (int) – Максимальное значение.

assign_monotone(color, minv=20, maxv=80)
Монотонная заливка разной яркости (оттенки красного, синего и т.п.). Цветовая схема HSL. Максимальное и минимальное значения задаются в интервале (0..100).

Параметры

- **color** (QColor) – Базовый цвет.
- **minV** – Минимальное значение.
- **maxV** – Максимальное значение.

assign_rainbow(sequential=True, saturation=90, value=90)
Распределение цветов по спектру. Цветовая схема HSV.

Параметры

- **sequential** (bool) – Если True, то последовательное распределение цветов. В противном случае распределение случайно.
- **saturation** (float) – Яркость. Задается в интервале (0..100)
- **value** (float) – Насыщенность. Задается в интервале (0..100)

assign_three_colors(colorMin, colorMax, colorBreak, br, useHSV=True)
Цвет, распределенный между тремя заданными цветами (с разрывом).

Параметры

- **colorMin** (QColor) – Цвет нижнего диапазона.
- **colorMax** (QColor) – Цвет верхнего диапазона.
- **colorBreak** (QColor) – Цвет на уровне разрыва.
- **br** (int) – Индекс интервала, на на котором используется цвет разрыва.
- **useHSV** (bool) – Если True, то будет использоваться схема HSV. В противном случае - RGB.

assign_two_colors(colorMin, colorMax, useHSV=False)
Равномерно распределяет оформление по заданным крайним цветам.

Параметры

- **colorMin** (QColor) – Цвет нижнего диапазона.
- **colorMax** (QColor) – Цвет верхнего диапазона.
- **useHSV** (bool) – Если True, то будет использоваться схема HSV. В противном случае - RGB.

property coordsystem
Координатная система, в которой находятся данные, отображаемые слоем.

Тип результата CoordSystem

property count
Количество значений в тематике.

classmethod create(dataObject)
Создает слой на базе открытой таблицы или растра.

Параметры dataObject (DataObject) – Таблица или растр. В зависимости от переданного объекта будет создан VectorLayer или RasterLayer.

Список 211: Пример создания слоя на базе файла.

```
# Векторный слой
table = provider_manager.openfile(filepath)
vector_layer = Layer.create(table)
# Подпишемся на обновление контента слоя
vector_layer.need_redraw.connect(lambda: print('Update layer'))
```

Тип результата Layer

property data_changed
Signal[] Сигнал об изменении контента слоя.

Тип результата Signal

property data_object
Источник данных для слоя.

Тип результата DataObject

get_bounds()
Возвращает область, в которую попадают все данные, которые могут быть отображены на слое.

Тип результата Rect

get_style(idx)
Стиль для указанного выражения.

Параметры **idx** (int) – Порядковый номер выражения.

Тип результата Style

get_value(idx)
Выражение по указанному индексу.

Параметры **idx** (int) – Индекс.

Тип результата Any

property is_valid
Проверка на валидность объекта. Слой мог быть удален, как пример, в связи с закрытием таблицы

Тип результата bool

property max_zoom
Максимальная ширина окна, при котором слой отображается на карте. Учитывается только при установленном zoom_restrict=True

Тип результата float

property min_zoom
Минимальная ширина окна, при котором слой отображается на карте. Учитывается только при установленном zoom_restrict=True

Тип результата float

property need_redraw
Signal[] Сигнал о необходимости перерисовать слой.

Тип результата Signal

property opacity
Прозрачность слоя в составе карты. Доступные значения от 0 до 100.

Тип результата int

set_style(idx, style)
Установка стиля оформления для выражения по его индексу в списке выражений.

Параметры

- **idx** (int) – Индекс.

- **style** (Style) – Назначаемый стиль.

Список 212: Пример установки стиля для значения с индексом 2 первого тематического слоя.

```
style_new = Style.from_mapinfo("Brush (2, 255, 0)")
world.thematic[0].set_style(2, style_new)
```

property title
Наименование слоя.

Тип результата str

property visible
Управляет видимостью слоя.

Выключение видимости верхнего слоя для активной карты:

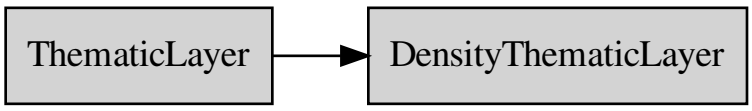
```
if view_manager.active is not None:
    view_manager.active.map.layers[0].visible = False
```

property zoom_restrict
Будет ли использоваться ограничение по отображению. Если установлено True, то для ограничения отображения слоя в зависимости от масштаба используются значения свойств zoom_min и zoom_max

Тип результата bool

DensityThematicLayer - Плотность точек

Иерархия классов:



class axipy.render.**DensityThematicLayer**(expression)

Базовые классы: axipy.render.ThematicLayer

Тематический слой с заполнением полигональных объектов точками, плотность которых зависит от вычисленного значения по выражению.

Параметры expression (str) – Наименование атрибута или выражение.

Список 213: Создание тематики с последующим добавлением ее к базовому слою.

```
density = DensityThematicLayer('Население')
density.pointForMaximum = 500
```

(continues on next page)

(продолжение с предыдущей страницы)

```
density.color = Qt.red
density.size = 1
world.thematic.add(density)
```

Attributes:

color	Цвет точек.
coordsystem	Координатная система, в которой находятся данные, отображаемые слоем.
data_changed	Signal[] Сигнал об изменении контента слоя.
data_object	Источник данных для слоя.
is_valid	Проверка на валидность объекта.
max_zoom	Максимальная ширина окна, при котором слой отображается на карте.
min_zoom	Минимальная ширина окна, при котором слой отображается на карте.
need_redraw	Signal[] Сигнал о необходимости перерисовать слой.
opacity	Прозрачность слоя в составе карты.
pointForMaximum	Количество точек для максимального значения.
size	Размер точек.
title	Наименование слоя.
visible	Управляет видимостью слоя.
zoom_restrict	Будет ли использоваться ограничение по отображению.

Methods:

create(dataObject)	Создает слой на базе открытой таблицы или растра.
get_bounds()	Возвращает область, в которую попадают все данные, которые могут быть отображены на слое.

property color
Цвет точек.

Тип результата QColor

property coordsystem
Координатная система, в которой находятся данные, отображаемые слоем.

Тип результата CoordSystem

classmethod create(dataObject)
Создает слой на базе открытой таблицы или растра.

Параметры dataObject (DataObject) – Таблица или растр. В зависимости от переданного объекта будет создан VectorLayer или RasterLayer.

Список 214: Пример создания слоя на базе файла.

```
# Векторный слой
table = provider_manager.openfile(filepath)
vector_layer = Layer.create(table)
# Подпишемся на обновление контента слоя
vector_layer.need_redraw.connect(lambda: print('Update layer'))
```

Тип результата Layer

property data_changed
Signal[] Сигнал об изменении контента слоя.

Тип результата Signal

property data_object
Источник данных для слоя.

Тип результата DataObject

get_bounds()
Возвращает область, в которую попадают все данные, которые могут быть отображены на слое.

Тип результата Rect

property is_valid
Проверка на валидность объекта. Слой мог быть удален, как пример, в связи с закрытием таблицы

Тип результата bool

property max_zoom
Максимальная ширина окна, при котором слой отображается на карте. Учитывается только при установленном zoom_restrict=True

Тип результата float

property min_zoom
Минимальная ширина окна, при котором слой отображается на карте. Учитывается только при установленном zoom_restrict=True

Тип результата float

property need_redraw
Signal[] Сигнал о необходимости перерисовать слой.

Тип результата Signal

property opacity
Прозрачность слоя в составе карты. Доступные значения от 0 до 100.

Тип результата int

property pointForMaximum
Количество точек для максимального значения.

Тип результата int

property size
Размер точек.

Тип результата float

property title
Наименование слоя.

Тип результата str

property visible
Управляет видимостью слоя.

Выключение видимости верхнего слоя для активной карты:

```
if view_manager.active is not None:
    view_manager.active.map.layers[0].visible = False
```

property zoom_restrict
Будет ли использоваться ограничение по отображению. Если установлено True, то для ограничения отображения слоя в зависимости от масштаба используются значения свойств zoom_min и zoom_max

Тип результата bool

AllocationThematic - Метод распределения значений для диаграмм

class axipy.render.**AllocationThematic**
Метод распределения значений для диаграмм.

Attributes:

allocationType	Тип распределения значений.

property allocationType
Тип распределения значений.

Таблица 195: Допустимые значения.

Константа	Значение	Описание
LINEAR	1	Линейное (по умолчанию)
SQRT	2	Квадратичное
LOG10	3	Логарифмическое

Тип результата int

OrientationThematic - Ориентация для диаграмм

class axipy.render.**OrientationThematic**
Ориентация тематического представления относительно центроида объекта.

Attributes:

orientationType	Ориентация относительно центроида.

property orientationType
Ориентация относительно центроида.

Таблица 197: Допустимые значения.

Константа	Значение	Описание
CENTER	0	Диаграмма рисуется по центру (по умолчанию)
LEFT_UP	1	Диаграмма выравнивается по левому верхнему краю
UP	2	Диаграмма выравнивается по верхнему краю
RIGHT_UP	3	Диаграмма выравнивается по верхнему правому краю
RIGHT	4	Диаграмма выравнивается по правому краю
RIGHT_DOWN	5	Диаграмма выравнивается по нижнему правому краю
DOWN	6	Диаграмма выравнивается по нижнему краю
LEFT_DOWN	7	Диаграмма выравнивается по нижнему левому краю
LEFT	8	Диаграмма выравнивается по левому краю

Тип результата int

StyledByIndexThematic - Стиль заливки

class axipy.render.**StyledByIndexThematic**

Поддержка набора индексированных стилей.

Methods:

get_style(idx)	Стиль для указанного выражения.
set_style(idx, style)	Установка стиля оформления для выражения по его индексу в списке выражений.

get_style(idx)

Стиль для указанного выражения.

Параметры **idx** (int) – Порядковый номер выражения.

Тип результата Style

set_style(idx, style)

Установка стиля оформления для выражения по его индексу в списке выражений.

Параметры

- **idx** (int) – Индекс.
- **style** (Style) – Назначаемый стиль.

Список 215: Пример установки стиля для значения с индексом 2 первого тематического слоя.

```
style_new = Style.from_mapinfo("Brush (2, 255, 0)")
world.thematic[0].set_style(2, style_new)
```

16.1.7.8 axipy.render report

Report - Отчет

class axipy.render.**Report**(printer)
План отчета для последующей печати.

Список 216: Пример создания пустого отчета и вывод его в pdf.

```
printer = QPrinter()
printer.setPageSize(QPageSize(QPageSize.A4))
printer.setOutputFormat(QPrinter.PdfFormat)
printer.setOutputFileName(filepath)
painterReport = QPainter(printer)
contextReport = Context(painterReport)
report = Report(printer)
report.horisontal_pages = 2
# Здесь добавляются элементы отчета
report.draw(contextReport)
```

Methods:

draw(context)	Выводит отчета в заданном контексте.
fill_on_pages()	Максимально заполняет страницу(ы) отчета.
fit_pages()	Подгоняет число страниц отчета под размер существующих элементов отчета.

Attributes:

horisontal_pages	Количество страниц отчета по горизонтали.
items	Элементы отчета.
name	Наименование отчета.
need_redraw	Signal[] Сигнал о необходимости перерисовки части или всего отчета.
page_size	Размеры одного листа отчета.
unit	Единицы измерения в отчете.
vertical_pages	Количество страниц отчета по вертикали.

draw(context)
Выводит отчета в заданном контексте.

Параметры context (Context) – Контекст, в котором будет отрисован отчет.

fill_on_pages()
Максимально заполняет страницу(ы) отчета. При этом элементы отчета пропорционально масштабируются.

fit_pages()
Подгоняет число страниц отчета под размер существующих элементов отчета.

При этом параметры элементов отчета не меняются.

property horisontal_pages
Количество страниц отчета по горизонтали.

Тип результата int

property items
Элементы отчета.

Тип результата ReportItems

property name
Наименование отчета.

Тип результата str

property need_redraw
Signal[] Сигнал о необходимости перерисовки части или всего отчета.

Параметры rect (Union[Rect, QRectF]) – Часть отчета, которую необходимо обновить.

Тип результата Signal

property page_size
Размеры одного листа отчета.

Тип результата QSizeF

property unit
Единицы измерения в отчете.

Тип результата LinearUnit

property vertical_pages
Количество страниц отчета по вертикали.

Тип результата int

ReportItems - Список элементов отчета

class axipy.render.**ReportItems**
Список элементов отчета.

add(item)
Добавляет новый элемент в отчет.

Параметры item (ReportItem) – Вставляемый элемент

at(idx)
Возвращает элемент отчета по его индексу.

Параметры idx (int) – Индекс.

Тип результата ReportItem

Результат Элемент отчета. Возвращает None в случае, если не найдено.

property count
Количество элементов отчета в текущем отчете на данный момент.

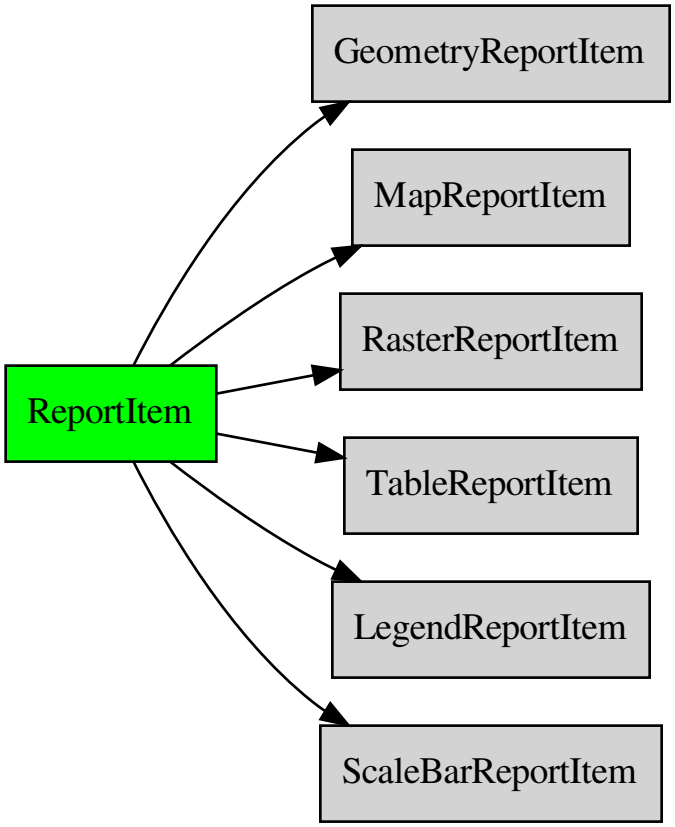
Тип результата int

remove(idx)
Удаляет элемент по его индексу. Если индекс корректен, элемент будет удален.

Параметры **idx** (int) – Индекс удаляемого элемента.

ReportItem - Элемент отчета

Элементы отчета:



class axipy.render.**ReportItem**
Базовый класс элемента отчета.

Attributes:

border_style	Стиль обводки элемента отчета.
fill_style	Стиль заливки элемента отчета.
rect	Размер (ограничивающий прямоугольник) элемента отчета в единицах измерения отчета.

Methods:

intersects(checkRect)	Пересекается ли с переданным прямоугольником.

property border_style
Стиль обводки элемента отчета.

Тип результата Style

property fill_style
Стиль заливки элемента отчета.

Тип результата Style

intersects(checkRect)
Пересекается ли с переданным прямоугольником.

Параметры checkRect (Union[Rect, QRectF]) – Прямоугольник для анализа.

property rect
Размер (ограничивающий прямоугольник) элемента отчета в единицах измерения отчета.

Тип результата Rect

GeometryReportItem - Элемент отчета: геометрия

class axipy.render.**GeometryReportItem**

Базовые классы: axipy.render.ReportItem

Элемент отчета типа геометрия.

Список 217: Пример создания полигона и добавления его в отчет.

```
geomItem = GeometryReportItem()
geomItem.geometry = Polygon((10, 10), (10, 100), (100, 100), (10, 10))
geomItem.style = PolygonStyle(45, Qt.red)
report.items.add(geomItem)
```

Список 218: Пример создания текста и добавления его в отчет.

```
r = Rect(8, 6, 14, 7)
txt = Text("Пример текста", r)
txt.angle = 20
style = Style.from_mapinfo('Font ("Times New Roman", 512, 0, 16711680, 16776960)')
geomItem = GeometryReportItem()
geomItem.style = style
geomItem.geometry = txt
report.items.add(geomItem)
```

Attributes:

border_style	Стиль обводки элемента отчета.
fill_style	Стиль заливки элемента отчета.
geometry	Геометрическое представление объекта.
rect	Размер (ограничивающий прямоугольник) элемента отчета в единицах измерения отчета.
style	Стиль геометрического представления объекта.

Methods:

intersects(checkRect)	Пересекается ли с переданным прямоугольником.

property border_style
Стиль обводки элемента отчета.

Тип результата Style

property fill_style
Стиль заливки элемента отчета.

Тип результата Style

property geometry
Геометрическое представление объекта.

Тип результата Geometry

intersects(checkRect)
Пересекается ли с переданным прямоугольником.

Параметры checkRect (Union[Rect, QRectF]) – Прямоугольник для анализа.

property rect
Размер (ограничивающий прямоугольник) элемента отчета в единицах измерения отчета.

Тип результата Rect

property style
Стиль геометрического представления объекта.

Тип результата Style

MapReportItem - Элемент отчета: карта

class axipy.render.**MapReportItem**(rect, map)

Базовые классы: axipy.render.ReportItem

Элемент отчета, основанный на созданной ранее карте.

Примечание: Перед созданием элемента отчета необходимо предварительно создать карту, на основе которой будет создан элемент отчета.

Параметры

- **rect** (Union[Rect, QRectF]) – Размер элемента отчета в единицах измерения отчета.
- **map** (Map) – Карта, на базе которой будет создан элемент отчета.

Список 219: Пример создания карты и добавления ее в отчет.

```
map_ = Map([world])
mapItem = MapReportItem(Rect(10, 110, 200, 210), map_)
mapItem.center = (100, 100)
mapItem.scale = 200000000
report.items.add(mapItem)
```

Attributes:

border_style	Стиль обводки элемента отчета.
center	Центр карты в координатах карты.
fill_style	Стиль заливки элемента отчета.
map_rect	Прямоугольник карты в единицах измерения карты.
rect	Размер (ограничивающий прямоугольник) элемента отчета в единицах измерения отчета.
scale	Текущее значение масштаба карты.

Methods:

intersects(checkRect)	Пересекается ли с переданным прямоугольником.
map()	Возвращает элемент типа карта, на основании которой создается элемент отчета.
show_all()	Меняет масштаб карты чтобы показать ее полностью.

property border_style

Стиль обводки элемента отчета.

Тип результата Style

property center
Центр карты в координатах карты.

Тип результата Pnt

property fill_style
Стиль заливки элемента отчета.

Тип результата Style

intersects(checkRect)
Пересекается ли с переданным прямоугольником.

Параметры checkRect (Union[Rect, QRectF]) – Прямоугольник для анализа.

map()
Возвращает элемент типа карта, на основании которой создается элемент отчета.

Тип результата Map

property map_rect
Прямоугольник карты в единицах измерения карты.

Тип результата Rect

property rect
Размер (ограничивающий прямоугольник) элемента отчета в единицах измерения отчета.

Тип результата Rect

property scale
Текущее значение масштаба карты.

Тип результата float

show_all()
Меняет масштаб карты чтобы показать ее полностью.

Пример замены масштаба для всех элементов отчета:

```
for item in reportView.report.items:
    if isinstance(item, MapReportItem):
        item.show_all()
```

RasterReportItem - Элемент отчета: растр

class axipy.render.**RasterReportItem**(rect, data)

Базовые классы: axipy.render.ReportItem

Элемент отчета, основанный на растре.

Примечание: В качестве источника может быть как локальный файл, расположенный в файловой системе, так и базе растра, размещенного на Web ресурсе.

Параметры

- **rect** (Union[Rect, QRectF]) – Размер элемента отчета в единицах измерения отчета.
- **data** (str) – Путь к растровому файлу или его URL.

Список 220: Пример элемента на базе URL.

```
report = create_report()
rasterReportItem = RasterReportItem(Rect(10, 10, 140, 70),
        'https://upload.wikimedia.org/wikipedia/commons/thumb/3/34/Gall%E2%80
↪%93Peters_projection_SW.jpg/1280px-Gall%E2%80%93Peters_projection_SW.jpg')
report.items.add(rasterReportItem)
```

Список 221: Пример элемента на базе локального файла.

```
rasterReportItem = RasterReportItem(Rect(10, 10, 140, 70), filename)
report.items.add(rasterReportItem)
```

Attributes:

border_style	Стиль обводки элемента отчета.
fill_style	Стиль заливки элемента отчета.
preserve_aspect_ratio	Сохранять пропорции при изменении размеров элемента.
rect	Размер (ограничивающий прямоугольник) элемента отчета в единицах измерения отчета.

Methods:

intersects(checkRect)	Пересекается ли с переданным прямоугольником.

property border_style
Стиль обводки элемента отчета.

Тип результата Style

property fill_style
Стиль заливки элемента отчета.

Тип результата Style

intersects(checkRect)
Пересекается ли с переданным прямоугольником.

Параметры checkRect (Union[Rect, QRectF]) – Прямоугольник для анализа.

property preserve_aspect_ratio
Сохранять пропорции при изменении размеров элемента.

Тип результата bool

property rect
Размер (ограничивающий прямоугольник) элемента отчета в единицах измерения отчета.

Тип результата Rect

TableReportItem - Элемент отчета: таблица

class axipy.render.**TableReportItem**(rect, table)

Базовые классы: axipy.render.ReportItem

Элемент отчета табличного представления данных.

Примечание: Позволяет отображать как таблицу целиком, так и накладывая дополнительные ограничения при отображении.

Параметры

- **rect** (Union[Rect, QRectF]) – Размер элемента отчета в единицах измерения отчета.
- **table** (Table) – Таблица.

Список 222: Пример.

```
table = provider_manager.openfile(filename)
tableReportItem = TableReportItem(Rect(210, 150, 480, 100), table)
tableReportItem.columns = table.schema.attribute_names[:3] # Берем для показа␣
↪первые три атрибута
tableReportItem.row_from = 5 # С 5-й сторки
tableReportItem.row_count = 4 # Показываем 4 строки
tableReportItem.start_number = 5 # Нумерация с 5
tableReportItem.border_style = LineStyle(3, Qt.red) # Стиль рамки
tableReportItem.fill_style = PolygonStyle(8, 65535) # Стиль фона
report.items.add(tableReportItem)
```

Attributes:

border_style	Стиль обводки элемента отчета.
columns	Перечень наименований для отображения.
fill_style	Стиль заливки элемента отчета.
rect	Размер (ограничивающий прямоугольник) элемента отчета в единицах измерения отчета.
row_count	Количество записей.
row_from	Номер первой строки из таблицы или запроса.
start_number	Нумерация записей.

Methods:

intersects(checkRect)	Пересекается ли с переданным прямоугольником.
refreshValues()	«Обновление данных из таблицы.

continues on next page

Таблица 210 – продолжение с предыдущей страницы

table()	Базовая таблица или запрос.

property border_style
Стиль обводки элемента отчета.

Тип результата Style

property columns
Перечень наименований для отображения. Если задать пустой список, будут отображены все поля таблицы.

Тип результата list

property fill_style
Стиль заливки элемента отчета.

Тип результата Style

intersects(checkRect)
Пересекается ли с переданным прямоугольником.

Параметры checkRect (Union[Rect, QRectF]) – Прямоугольник для анализа.

property rect
Размер (ограничивающий прямоугольник) элемента отчета в единицах измерения отчета.

Тип результата Rect

refreshValues()
«Обновление данных из таблицы.

property row_count
Количество записей. Если указано -1, то берутся все оставшиеся записи.

Тип результата int

property row_from
Номер первой строки из таблицы или запроса.

Тип результата int

property start_number
Нумерация записей. Порядковый номер первой записи.

Тип результата int

table()
Базовая таблица или запрос.

Тип результата Table

LegendReportItem - Элемент отчета: легенда

class axipy.render.**LegendReportItem**(rect, legend)

Базовые классы: axipy.render.ReportItem

Элемент отчета, основанный на легенде векторного или тематического слоя.

Параметры

- **rect** (Union[Rect, QRectF]) – Размер элемента отчета в единицах измерения отчета.
- **legend** (Legend) – Предварительно созданная легенда. Она может относиться как к векторному, так и к тематическому слою.

Список 223: Пример создания легенды для тематического слоя.

```
range_ = RangeThematicLayer("Население")
world.thematic.add(range_)
legend = Legend(range_)
legend.columns = 2 # Разобъьем на 2 колонки
legendReportItem = LegendReportItem(Rect(100, 230, 50, 70), legend) # Элемент␣
↪отчета
report.items.add(legendReportItem)  # Добавляем в отчет
```

Attributes:

border_style	Стиль обводки элемента отчета.
fill_style	Стиль заливки элемента отчета.
legend	Легенда на базе которой создан элемент отчета.
rect	Размер (ограничивающий прямоугольник) элемента отчета в единицах измерения отчета.

Methods:

intersects(checkRect)	Пересекается ли с переданным прямоугольником.

property border_style
Стиль обводки элемента отчета.

Тип результата Style

property fill_style
Стиль заливки элемента отчета.

Тип результата Style

intersects(checkRect)
Пересекается ли с переданным прямоугольником.

Параметры checkRect (Union[Rect, QRectF]) – Прямоугольник для анализа.

property legend
Легенда на базе которой создан элемент отчета.

Тип результата Legend

property rect
Размер (ограничивающий прямоугольник) элемента отчета в единицах измерения отчета.

Тип результата Rect

ScaleBarReportItem - Элемент отчета: масштабная линейка

class axipy.render.**ScaleBarReportItem**(rect, map)

Базовые классы: axipy.render.ReportItem

Элемент отчета - масштабная линейка для карты.

Список 224: Пример создания масштабной линейки на базе существующего элемента - карты.

```
scaleBarReportItem = ScaleBarReportItem(Rect(120, 130, 80, 50), mapItem)
report.items.add(scaleBarReportItem)
```

Attributes:

border_style	Стиль обводки элемента отчета.
fill_style	Стиль заливки элемента отчета.
rect	Размер (ограничивающий прямоугольник) элемента отчета в единицах измерения отчета.

Methods:

intersects(checkRect)	Пересекается ли с переданным прямоугольником.

property border_style
Стиль обводки элемента отчета.

Тип результата Style

property fill_style
Стиль заливки элемента отчета.

Тип результата Style

intersects(checkRect)
Пересекается ли с переданным прямоугольником.

Параметры checkRect (Union[Rect, QRectF]) – Прямоугольник для анализа.

property rect
Размер (ограничивающий прямоугольник) элемента отчета в единицах измерения отчета.

Тип результата Rect

16.1.7.9 Context - Контекст рисования

class axipy.render.**Context**(painter)

Контекст рисования.

Содержит информацию о том, куда производится рисование (QPainter), а так же о необходимых преобразованиях, которые необходимо применить к объекту непосредственно перед его отрисовкой.

Параметры painter (QPainter) – Объект QPainter для рисования.

Пример создания контекста на базе растра. Далее его можно использовать для отрисовки карты Map, отчета Report или легенды Legend:

```
image = QImage(1600, 800, QImage.Format_ARGB32_Premultiplied)
image.fill(Qt.white)
painter = QPainter(image)
context = Context(painter)
```

Attributes:

coordsystem	Координатная система.
dpi	Количество точек на дюйм, с которым происходит рисование.
rect	Прямоугольник в координатах карты, который будет отрисован.

property coordsystem

Координатная система.

Если она не задана, берется наиболее подходящая исходя из текущего контента.

Тип результата CoordSystem

property dpi

Количество точек на дюйм, с которым происходит рисование.

Влияет на отрисовку в «реальных» единицах измерения (мм, см, пункты).

Тип результата float

property rect

Прямоугольник в координатах карты, который будет отрисован.

Тип результата Rect

16.1.7.10 CustomLabels - Пользовательские метки карты

class axipy.render.**CustomLabels**
Пользовательские метки. Используется для задания параметров через свойство Map.custom_labels

Methods:

get(layer, id)	Производит запрос параметров.
set(layer, id, properties)	Устанавливает параметры.

get(layer, id)
Производит запрос параметров. Если для данного id не определены, возвращает None.

Параметры

- **layer** (VectorLayer) – Слой карты
- **id** (int) – Идентификатор записи

Тип результата Optional[CustomLabelProperties]

set(layer, id, properties)
Устанавливает параметры.

Параметры

- **layer** (VectorLayer) – Слой карты
- **id** (int) – Идентификатор записи
- **properties** (Optional[CustomLabelProperties]) – Устанавливаемые свойства. Если задать None, существующие параметры будут сброшены

16.1.8 axipy.gui

Модуль пользовательского интерфейса.

В данном модуле содержатся классы связанные с пользовательским интерфейсом.

axipy.gui.**view_manager**
Экземпляр менеджера содержимого окон.

Type ViewManager

axipy.gui.**selection_manager**
Экземпляр доступа к выделенным объектам.

Type SelectionManager

16.1.8.1 MapTool - Инструмент окна карты

class axipy.gui.**MapTool**

Инструмент окна карты.

При создании своего инструмента новый инструмент наследуется от этого класса, и переопределяет необходимые обработчики событий.

См.также:

axipy.da.StateManager.

PassEvent

Передать событие дальше. Значение False.

Type bool

BlockEvent

Прекратить обработку события. Значение True.

Type bool

enable_on

Идентификатор наблюдателя для определения доступности инструмента. По умолчанию отсутствует.

Type Union[str, DefaultKeys]

Пример:

```
MyTool(MapTool):

    def mousePressEvent(self, event):
        print('mouse pressed')
        return self.PassEvent
```

Methods:

canDeactivate(reason)	Обрабатывает причину выключения инструмента.
canUnload(reason)	Обрабатывает причину выключения инструмента.
deactivate()	Выполняет действия непосредственно перед выключением инструмента и перед его удалением.
get_select_rect(device[, size])	Возвращает прямоугольник в координатах карты для точки на экране.
handleEvent(event)	Первичный обработчик всех событий инструмента.
is_snapped()	Проверяет, сработала ли привязка к элементам карты или отчета для текущего положения указателя мыши.
keyPressEvent(event)	Обрабатывает событие нажатия клавиши клавиатуры.
keyReleaseEvent(event)	Обрабатывает событие отпускания клавиши клавиатуры.

continues on next page

Таблица 217 – продолжение с предыдущей страницы

load()	Выполняет действия непосредственно перед включением инструмента.
mouseDoubleClickEvent(event)	Обрабатывает событие двойного клика мыши.
mouseMoveEvent(event)	Обрабатывает событие перемещения мыши.
mousePressEvent(event)	Обрабатывает событие нажатия клавиши мыши.
mouseReleaseEvent(event)	Обрабатывает событие отпускания клавиши мыши.
paintEvent(event, painter)	Обрабатывает событие отрисовки.
redraw()	Перерисовывает окно карты.
reset()	Переключает текущий инструмент на инструмент по умолчанию.
snap([default_value])	Возвращает исправленные координаты, если сработала привязка к элементам в единицах измерения карты или отчета.
snap_device([default_value])	Возвращает исправленные координаты, если сработала привязка к элементам в единицах измерения окна карты (виджета).
to_device(scene)	Переводит точки из координат на карте в координаты окна(пиксели).
to_scene(device)	Переводит точки из координат окна(пикселей) в координаты на карте.
unload()	Выполняет действия непосредственно перед выключением инструмента и перед его удалением.
wheelEvent(event)	Обрабатывает событие колеса мыши.

Attributes:

cursor	Текущий курсор для данного инструмента.
view	Отображение данных в окне.

canDeactivate(reason)

Обрабатывает причину выключения инструмента.

Переопределите этот метод, чтобы задать свой обработчик.

Параметры **reason** (DeactivationReason) – причина выключения.

Тип результата bool

Результат False чтобы прервать выключение, иначе True.

Не рекомендуется, начиная с версии 3.6: Используйте canUnload().

canUnload(reason)

Обрабатывает причину выключения инструмента.

Переопределите этот метод, чтобы задать свой обработчик.

Параметры **reason** (DeactivationReason) – причина выключения.

Тип результата bool

Результат False чтобы прервать выключение, иначе True.

property cursor

Текущий курсор для данного инструмента.

Первоначально курсор для инструмента можно установить, переопределив метод load():

```
class MyTool(MapTool):
    def load(self):
        self.cursor = QCursor(Qt.SizeAllCursor)
```

Если же требуется устанавливать различный типы курсора в зависимости от статуса нажатия ПКМ, следует переопределить методы mousePressEvent() и mouseReleaseEvent() и установить нужное значение там:

```
class MyTool(MapTool):
    def mousePressEvent(self, event) -> bool:
        if event.button() == Qt.LeftButton:
            self.cursor = QCursor(Qt.SizeAllCursor)
```

Тип результата QCursor

deactivate()

Выполняет действия непосредственно перед выключением инструмента и перед его удалением.

Не рекомендуется, начиная с версии 3.6: Используйте unload().

get_select_rect(device, size=3)

Возвращает прямоугольник в координатах карты для точки на экране.

Удобно для использования при поиске объектов.

Параметры

- **device** (QPoint) – Точка в координатах окна.
- **size** (int) – Размер квадрата в пикселях.

Тип результата Rect

Результат Прямоугольник в координатах карты.

Пример:

```
device_point = event.pos()
bbox = self.get_select_rect(device_point, 30)
features = table.items(bbox=bbox)
```

handleEvent(event)

Первичный обработчик всех событий инструмента.

Если событие не блокируется этим обработчиком, то оно будет передано дальше в соответствующий специализированный обработчик mousePressEvent(), keyReleaseEvent() и прочие в зависимости от типа.

Параметры **event** (QEvent) – Событие.

Тип результата Optional[bool]

Результат BlockEvent, чтобы блокировать дальнейшую обработку события.

is_snapped()
Проверяет, сработала ли привязка к элементам карты или отчета для текущего положения указателя мыши.

См.также:

snap(), snap_device().

Тип результата bool

keyPressEvent(event)
Обрабатывает событие нажатия клавиши клавиатуры.

Параметры event (QKeyEvent) – Событие нажатия клавиши клавиатуры.

Тип результата Optional[bool]

Результат PassEvent, чтобы пропустить событие дальше по цепочке обработчиков. None или BlockEvent, чтобы блокировать дальнейшую обработку события.

keyReleaseEvent(event)
Обрабатывает событие отпускания клавиши клавиатуры.

Параметры event (QKeyEvent) – Событие отпускания клавиши клавиатуры.

Тип результата Optional[bool]

Результат PassEvent, чтобы пропустить событие дальше по цепочке обработчиков. None или BlockEvent, чтобы блокировать дальнейшую обработку события.

load()
Выполняет действия непосредственно перед включением инструмента.

Переопределите этот метод, чтобы задать свои действия.

См.также:

unload().

mouseDoubleClickEvent(event)
Обрабатывает событие двойного клика мыши.

Параметры event (QMouseEvent) – Событие двойного клика мыши.

Тип результата Optional[bool]

Результат PassEvent, чтобы пропустить событие дальше по цепочке обработчиков. None или BlockEvent, чтобы блокировать дальнейшую обработку события.

mouseMoveEvent(event)
Обрабатывает событие перемещения мыши.

Параметры event (QMouseEvent) – Событие перемещения мыши.

Тип результата Optional[bool]

Результат PassEvent или None, чтобы пропустить событие дальше по цепочке обработчиков. BlockEvent, чтобы блокировать дальнейшую обработку события.

mousePressEvent(event)
Обрабатывает событие нажатия клавиши мыши.

Параметры **event** (QMouseEvent) – Событие нажатия клавиши мыши.

Тип результата Optional[bool]

Результат PassEvent, чтобы пропустить событие дальше по цепочке обработчиков. None или BlockEvent, чтобы блокировать дальнейшую обработку события.

mouseReleaseEvent(event)
Обрабатывает событие отпускания клавиши мыши.

Параметры **event** (QMouseEvent) – Событие отпускания клавиши мыши.

Тип результата Optional[bool]

Результат PassEvent, чтобы пропустить событие дальше по цепочке обработчиков. None или BlockEvent, чтобы блокировать дальнейшую обработку события.

paintEvent(event, painter)
Обрабатывает событие отрисовки.

Параметры

- **event** (QPaintEvent) – Событие отрисовки.
- **painter** (QPainter) – QPainter для рисования поверх виджета

redraw()
Перерисовывает окно карты.

Создает событие PySide2.QtGui.QPaintEvent и помещает его в очередь обработки событий. Аналогично PySide2.QtWidgets.QWidget.update().

static reset()
Переключает текущий инструмент на инструмент по умолчанию.

Обычно инструментом по умолчанию является Выбор.

snap(default_value=None)
Возвращает исправленные координаты, если сработала привязка к элементам в единицах измерения карты или отчета.

Параметры **default_value** (Optional[Pnt]) – Значение по умолчанию.

Возвращает значение по умолчанию, если не сработала привязка к элементам карты или отчета.

Пример:

```
point = self.to_scene(event.pos())
current_point = self.snap(point)
```

См.также:

is_snapped(), snap_device(), to_scene().

Тип результата Optional[Pnt]

snap_device(default_value=None)
Возвращает исправленные координаты, если сработала привязка к элементам в единицах измерения окна карты (виджета).

Параметры default_value (Optional[QPoint]) – Значение по умолчанию.

Возвращает значение по умолчанию, если не сработала привязка к элементам карты или отчета.

Пример:

```
device_point = event.pos()
current_device_point = self.snap_device(device_point)
```

См.также:

is_snapped(), snap().

Тип результата Optional[QPoint]

to_device(scene)
Переводит точки из координат на карте в координаты окна(пиксели).

Параметры scene (Union[Pnt, Rect]) – Точки в координатах карты.

Тип результата Union[QPoint, QRect]

Результат Точки в координатах окна.

to_scene(device)
Переводит точки из координат окна(пикселей) в координаты на карте.

Параметры device (Union[QPoint, QRect]) – Точки в координатах окна.

Тип результата Union[Pnt, Rect]

Результат Точки в координатах карты.

unload()
Выполняет действия непосредственно перед выключением инструмента и перед его удалением.

Переопределите этот метод, чтобы задать свои действия.

См.также:

load().

property view
Отображение данных в окне.

Тип результата MapView

wheelEvent(event)
Обрабатывает событие колеса мыши.

Параметры event (QWheelEvent) – Событие колеса мыши.

Тип результата Optional[bool]

Результат PassEvent, чтобы пропустить событие дальше по цепочке обработчиков. None или BlockEvent, чтобы блокировать дальнейшую обработку события.

class axipy.gui.**DeactivationReason**(value)
Причина выключения инструмента.

Таблица 219: Значения

Значение	Наименование
Unknown	Не определено
ObjectClose	Закрытие объекта данных
WindowClose	Закрытие окна
ActionClick	Нажатие на действие
ActionShortcut	Вызов действия комбинацией клавиш
LayerClick	Нажатие на свойства слоя

16.1.8.2 ActiveToolPanel - Панель активного инструмента

class axipy.gui.**ActiveToolPanel**
Сервис предоставляющий доступ к панели активного инструмента.

Список 225: Пример использования.

```
service = ActiveToolPanel()
# Любой пользовательский графический элемент
widget = QWidget()

# Создаём обработчик для панели активного инструмента через который
# будем управлять панелью.
tool_panel = service.make_acceptable(
    title="Мой инструмент",
    observer_id=DefaultKeys.SelectionEditable,
    widget=widget)

# Подписываемся на сигнал отправляемый после нажатия на кнопку "Применить" в␣
↪панели
tool_panel.accepted.connect(lambda: print("Применяем изменения"))
```

Чтобы отобразить переданный ранее графический элемент нужно вызвать `activate()`. Например при нажатии на пользовательскую кнопку.

Панель активного инструмента созданная через `make_acceptable()` по умолчанию содержит кнопку «Применить». По нажатию на эту кнопку отсылается сигнал `accepted()` который можно обработать в пользовательском инструменте.

Панель активного инструмента созданная через `make_custom()` представляет из себя пустой контейнер в который можно поместить пользовательский графический элемент. Это дает больше свободы для реализации управления панелью активного инструмента.

Переданный идентификатор наблюдателя используется для управления видимостью и доступностью панели. Панель активного инструмента сразу закроется как только наблюдатель вернет False.

make_acceptable(title, observer_id, widget=None)
Создает экземпляр обработчика через который можно взаимодействовать с панелью активного инструмента. По умолчанию добавляются кнопки «Применить/Отменить».

Параметры

- **title** (str) – Заголовок панели активного инструмента. Обычно это название инструмента.
- **observer_id** (Union[str, Key]) – Идентификатор наблюдателя для управления видимостью и доступностью.
- **widget** (Optional[QWidget]) – Пользовательский виджет который будет отображаться в панели активного инструмента.

Тип результата AxipyAcceptableActiveToolHandler

make_custom(title, observer_id, widget=None)

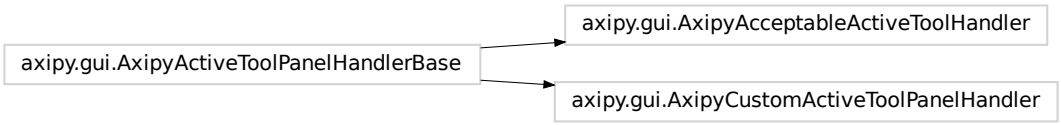
Создает экземпляр обработчика панели активного инструмента. В который можно установить любой пользовательский графический элемент. Используется когда пользователю не нужны предустановленные элементы управления.

Параметры

- **title** (str) – Заголовок панели активного инструмента. Обычно это название инструмента.
- **observer_id** (Union[str, Key]) – Идентификатор наблюдателя для управления видимостью и доступностью.
- **widget** (Optional[QWidget]) – Пользовательский виджет который будет отображаться в панели активного инструмента.

Тип результата AxipyCustomActiveToolPanelHandler

16.1.8.3 AxipyActiveToolPanelHandlerBase - Базовый класс обработчика панели активного инструмента



class axipy.gui.**AxipyActiveToolPanelHandlerBase**(shadow_handler)

Базовый класс обработчика панели активного инструмента.

Methods:

activate()	Показывает пользовательский графический элемент в панели активного инструмента.
deactivate()	Скрывает пользовательский графический элемент из панели активного инструмента.
set_observer(observer_id)	Метод устанавливает наблюдателя.

continues on next page

Таблица 220 – продолжение с предыдущей страницы

set_panel_title(title)	Устанавливает заголовок панели активного инструмента.
set_widget(widget)	Пользовательский графический элемент будет помещен в панель активного инструмента при активации обработчика.

Attributes:

activated	Signal[] Сигнал испускается когда обработчик панели активного инструмента становится активным.
deactivated	Signal[] Сигнал испускается перед тем как обработчик панели активного инструмента перестает быть активным.
panel_was_closed	Signal[] Сигнал испускается после закрытия панели активного инструмента
widget	Возвращает пользовательский графический элемент.

activate()
Показывает пользовательский графический элемент в панели активного инструмента.

property activated
Signal[] Сигнал испускается когда обработчик панели активного инструмента становится активным.

Тип результата Signal

deactivate()
Скрывает пользовательский графический элемент из панели активного инструмента.

property deactivated
Signal[] Сигнал испускается перед тем как обработчик панели активного инструмента перестает быть активным.

Тип результата Signal

property panel_was_closed
Signal[] Сигнал испускается после закрытия панели активного инструмента

Тип результата Signal

set_observer(observer_id)
Метод устанавливает наблюдателя. Если наблюдатель сигнализирует, что условия доступности кнопки нарушены, то панель активного инструмента сразу же закроется.

Параметры observer_id (Union[str, Key]) – Идентификатор наблюдателя для управления видимостью и доступностью

См.также:

Наблюдатели за состоянием инструмента observers

set_panel_title(title)
Устанавливает заголовок панели активного инструмента.

Параметры title (str) – Новый заголовок.

set_widget(widget)
Пользовательский графический элемент будет помещен в панель активного инструмента при активации обработчика. Владение графическим элементом передаётся обработчику. Это значит, что не следует использовать и сохранять где-либо ссылку на этот объект. Для получения графического элемента обратно используйте widget().

property widget
Возвращает пользовательский графический элемент.

Тип результата QWidget

Результат Переданный ранее пользовательский графический элемент.

16.1.8.4 AxipyAcceptableActiveToolHandler - Управление панелью активного инструмента с предустановленными кнопками

class axipy.gui.**AxipyAcceptableActiveToolHandler**(shadow_handler)

Базовые классы: axipy.gui.AxipyActiveToolPanelHandlerBase

Обработчик панели активного инструмента, который предоставляет по умолчанию кнопку Применить. При нажатии на эту кнопку испускается сигнал accepted().

Attributes:

accepted	Signal[] Отсылается после того как пользователь нажал кнопку «Применить» в панели активного инструмента.
activated	Signal[] Сигнал испускается когда обработчик панели активного инструмента становится активным.
deactivated	Signal[] Сигнал испускается перед тем как обработчик панели активного инструмента перестает быть активным.
panel_was_closed	Signal[] Сигнал испускается после закрытия панели активного инструмента
widget	Возвращает пользовательский графический элемент.

Methods:

activate()	Показывает пользовательский графический элемент в панели активного инструмента.
blockSignals(self, b)	

continues on next page

Таблица 223 – продолжение с предыдущей страницы

childEvent(self, event)	
children(self)	
connect(arg__1, arg__2, arg__3, type)	connect(arg__1: bytes, arg__2: typing.Callable, type: PySide2.QtCore.Qt.ConnectionType = PySide2.QtCore.Qt.ConnectionType.AutoConnection) -> bool connect(arg__1: bytes, arg__2: PySide2.QtCore.QObject, arg__3: bytes, type: PySide2.QtCore.Qt.ConnectionType = PySide2.QtCore.Qt.ConnectionType.AutoConnection) -> bool connect(sender: PySide2.QtCore.QObject, signal: PySide2.QtCore.QMetaMethod, receiver: PySide2.QtCore.QObject, method: PySide2.QtCore.QMetaMethod, type: PySide2.QtCore.Qt.ConnectionType = PySide2.QtCore.Qt.ConnectionType.AutoConnection) -> PySide2.QtCore.QMetaObject.Connection connect(sender: PySide2.QtCore.QObject, signal: bytes, member: bytes, type: PySide2.QtCore.Qt.ConnectionType = PySide2.QtCore.Qt.ConnectionType.AutoConnection) -> PySide2.QtCore.QMetaObject.Connection connect(sender: PySide2.QtCore.QObject, signal: bytes, receiver: PySide2.QtCore.QObject, member: bytes, type: PySide2.QtCore.Qt.ConnectionType = PySide2.QtCore.Qt.ConnectionType.AutoConnection) -> PySide2.QtCore.QMetaObject.Connection
connectNotify(self, signal)	
customEvent(self, event)	
deactivate()	Скрывает пользовательский графический элемент из панели активного инструмента.
deleteLater(self)	
disable()	Отключает доступность блока с кнопкой Применить.

continues on next page

Таблица 223 – продолжение с предыдущей страницы

disconnect(arg__1)	disconnect(arg__1: PySide2.QtCore.QObject, arg__2: bytes, arg__3: typing.Callable) -> bool disconnect(arg__1: bytes, arg__2: typing.Callable) -> bool disconnect(receiver: PySide2.QtCore.QObject, member: typing.Union[bytes, NoneType] = None) -> bool disconnect(sender: PySide2.QtCore.QObject, signal: PySide2.QtCore.QMetaMethod, receiver: PySide2.QtCore.QObject, member: PySide2.QtCore.QMetaMethod) -> bool disconnect(sender: PySide2.QtCore.QObject, signal: bytes, receiver: PySide2.QtCore.QObject, member: bytes) -> bool disconnect(signal: bytes, receiver: PySide2.QtCore.QObject, member: bytes) -> bool
disconnectNotify(self, signal)	
dumpObjectInfo(self)	
dumpObjectTree(self)	
dynamicPropertyNames(self)	
emit(self, arg__1, *args)	
event(self, event)	
eventFilter(self, watched, event)	
findChild(self, arg__1, arg__2)	
findChildren(self, arg__1, arg__2)	findChildren(self, arg__1: type, arg__2: str = „“) -> typing.Iterable
inherits(self, classname)	
installEventFilter(self, filterObj)	
isSignalConnected(self, signal)	
isWidgetType(self)	
isWindowType(self)	
killTimer(self, id)	

continues on next page

Таблица 223 – продолжение с предыдущей страницы

metaObject(self)	
moveToThread(self, thread)	
objectName(self)	
parent(self)	
property(self, name)	
receivers(self, signal)	
registerUserData()	
removeEventFilter(self, obj)	
sender(self)	
senderSignalIndex(self)	
setObjectName(self, name)	
setParent(self, parent)	
setProperty(self, name, value)	
set_observer(observer_id)	Метод устанавливает наблюдателя.
set_panel_title(title)	Устанавливает заголовок панели активного инструмента.
set_widget(widget)	Пользовательский графический элемент будет помещен в панель активного инструмента при активации обработчика.
signalsBlocked(self)	
startTimer(self, interval, timerType)	
thread(self)	
timerEvent(self, event)	
tr(self, arg__1, arg__2, arg__3)	
try_enable()	Включает доступность блока с кнопкой Применить если наблюдатель, связанный с панелью активного инструмента, подтверждает доступность.

property accepted
: `Signal[]` Отсылается после того как пользователь нажал кнопку «Применить» в панели активного инструмента.

Тип результата Signal

activate()
Показывает пользовательский графический элемент в панели активного инструмента.

property activated
Signal[] Сигнал испускается когда обработчик панели активного инструмента становится активным.

Тип результата Signal

blockSignals(self, b: bool) → bool

childEvent(self, event: PySide2.QtCore.QChildEvent)

children(self) → typing.List[PySide2.QtCore.QObject]

static connect(arg__1: PySide2.QtCore.QObject, arg__2: bytes, arg__3: typing.Callable, type: PySide2.QtCore.Qt.ConnectionType = PySide2.QtCore.Qt.ConnectionType.AutoConnection) → bool
connect(arg__1: bytes, arg__2: typing.Callable, type: PySide2.QtCore.Qt.ConnectionType = PySide2.QtCore.Qt.ConnectionType.AutoConnection) -> bool connect(arg__1: bytes, arg__2: PySide2.QtCore.QObject, arg__3: bytes, type: PySide2.QtCore.Qt.ConnectionType = PySide2.QtCore.Qt.ConnectionType.AutoConnection) -> bool connect(sender: PySide2.QtCore.QObject, signal: PySide2.QtCore.QMetaMethod, receiver: PySide2.QtCore.QObject, method: PySide2.QtCore.QMetaMethod, type: PySide2.QtCore.Qt.ConnectionType = PySide2.QtCore.Qt.ConnectionType.AutoConnection) -> PySide2.QtCore.QMetaObject.Connection connect(sender: PySide2.QtCore.QObject, signal: bytes, member: bytes, type: PySide2.QtCore.Qt.ConnectionType = PySide2.QtCore.Qt.ConnectionType.AutoConnection) -> PySide2.QtCore.QMetaObject.Connection connect(sender: PySide2.QtCore.QObject, signal: bytes, receiver: PySide2.QtCore.QObject, member: bytes, type: PySide2.QtCore.Qt.ConnectionType = PySide2.QtCore.Qt.ConnectionType.AutoConnection) -> PySide2.QtCore.QMetaObject.Connection

connectNotify(self, signal: PySide2.QtCore.QMetaMethod)

customEvent(self, event: PySide2.QtCore.QEvent)

deactivate()
Скрывает пользовательский графический элемент из панели активного инструмента.

property deactivated
Signal[] Сигнал испускается перед тем как обработчик панели активного инструмента перестает быть активным.

Тип результата Signal

deleteLater(self)

disable()
Отключает доступность блока с кнопкой Применить. Если инструмент запускает фоновые задачи с использованием TaskManager, то следует вызвать эту функцию перед началом выполнения задачи. Иначе у пользователя может быть возможность добавить множество одинаковых задач, несколько раз нажав на кнопку.

static disconnect(arg__1: PySide2.QtCore.QMetaObject.Connection) → bool
disconnect(arg__1: PySide2.QtCore.QObject, arg__2: bytes, arg__3: typing.Callable) -> bool disconnect(arg__1: bytes, arg__2: typing.Callable) -> bool disconnect(receiver: PySide2.QtCore.QObject, member: typing.Union[bytes, NoneType] = None) -> bool disconnect(sender: PySide2.QtCore.QObject, signal: PySide2.QtCore.QMetaMethod, receiver: PySide2.QtCore.QObject, member: PySide2.QtCore.QMetaMethod) -> bool disconnect(sender: PySide2.QtCore.QObject, signal: bytes, receiver: PySide2.QtCore.QObject, member: bytes) -> bool disconnect(signal: bytes, receiver: PySide2.QtCore.QObject, member: bytes) -> bool

disconnectNotify(self, signal: PySide2.QtCore.QMetaMethod)

dumpObjectInfo(self)

dumpObjectTree(self)

dynamicPropertyNames(self) → typing.List[PySide2.QtCore.QByteArray]

emit(self, arg__1: bytes, *args: None) → bool

event(self, event: PySide2.QtCore.QEvent) → bool

eventFilter(self, watched: PySide2.QtCore.QObject, event: PySide2.QtCore.QEvent) → bool

findChild(self, arg__1: type, arg__2: str = '') → object

findChildren(self, arg__1: type, arg__2: PySide2.QtCore.QRegExp) → typing.Iterable
findChildren(self, arg__1: type, arg__2: str = „“) -> typing.Iterable

inherits(self, classname: bytes) → bool

installEventFilter(self, filterObj: PySide2.QtCore.QObject)

isSignalConnected(self, signal: PySide2.QtCore.QMetaMethod) → bool

isWidgetType(self) → bool

isWindowType(self) → bool

killTimer(self, id: int)

metaObject(self) → PySide2.QtCore.QMetaObject

moveToThread(self, thread: PySide2.QtCore.QThread)

objectName(self) → str

property panel_was_closed
`Signal[]` Сигнал испускается после закрытия панели активного инструмента

Тип результата `Signal`

parent(self) → PySide2.QtCore.QObject

property(self, name: bytes) → typing.Any

receivers(self, signal: bytes) → int

static registerUserData() → int

removeEventFilter(self, obj: PySide2.QtCore.QObject)

sender(self) → PySide2.QtCore.QObject

senderSignalIndex(self) → int

setObjectName(self, name: str)

setParent(self, parent: PySide2.QtCore.QObject)

setProperty(self, name: bytes, value: typing.Any) → bool

set_observer(observer_id)
Метод устанавливает наблюдателя. Если наблюдатель сигнализирует, что условия доступности кнопки нарушены, то панель активного инструмента сразу же закроется.

Параметры observer_id (Union[str, Key]) – Идентификатор наблюдателя для управления видимостью и доступностью

См.также:

Наблюдатели за состоянием инструмента `observers`

set_panel_title(title)
Устанавливает заголовок панели активного инструмента.

Параметры title (str) – Новый заголовок.

set_widget(widget)
Пользовательский графический элемент будет помещен в панель активного инструмента при активации обработчика. Владение графическим элементом передаётся обработчику. Это значит, что не следует использовать и сохранять где-либо ссылку на этот объект. Для получения графического элемента обратно используйте `widget()`.

signalsBlocked(self) → bool

startTimer(self, interval: int, timerType: PySide2.QtCore.Qt.TimerType = PySide2.QtCore.Qt.TimerType.CoarseTimer) → int

thread(self) → PySide2.QtCore.QThread

timerEvent(self, event: PySide2.QtCore.QTimerEvent)

tr(self, arg__1: bytes, arg__2: bytes = b'', arg__3: int = - 1) → str

try_enable()
Включает доступность блока с кнопкой Применить если наблюдатель, связанный с панелью активного инструмента, подтверждает доступность.

property widget
Возвращает пользовательский графический элемент.

Тип результата QWidget

Результат Переданный ранее пользовательский графический элемент.

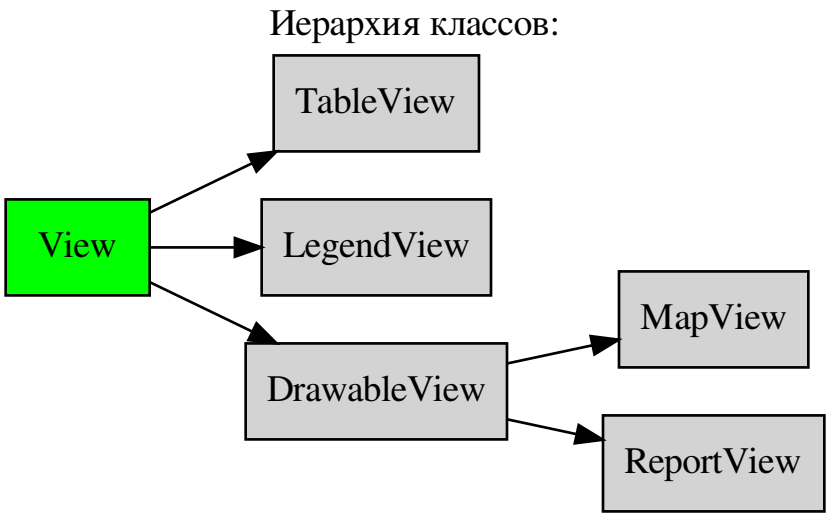
16.1.8.5 AxipyCustomActiveToolPanelHandler - Управление панелью активного инструмента без предустановленных кнопок управления

class axipy.gui.**AxipyCustomActiveToolPanelHandler**(shadow_handler)

Базовые классы: axipy.gui.AxipyActiveToolPanelHandlerBase

Обработчик панели активного инструмента который не предоставляет никаких встроенных по умолчанию элементов управления.

16.1.8.6 View - Базовый класс для отображения данных в окне



class axipy.gui.**View**

Базовый класс для отображения данных в окне.

Methods:

close()	Закрывает окно.
show([type])	Показывает окно в соответствие с приведенным типом.

Attributes:

position	Размер и положение окна.
rect	Размер и положение окна.
show_type	Возвращает тип состояния окна.
title	Заголовок окна просмотра.
widget	Виджет, соответствующий содержимому окна.

close()

Закрывает окно.

property position

Размер и положение окна.

Тип результата QRect

property rect

Размер и положение окна.

> **Предупреждение:** Не рекомендуется, начиная с версии 4.0: Используйте position.

Тип результата QRect

show(type=1)
Показывает окно в соответствие с приведенным типом.

Таблица 226: Допустимые значения:

Константа	Значение	Описание
SHOW_NORMAL	1	Обычный показ окна (по умолчанию).
SHOW_MINIMIZED	2	Показ окна в режиме минимизации.
SHOW_MAXIMIZED	3	Показ окна в режиме распахивания.

property show_type
Возвращает тип состояния окна. Подробнее см. show()

Тип результата int

property title
Заголовок окна просмотра.

Тип результата str

property widget
Виджет, соответствующий содержимому окна.

Тип результата QWidget

Результат Qt5 виджет содержимого.

16.1.8.7 TableView - Таблица просмотра атрибутивной информации

class axipy.gui.**TableView**

Базовые классы: axipy.gui.View

Таблица просмотра атрибутивной информации. Для создания экземпляра необходимо использовать axipy.gui.ViewManager.create_tableview() через экземпляр view_manager

Methods:

close()	Закрывает окно.
show([type])	Показывает окно в соответствие с приведенным типом.

Attributes:

data_object	Таблица, на основании которой создается данное окно просмотра.
position	Размер и положение окна.
rect	Размер и положение окна.

continues on next page

Таблица 228 – продолжение с предыдущей страницы

show_type	Возвращает тип состояния окна.
table_view	Ссылка на объект таблицы просмотра.
title	Заголовок окна просмотра.
widget	Виджет, соответствующий содержимому окна.

close()
Закрывает окно.

property data_object
Таблица, на основании которой создается данное окно просмотра.

Тип результата DataObject

Результат Таблица.

property position
Размер и положение окна.

Тип результата QRect

property rect
Размер и положение окна.

Предупреждение: Не рекомендуется, начиная с версии 4.0: Используйте position.

Тип результата QRect

show(type=1)
Показывает окно в соответствие с приведенным типом.

Таблица 229: Допустимые значения:

Константа	Значение	Описание
SHOW_NORMAL	1	Обычный показ окна (по умолчанию).
SHOW_MINIMIZED	2	Показ окна в режиме минимизации.
SHOW_MAXIMIZED	3	Показ окна в режиме распахивания.

property show_type
Возвращает тип состояния окна. Подробнее см. show()

Тип результата int

property table_view
Ссылка на объект таблицы просмотра.

Пример установки сортировки для таблицы в текущем окне по второй колонке по возрастанию:

```
if isinstance(view_manager.active, TableView):
    view_manager.active.table_view.sortByColumn(2, Qt.AscendingOrder)
```

Тип результата QTableView

property title
Заголовок окна просмотра.

Тип результата str

property widget
Виджет, соответствующий содержимому окна.

Тип результата QWidget

Результат Qt5 виджет содержимого.

16.1.8.8 DrawableView - Базовый класс с поддержкой визуального редактирования геометрий

class axipy.gui.**DrawableView**

Базовые классы: axipy.gui.View

«Базовый класс для визуализации геометрических данных.

Attributes:

can_redo	Возможен ли откат на один шаг вперед.
can_undo	Возможен ли откат на один шаг назад.
is_modified	Есть ли изменения в окне.
position	Размер и положение окна.
rect	Размер и положение окна.
scene_changed	«Сигнал об изменении контента окна.
show_type	Возвращает тип состояния окна.
snap_mode	Включает режим привязки координат при редактировании геометрии в окне карты или отчета.
title	Заголовок окна просмотра.
widget	Виджет, соответствующий содержимому окна.

Methods:

close()	Закрывает окно.
redo()	Производит откат на один шаг вперед.
scale_with_center(scale, center)	Установка нового центра с заданным масштабированием.
show([type])	Показывает окно в соответствие с приведенным типом.
undo()	Производит откат на один шаг назад.

property can_redo
Возможен ли откат на один шаг вперед.

Тип результата bool

property can_undo
Возможен ли откат на один шаг назад.

Тип результата bool

close()
Закрывает окно.

property is_modified
Есть ли изменения в окне.

Тип результата bool

property position
Размер и положение окна.

Тип результата QRect

property rect
Размер и положение окна.

> **Предупреждение:** Не рекомендуется, начиная с версии 4.0: Используйте position.

Тип результата QRect

redo()
Производит откат на один шаг вперед. При этом возвращается состояние до последней отмены.

scale_with_center(scale, center)
Установка нового центра с заданным масштабированием.

Параметры

- **scale** (float) – Коэффициент масштабирования по отношению к текущему.
- **center** (Pnt) – Устанавливаемый центр.

property scene_changed
«Сигнал об изменении контента окна.

Тип результата Signal

show(type=1)
Показывает окно в соответствие с приведенным типом.

Таблица 232: Допустимые значения:

Константа	Значение	Описание
SHOW_NORMAL	1	Обычный показ окна (по умолчанию).
SHOW_MINIMIZED	2	Показ окна в режиме минимизации.
SHOW_MAXIMIZED	3	Показ окна в режиме распахивания.

property show_type
Возвращает тип состояния окна. Подробнее см. show()

Тип результата int

property snap_mode
Включает режим привязки координат при редактировании геометрии в окне карты или отчета.

Тип результата bool

property title
Заголовок окна просмотра.

Тип результата str

undo()
Производит откат на один шаг назад.

property widget
Виджет, соответствующий содержимому окна.

Тип результата QWidget

Результат Qt5 виджет содержимого.

16.1.8.9 MapView - Окно просмотра карты

class axipy.gui.**MapView**

Базовые классы: axipy.gui.DrawableView

Окно просмотра карты. Используется для проведения различных манипуляций с картой. Для создания экземпляра необходимо использовать axipy.gui.ViewManager.create_mapview() через экземпляр view_manager (пример см. ниже).

Примечание: При создании „MapView“ посредством axipy.gui.ViewManager.create_mapview() производится клонирование экземпляра карты axipy.render.Map и для последующей работы при доступе к данному объекту необходимо использовать свойство map.

Свойство device_rect определяет размер самого окна карты, а свойство scene_rect - прямоугольную область, которая умещается в этом окне в СК карты.

Преобразование между этими двумя прямоугольниками производится с помощью матриц трансформации scene_to_device_transform и device_to_scene_transform.

К параметрам самой карты можно получить доступ через свойство map. Единицы измерения координат (unit) также берутся из наиболее подходящей СК, но при желании они могут быть изменены. К примеру, вместо метров могут быть установлены километры.

Рассмотрим пример создания карты с последующим помещением ее в окно ее просмотра. Далее, попробуем преобразовать объект типа полигон из координат окна экрана в координаты СК слоя посредством axipy.da.Geometry.affine_transform().

```
# Откроем таблицу, создадим на ее базе слой и добавим в карту
table_world = provider_manager.openfile('world.tab')
world = Layer.create(table_world)
map = Map([ world ])
# Для полученной карты создадим окно просмотра
mapview = view_manager.create_mapview(map)
# Выведем полученные параметры отображения
print('Прямоугольник экрана:', mapview.device_rect)
print('Прямоугольник карты:', mapview.scene_rect)
# Установим ширину карты и ее центр
mapview.center = (1000000, 1000000)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
mapview.set_zoom(10e6 )

>>> Прямоугольник экрана: (0.0 0.0) (300.0 200.0)
>>> Прямоугольник карты: (-16194966.287183324 -8621185.324024437) (16789976.
↪633236416 8326222.646170927)
```

```
#Создадим геометрический объект полигон в координатах экрана и преобразуем его в␣
↪СК карты
poly_device = Polygon([(100,100), (100,150), (150, 150), (150,100)])
# Используя матрицу трансформации, преобразуем его в координаты карты.
poly_scene = poly_device.affine_transform(mapview.device_to_scene_transform)
# Для контроля выведем полученный полигон в виде WKT
print('WKT:', poly_scene.wkt)

>>> WKT: POLYGON ((-5199985.31371008 -147481.338926755, -5199985.31371008 -
↪4384333.3314756, 297505.173026545 -4384333.3314756, 297505.173026545 -147481.
↪338926755, -5199985.31371008 -147481.338926755))
```

Attributes:

can_redo	Возможен ли откат на один шаг вперед.
can_undo	Возможен ли откат на один шаг назад.
center	Центр окна карты.
coordsystem	Система координат карты.
coordsystem_changed	Сигнал о том, что система координат изменилась.
coordsystem_visual	Система координат карты с учетом поправки цены градуса по широте.
device_rect	Видимая область в координатах окна (пиксели).
device_to_scene_transform	Объект трансформации из координат окна в координаты карты.
editable_layer	Редактируемый слой на карте.
editable_layer_changed	Сигнал о том, что редактируемый слой сменился.
is_modified	Есть ли изменения в окне.
map	Объект карты.
position	Размер и положение окна.
rect	Размер и положение окна.
scale	Масштаб карты.
scene_changed	«Сигнал об изменении контента окна.
scene_rect	Видимая область в координатах карты (в единицах измерения СК).
scene_to_device_transform	Объект трансформации из координат карты в координаты окна.
selected_layer	Выделенный слой на карте.
show_type	Возвращает тип состояния окна.
snap_mode	Включает режим привязки координат при редактировании геометрии в окне карты или отчета.
title	Заголовок окна просмотра.
unit	Единицы измерения координат карты.

continues on next page

Таблица 233 – продолжение с предыдущей страницы

widget	Виджет, соответствующий содержимому окна.

Methods:

close()	Закрывает окно.
mouse_moved(x, y)	Сигнал при смещении курсора мыши.
redo()	Производит откат на один шаг вперед.
scale_with_center(scale, center)	Установка нового центра с заданным масштабированием.
set_zoom(zoom[, unit])	«Задание ширины окна карты.
set_zoom_and_center(zoom, center[, unit])	«Задает новый центр и ширину окна карты.
show([type])	Показывает окно в соответствие с приведенным типом.
show_all()	Полностью показывает все слои карты.
undo()	Производит откат на один шаг назад.
zoom([unit])	Ширина окна карты.

property can_redo
Возможен ли откат на один шаг вперед.

Тип результата bool

property can_undo
Возможен ли откат на один шаг назад.

Тип результата bool

property center
Центр окна карты.

Тип результата Pnt

close()
Закрывает окно.

property coordsystem
Система координат карты.

Тип результата CoordSystem

property coordsystem_changed
Сигнал о том, что система координат изменилась.

Пример:

```
layer_world = Layer.create(table_world)
map = Map([ layer_world ])
mapview = view_manager.create_mapview(map)
mapview.coordsystem_changed.connect(lambda: print('СК была изменена'))
csLL = CoordSystem.from_prj("1, 104")
mapview.coordsystem = csLL

>>> СК была изменена
```

Тип результата Signal

property coordsystem_visual
Система координат карты с учетом поправки цены градуса по широте. Отличается от coordsystem только лишь в случае, когда основная система координат - это широта/долгота и широта имеет ненулевое значение. При этом в диапазоне широты (-70...70) градусов вводится поправочный коэффициент, растягивающий изображение по широте и равный 1/cos(y).

Тип результата CoordSystem

property device_rect
Видимая область в координатах окна (пиксели).

Тип результата Rect

Результат Прямоугольник в координатах окна.

property device_to_scene_transform
Объект трансформации из координат окна в координаты карты.

Тип результата QTransform

Результат Объект трансформации.

property editable_layer
Редактируемый слой на карте.

Тип результата Optional[VectorLayer]

Результат Редактируемый слой. Если не определен, возвращает None.

property editable_layer_changed
Сигнал о том, что редактируемый слой сменился.

Тип результата Signal

property is_modified
Есть ли изменения в окне.

Тип результата bool

property map
Объект карты.

Тип результата Map

Результат Карта.

mouse_moved(x, y)
Сигнал при смещении курсора мыши. Возвращает значения в СК карты.

Параметры

- **x** (float) – X координата
- **y** (float) – Y координата

Пример:

```
mapview.mouse_moved.connect(lambda x,y: print('Coords: {} {}'.format(x, y)))

>> Coords: -5732500.0 958500.0
>> Coords: -5621900.0 847900.0
```

Тип результата Signal

property position
Размер и положение окна.

Тип результата QRect

property rect
Размер и положение окна.

> **Предупреждение:** Не рекомендуется, начиная с версии 4.0: Используйте position.

Тип результата QRect

redo()
Производит откат на один шаг вперед. При этом возвращается состояние до последней отмены.

property scale
Масштаб карты.

Тип результата float

scale_with_center(scale, center)
Установка нового центра с заданным масштабированием.

Параметры

- **scale** (float) – Коэффициент масштабирования по отношению к текущему.
- **center** (Pnt) – Устанавливаемый центр.

property scene_changed
«Сигнал об изменении контента окна.

Тип результата Signal

property scene_rect
Видимая область в координатах карты (в единицах измерения СК).

Тип результата Rect

Результат Прямоугольник в координатах карты.

property scene_to_device_transform
Объект трансформации из координат карты в координаты окна.

Тип результата QTransform

Результат Объект трансформации.

property selected_layer
Выделенный слой на карте.

Тип результата Optional[VectorLayer]

Результат Выделенный слой. Если выделение отсутствует, возвращает None.

set_zoom(zoom, unit=None)
«Задание ширины окна карты.

Параметры

- **zoom** (float) – Значение ширины карты.
- **unit** (Optional[LinearUnit]) – Единицы измерения. Если не заданы, берутся текущие для карты.

set_zoom_and_center(zoom, center, unit=None)
«Задает новый центр и ширину окна карты.

Параметры

- **zoom** (float) – Значение ширины карты.
- **center** (Pnt) – Центр карты.
- **unit** (Optional[LinearUnit]) – Единицы измерения. Если не заданы, берутся текущие для карты.

show(type=1)
Показывает окно в соответствие с приведенным типом.

Таблица 235: Допустимые значения:

Константа	Значение	Описание
SHOW_NORMAL	1	Обычный показ окна (по умолчанию).
SHOW_MINIMIZED	2	Показ окна в режиме минимизации.
SHOW_MAXIMIZED	3	Показ окна в режиме распахивания.

show_all()
Полностью показывает все слои карты.

property show_type
Возвращает тип состояния окна. Подробнее см. show()

Тип результата int

property snap_mode
Включает режим привязки координат при редактировании геометрии в окне карты или отчета.

Тип результата bool

property title
Заголовок окна просмотра.

Тип результата str

undo()
Производит откат на один шаг назад.

property unit
Единицы измерения координат карты.

Тип результата LinearUnit

property widget
Виджет, соответствующий содержимому окна.

Тип результата QWidget

Результат Qt5 виджет содержимого.

zoom(unit=None)
Ширина окна карты.

Параметры **unit** (Optional[LinearUnit]) – Единицы измерения. Если не заданы, берутся текущие для карты.

Тип результата float

16.1.8.10 ReportView - Окно просмотра отчета

class axipy.gui.**ReportView**

Базовые классы: axipy.gui.DrawableView

Окно с планом отчета. Для создания экземпляра необходимо использовать axipy.gui.ViewManager.create_reportview() через экземпляр view_manager. До параметров самого отчета axipy.render.Report можно доступиться через свойство ReportView.report()

Пример создания отчета:

```
reportview = view_manager.create_reportview()

# Добавим полигон
geomItem = GeometryReportItem()
geomItem.geometry = Polygon((10,10), (10, 100), (100, 100), (10, 10))
geomItem.style = PolygonStyle(45, Qt.red)
reportview.report.items.add(geomItem)

# Установим текущий масштаб
reportview.view_scale = 33
```

Attributes:

can_redo	Возможен ли откат на один шаг вперед.
can_undo	Возможен ли откат на один шаг назад.
is_modified	Есть ли изменения в окне.
mesh_size	Размер ячейки сетки.
position	Размер и положение окна.
rect	Размер и положение окна.
report	Объект отчета.
scene_changed	«Сигнал об изменении контента окна.
show_borders	Показывать границы страниц.
show_mesh	Показывать сетку привязки.
show_ruler	Показывать линейку по краям.
show_type	Возвращает тип состояния окна.
snap_mode	Включает режим привязки координат при редактировании геометрии в окне карты или отчета.
snap_to_guidelines	Включение режима притяжения элементов отчета к направляющим.
snap_to_mesh	Включение режима притяжения элементов отчета к узлам сетки.
title	Заголовок окна просмотра.
view_scale	Текущий масштаб.
widget	Виджет, соответствующий содержимому окна.

continues on next page

Таблица 236 – продолжение с предыдущей страницы

x_guidelines	Вертикальные направляющие.
y_guidelines	Горизонтальные направляющие.

Methods:

clear_guidelines()	Удаляет все направляющие.
clear_selected_guidelines()	Очищает выбранные направляющие.
close()	Закрывает окно.
fill_on_pages()	Наиболее эффективно заполняет пространство отчета масштабированием его элементов.
get_printer()	Ссылка на используемый текущий принтер.
mouse_moved(x, y)	Сигнал при смещении курсора мыши.
redo()	Производит откат на один шаг вперед.
scale_with_center(scale, center)	Установка нового центра с заданным масштабированием.
set_printer(printer)	Устанавливает для отчета объект PySide2.QtPrintSupport.QPrinter
show([type])	Показывает окно в соответствие с приведенным типом.
undo()	Производит откат на один шаг назад.

property can_redo
Возможен ли откат на один шаг вперед.

Тип результата bool

property can_undo
Возможен ли откат на один шаг назад.

Тип результата bool

clear_guidelines()
Удаляет все направляющие.

clear_selected_guidelines()
Очищает выбранные направляющие.

close()
Закрывает окно.

fill_on_pages()
Наиболее эффективно заполняет пространство отчета масштабированием его элементов.

get_printer()
Ссылка на используемый текущий принтер. Для того, чтобы изменить настройки, нужно запросить существующий объект, поменять необходимые значения и снова назначить посредством ReportView.set_printer(). Или же установить другой объект PySide2.QtPrintSupport.QPrinter.

Список 226: Пример смены свойств у текущего окна отчета

```
from PySide2.QtGui import QPageLayout, QPageSize
import axipy
# Получение текущего окна отчета
reportview = axipy.view_manager.active
if isinstance(reportview, axipy.ReportView):
    # Получение текущего принтера
    printer = reportview.get_printer()
    # Именение размера страницы
    printer.setPageSize(QPageSize(QPageSize.A3))
    # Изменение ориентации страницы
    printer.setPageOrientation(QPageLayout.Landscape)
    # Установление нового значения
    reportview.set_printer(printer)
```

Тип результата QPrinter

property is_modified
Есть ли изменения в окне.

Тип результата bool

property mesh_size
Размер ячейки сетки.

Тип результата float

mouse_moved(x, y)
Сигнал при смещении курсора мыши. Возвращает значения в координатах отчета.

Параметры

- **x** (float) – X координата
- **y** (float) – Y координата

Пример:

```
reportview.mouse_moved.connect(lambda x,y: print('Coords: {} {}'.format(x,␣
↪y)))
```

Тип результата Signal

property position
Размер и положение окна.

Тип результата QRect

property rect
Размер и положение окна.

Предупреждение: Не рекомендуется, начиная с версии 4.0: Используйте position.

Тип результата QRect

redo()
Производит откат на один шаг вперед. При этом возвращается состояние до последней отмены.

property report
Объект отчета.

Тип результата Report

Результат Отчет.

scale_with_center(scale, center)
Установка нового центра с заданным масштабированием.

Параметры

- **scale** (float) – Коэффициент масштабирования по отношению к текущему.
- **center** (Pnt) – Устанавливаемый центр.

property scene_changed
«Сигнал об изменении контента окна.

Тип результата Signal

set_printer(printer)
Устанавливает для отчета объект PySide2.QtPrintSupport.QPrinter

Параметры printer (QPrinter) – Новое значение принтера или измененное запрошенное ранее через ReportView.get_printer()

show(type=1)
Показывает окно в соответствие с приведенным типом.

Таблица 238: Допустимые значения:

Константа	Значение	Описание
SHOW_NORMAL	1	Обычный показ окна (по умолчанию).
SHOW_MINIMIZED	2	Показ окна в режиме минимизации.
SHOW_MAXIMIZED	3	Показ окна в режиме распахивания.

property show_borders
Показывать границы страниц.

Тип результата float

property show_mesh
Показывать сетку привязки.

Тип результата bool

property show_ruler
Показывать линейку по краям.

Тип результата float

property show_type
Возвращает тип состояния окна. Подробнее см. show()

Тип результата int

property snap_mode
Включает режим привязки координат при редактировании геометрии в окне карты или отчета.

Тип результата bool

property snap_to_guidelines
Включение режима притяжения элементов отчета к направляющим.

Тип результата bool

property snap_to_mesh
Включение режима притяжения элементов отчета к узлам сетки.

Тип результата bool

property title
Заголовок окна просмотра.

Тип результата str

undo()
Производит откат на один шаг назад.

property view_scale
Текущий масштаб.

Тип результата float

property widget
Виджет, соответствующий содержимому окна.

Тип результата QWidget

Результат Qt5 виджет содержимого.

property x_guidelines
Вертикальные направляющие. Значения содержатся в единицах измерения отчета.

Рассмотрим на примере:

```
# Добавление вертикальной направляющей
reportview.x_guidelines.append(20)
# Изменение значения направляющей по индексу
reportview.x_guidelines[0] = 80
# Удаление всех направляющих.
reportview.clear_guidelines()
```

property y_guidelines
Горизонтальные направляющие. Работа с ними производится по аналогии с вертикальными направляющими.

16.1.8.11 ListLegend - Список легенд

class axipy.gui.**ListLegend**

Базовые классы: object

Список добавленных в окно легенд.

16.1.8.12 LegendView - Окно просмотра легенд карты

class axipy.gui.**LegendView**

Базовые классы: axipy.gui.View

Легенда для карты. Для создания экземпляра необходимо использовать axipy.gui.ViewManager.create_legendview() через экземпляр view_manager. В качестве параметра передается открытое ранее окно с картой:

```
legendView = view_manager.create_legendview(map_view)
```

Список легенд доступен через свойство legends:

```
for legend in legendView.legends:
   print(legend.caption)
```

Состав может меняться посредством вызова соответствующих методов свойства legends.

Добавление легенды для слоя карты:

```
legend = Legend(map_view.map.layers[0])
legend.caption = 'Легенда слоя'
legendView.legends.append(legend)
legendView.arrange()
```

Доступ к элементу по индексу. Поменяем описание четвертого оформления у первой легенды axipy.render.Legend окна:

```
legend = legendView.legends[1]
item = legend.items[3]
item.title = 'Описание'
legend.items[3] = item
```

Удаление первой легенды из окна:

```
legendView.legends.remove(0)
```

Methods:

arrange()	Упорядочивает легенды с целью устранения наложений легенд друг на друга.
close()	Закрывает окно.
show([type])	Показывает окно в соответствие с приведенным типом.

Attributes:

legends	Перечень добавленных в окно легенд.
position	Размер и положение окна.
rect	Размер и положение окна.
show_type	Возвращает тип состояния окна.
title	Заголовок окна просмотра.
widget	Виджет, соответствующий содержимому окна.

arrange()
Упорядочивает легенды с целью устранения наложений легенд друг на друга.

close()
Закрывает окно.

property legends
Перечень добавленных в окно легенд.

Тип результата ListLegend

property position
Размер и положение окна.

Тип результата QRect

property rect
Размер и положение окна.

> **Предупреждение:** Не рекомендуется, начиная с версии 4.0: Используйте position.

Тип результата QRect

show(type=1)
Показывает окно в соответствие с приведенным типом.

Таблица 241: Допустимые значения:

Константа	Значение	Описание
SHOW_NORMAL	1	Обычный показ окна (по умолчанию).
SHOW_MINIMIZED	2	Показ окна в режиме минимизации.
SHOW_MAXIMIZED	3	Показ окна в режиме распахивания.

property show_type
Возвращает тип состояния окна. Подробнее см. show()

Тип результата int

property title
Заголовок окна просмотра.

Тип результата str

property widget
Виджет, соответствующий содержимому окна.

Тип результата QWidget

Результат Qt5 виджет содержимого.

16.1.8.13 SelectionManager - Доступ к выделенным объектам

class axipy.gui.**SelectionManager**

Класс доступа к выделенным объектам.

Примечание: Получить экземпляр сервиса можно в атрибуте `axipy.gui.selection_manager`.

Methods:

`add`(table, ids)	Добавляет выборку записи таблицы по их идентификаторам.
`clear`()	Очищает выборку.
`get_as_cursor`()	Возвращает выборку в виде итератора по записям.
`get_as_table`()	Возвращает выборку в виде таблицы.
`remove`(table, ids)	Удаляет из выборки записи таблицы по их идентификаторам.
`set`(table, ids)	Создает выборку из записей таблицы по их идентификаторам.

Attributes:

`changed`	`Signal[]` Выделение было изменено.
`count`	Размер выделения, то есть количество выделенных записей (количество элементов в списке идентификаторов).
`ids`	Список идентификаторов выделенных записей.
`table`	Таблица, являющаяся источником текущего выделения.

add(table, ids)

Добавляет выборку записи таблицы по их идентификаторам.

Параметры

- **table** (`Table`) – Таблица.
- **ids** (`Union[List[int], int]`) – Идентификаторы записей.

property changed

`Signal[]` Выделение было изменено.

Тип результата `Signal`

clear()

Очищает выборку.

property count

Размер выделения, то есть количество выделенных записей (количество элементов в списке идентификаторов).

Тип результата int

get_as_cursor()
Возвращает выборку в виде итератора по записям.

Пример:

```
for f in selection_manager.get_as_cursor():
    print('Feature id={}. Страна={}'.format(f.id, f['Страна']))
```

Предупреждение: Не рекомендуется, начиная с версии 3.5: Используйте axipy.da.DataManager.selection.

Тип результата Iterator[Feature]

get_as_table()
Возвращает выборку в виде таблицы.

Тип результата Optional[Table]

Результат Таблица или None, если выборки нет.

Содержимое таблицы основывается на текущей выборке на момент вызова данного метода. При последующем изменении или сбросе выборки контент данной таблицы не меняется. Результирующей таблице присваивается наименование в формате data*, которое в последствии можно изменить. При закрытии базовой таблицы данная таблицы так-же закрывается.

Пример:

```
# Получаем таблицу из выборки.
tbl = selection_manager.get_as_table()
# Задаем желаемое имя таблицы (необязательно)
tbl.name = 'my_table'
# Регистрация в каталоге (необязательно)
app.mainwindow.catalog().add(tbl)
for f in tbl.items():
    print('Feature id={}. Страна={}'.format(f.id, f['Страна']))
```

Предупреждение: Не рекомендуется, начиная с версии 3.5: Используйте axipy.da.DataManager.selection.

property ids
Список идентификаторов выделенных записей.

Тип результата List[int]

remove(table, ids)
Удаляет из выборки записи таблицы по их идентификаторам.

Параметры

- **tbl** – Таблица.
- **ids** (Union[List[int], int]) – Идентификаторы записей.

set(table, ids)
Создает выборку из записей таблицы по их идентификаторам.

Параметры

- **table** (Table) – Таблица.
- **ids** (Union[List[int], int]) – Идентификаторы записей.

property table
Таблица, являющаяся источником текущего выделения.

Тип результата Optional[Table]

16.1.8.14 ViewManager - Менеджер содержимого окон

class axipy.gui.**ViewManager**
Менеджер содержимого окон.

Примечание: Используйте готовый экземпляр этого класса view_manager.

Список 227: Пример использования

```
table = provider_manager.openfile(filepath)
m = Map([table])
view_manager.create_mapview(m)
```

Methods:

activate(view)	Делает заданное окно активным.
close(view)	Закрывает окно.
close_all()	Закрывает все окна.
create_legendview(mapview)	Создает окно легенды для карты.
create_mapview(map)	Создает окно из для переданного объекта карты.
create_reportview([report])	Создает окно с планом отчета.
create_tableview(table)	Создает окно в виде табличного представления из объекта данных.

Attributes:

active	Текущее активное окно.
active_changed	Signal[] Активное окно изменилось.
count	Количество окон.
count_changed	Signal[] Количество окон изменилось.
global_parent	Может использоваться как „parent“ при использовании стандартных диалогов Qt.
legendviews	Список всех окон с легендами.
mapviews	Список всех окон с картами.
reportviews	Список всех окон с отчетами.

continues on next page

Таблица 245 – продолжение с предыдущей страницы

tableviews	Список всех окон с таблицами просмотра.
views	Список всех известных окон.

activate(view)
Делает заданное окно активным.

Параметры **view** (View) – Содержимое окна.

property active
Текущее активное окно.

Тип результата Optional[View]

Результат None, если нет активных окон.

property active_changed
Signal[] Активное окно изменилось.

Тип результата Signal

close(view)
Закрывает окно.

Параметры **view** (View) – Содержимое окна.

close_all()
Закрывает все окна.

property count
Количество окон.

Тип результата int

property count_changed
Signal[] Количество окон изменилось.

Тип результата Signal

create_legendview(mapview)
Создает окно легенды для карты.

Параметры **mapview** (MapView) – Окно с картой.

Тип результата LegendView

Результат Окно с легендой.

create_mapview(map)
Создает окно из для переданного объекта карты.

Параметры **map** (Map) – Карта.

Примечание: Переданная карта копируется.

Тип результата MapView

Результат Окно карты.

create_reportview(report=None)
Создает окно с планом отчета.

Параметры **report** (Optional[Report]) – План отчета. Если не передан, то создается по умолчанию.

Тип результата ReportView

Результат Окно отчета.

create_tableview(table)
Создает окно в виде табличного представления из объекта данных.

Параметры **table** (Table) – Таблица.

Тип результата TableView

Результат Окно таблицы.

property global_parent
Может использоваться как „parent“ при использовании стандартных диалогов Qt. Использование данного свойства решает проблему, когда окно показывается в панели задач как отдельное приложение.

Пример:

```
if QMessageBox.question(view_manager.global_parent, 'Вопрос', 'Отменить␣
↪действие?') == QMessageBox.Yes:
    pass
```

Тип результата QWidget

property legendviews
Список всех окон с легендами.

Тип результата List[LegendView]

property mapviews
Список всех окон с картами.

Пример:

```
for v in view_manager.mapviews:
    print('Widget:', v.title)

>>> Widget: Карта: world
>>> Widget: Карта: rus_obl
```

Тип результата List[MapView]

property reportviews
Список всех окон с отчетами.

Тип результата List[ReportView]

property tableviews
Список всех окон с таблицами просмотра.

Тип результата List[TableView]

property views
Список всех известных окон.

Тип результата List[View]

16.1.8.15 ChooseCoordSystemDialog - Диалог выбора СК

class axipy.gui.**ChooseCoordSystemDialog**(coordsystem=None)

Диалог выбора координатной системы.

Параметры **coordsystem** (Optional[CoordSystem]) – Система координат по умолчанию. Если не указана, то задается как значение по умолчанию axipy.cs.CoordSystem.current()

Пример:

```
csMercator = CoordSystem.from_prj('10, 104, 7, 0')
dialog = ChooseCoordSystemDialog(csMercator)
if dialog.exec() == QDialog.Accepted:
    result_cs = dialog.chosenCoordSystem()
    print("Выбрано:", result_cs.description)
```

Methods:

chosenCoordSystem()	Возвращает выбранную в диалоге систему координат.

chosenCoordSystem()

Возвращает выбранную в диалоге систему координат.

Тип результата CoordSystem

16.1.8.16 PasswordDialog - Диалог аутентификации пользователя.

class axipy.gui.**PasswordDialog**

Диалог задания или корректировки данных аутентификации пользователя.

Пример:

```
dialog = PasswordDialog()
dialog.user_name = 'user'
dialog.password = 'pass'
if dialog.exec() == QDialog.Accepted:
    print(dialog.user_name, dialog.password)
```

Attributes:

password	Пароль.
user_name	Имя пользователя.

property password

Пароль.

Тип результата str

property user_name

Имя пользователя.

Тип результата str

16.1.8.17 StyledButton - Кнопка выбора стиля

class axipy.gui.**StyledButton**(style, parent=None)
Кнопка, отображающая стиль и позволяющая его менять

Примечание: Сигнал styleChanged испускается при смене стиля.

Параметры **style** (Style) – Стиль по умолчанию.

Пример добавления кнопки на диалог:

```
style = Style.from_mapinfo("Pen (1, 2, 8421504)  Brush (2, 255, 0)")

class Dialog(QDialog):
    def __init__(self, parent = None):
        QDialog.__init__(self, parent)
        self.pb = StyledButton( style, self)
        self.pb.styleChanged.connect(self.styleResult)
        self.pb.setGeometry(100, 100, 100, 50)

    def styleResult(self):
        print('Стиль изменен', self.pb.style())

dialog = Dialog()
dialog.exec()
```

Methods:

style()	Результирующий стиль.

style()
Результирующий стиль.

Тип результата Style

16.1.8.18 Workspace - Рабочее пространство

class axipy.gui.**Workspace**
Рабочее пространство для сохранения текущего состояния.

Примечание: Данный класс следует использовать в случае, когда отсутствует главное окно приложения axipy.app.MainWindow для сохранения или чтения текущего состояния. Если же главное окно присутствует, то можно воспользоваться методами axipy.app.MainWindow.load_workspace() для чтения и axipy.app.MainWindow.save_workspace() для записи рабочего пространства.

Рабочее пространство можно как сохранять в файл так и читать из него. При чтении из файла рабочего пространства посредством метода load_file() все источники данных (таблицы) открываются и добавляются в каталог axipy.da.DataManager, доступный через переменную axipy.da.data_manager. А окна с наполнением добавляются в менеджер окон axipy.gui.ViewManager, доступный через переменную view_manager.

В случае записи текущего состояния в файл рабочего пространства последовательность обратная рассмотренной. Состояние каталога и менеджера окон записывается в рабочее пространство посредством метода `save_file()`.

Рассмотрим пример чтения данных в рамках отдельного приложения:

```
import axipy

# инициализация ядра
app = axipy.init_axioma()
print('Before: tables({}), views({})'.format(len(data_manager), len(view_
↪manager)))
# Чтение рабочего набора
ws.load_file('ws.mws')
print('After: tables({}), views({})'.format(len(data_manager), len(view_manager)))
# Если есть хотя бы одно окно с картой, показываем его
if len(axipy.view_manager.mapviews):
    mapview = axipy.view_manager.mapviews[0]
    mapview.show()
app.exec_()

>>> Before: tables(0), views(0)
>>> After: tables(5), views(3)
```

Пример записи рабочего пространства:

```
ws = Workspace()
ws.save_file('ws_out.mws')
```

load_file(fileName)
Читает из файла рабочего пространства и заносит данные в текущее окружение.

Параметры fileName (str) – Наименование входного файла.

save_file(fileName)
Сохраняет текущее состояние в файл рабочего пространства.

Параметры fileName (str) – Наименование выходного файла.

16.1.9 axipy.menubar

Модуль меню главного окна ГИС Аксиома.

16.1.9.1 Button - Кнопка

class axipy.menubar.**Button**
Кнопка с инструментом для добавления в меню. Абстрактный класс.

Для создания объекта этого класса используйте `axipy.menubar.ActionButton`, `axipy.menubar.ToolButton`.

Attributes:

action	Ссылка на объект QAction.

continues on next page

Таблица 249 – продолжение с предыдущей страницы

observer_id	Идентификатор наблюдателя для определения доступности инструмента.

Methods:

remove()	Удаляет кнопку из меню.

property action
Ссылка на объект QAction. Через него можно производить дополнительные необходимые действия через объект Qt.

Пример задания всплывающей подсказки, используя метод класса QAction:

```
button.action.setToolTip('Всплывающая подсказка')
```

Тип результата QAction

property observer_id
Идентификатор наблюдателя для определения доступности инструмента.

Тип результата str

remove()
Удаляет кнопку из меню.

16.1.9.2 ActionButton - Кнопка с действием

class axipy.menubar.**ActionButton**(title, on_click, icon='', enable_on=None, tooltip=None)

Базовые классы: axipy.menubar.Button

Кнопка с действием.

Параметры

- **title** (str) – Текст.
- **on_click** (Callable[[], Any]) – Действие на нажатие. Делегируется функция, которая будет вызвана при активации инструмента.
- **icon** (Union[str, QIcon]) – Иконка. Может быть путем к файлу или адресом ресурса.
- **enable_on** (Union[str, DefaultKeys, None]) – Идентификатор наблюдателя для определения доступности кнопки. Если это пользовательский наблюдатель, то указывается его наименование при создании.
- **tooltip** (Optional[str]) – Строка с дополнительной короткой информацией по данному действию.

См.также:

axipy.da.StateManager.

Список 228: Пример со встроенным наблюдателем.

```
button = menubar.ActionButton('Мое действие', on_click=lambda: print('clicked'),
                      enable_on=state_manager.HasTables)
```

Список 229: Пример со пользовательским наблюдателем.

```
my_observer = state_manager.create('MyStateManager', False)
button = menubar.ActionButton('Мое действие', on_click=lambda: print('clicked'),
                      enable_on='MyStateManager')
```

Attributes:

action	Ссылка на объект QAction.
observer_id	Идентификатор наблюдателя для определения доступности инструмента.

Methods:

remove()	Удаляет кнопку из меню.

property action
Ссылка на объект QAction. Через него можно производить дополнительные необходимые действия через объект Qt.

Пример задания всплывающей подсказки, используя метод класса QAction:

```
button.action.setToolTip('Всплывающая подсказка')
```

Тип результата QAction

property observer_id
Идентификатор наблюдателя для определения доступности инструмента.

Тип результата str

remove()
Удаляет кнопку из меню.

16.1.9.3 ToolButton - Кнопка с инструментом

class axipy.menubar.**ToolButton**(title, on_click, icon='', enable_on=None, tooltip=None)

Базовые классы: axipy.menubar.Button

Кнопка с инструментом.

Параметры

- **title** (str) – Текст.

- **on_click** (Callable[[], MapTool]) – Класс инструмента, наследник от :class:“axipy.gui.MapTool“.
- **icon** (Union[str, QIcon]) – Иконка. Может быть путем к файлу или адресом ресурса.
- **enable_on** (Union[str, DefaultKeys, None]) – Идентификатор наблюдателя для определения доступности кнопки.

См.также:

axipy.da.StateManager.

Список 230: Пример

```
# Класс инструмента
class MyTool(MapTool):
    pass
param = 'Передаваемый параметр'
# Передача имени класса MapTool как параметр
button = ToolButton('Мой инструмент', MyTool)
# Если необходимо передавать параметры в конструктор, то можно передать как␣
↪конструктор
# внутри lambda функции
button = ToolButton('Мой инструмент', lambda: MyTool(param))
```

Attributes:

action	Ссылка на объект QAction.
observer_id	Идентификатор наблюдателя для определения доступности инструмента.

Methods:

remove()	Удаляет кнопку из меню.

property action

Ссылка на объект QAction. Через него можно производить дополнительные необходимые действия через объект Qt.

Пример задания всплывающей подсказки, используя метод класса QAction:

```
button.action.setToolTip('Всплывающая подсказка')
```

Тип результата QAction

property observer_id

Идентификатор наблюдателя для определения доступности инструмента.

Тип результата str

remove()

Удаляет кнопку из меню.

16.1.9.4 Separator - Разделитель

class axipy.menubar.**Separator**

Базовые классы: axipy.menubar.Button

Разделитель.

Attributes:

action	Ссылка на объект QAction.
observer_id	Идентификатор наблюдателя для определения доступности инструмента.

Methods:

remove()	Удаляет кнопку из меню.

property action
Ссылка на объект QAction. Через него можно производить дополнительные необходимые действия через объект Qt.

Пример задания всплывающей подсказки, используя метод класса QAction:

```
button.action.setToolTip('Всплывающая подсказка')
```

Тип результата QAction

property observer_id
Идентификатор наблюдателя для определения доступности инструмента.

Тип результата str

remove()
Удаляет кнопку из меню.

16.1.9.5 Position - Положение кнопки

class axipy.menubar.**Position**(tab, group)
Положение кнопки в меню.

Параметры

- **tab** (str) – Название вкладки.
- **group** (str) – Название группы.

Пример:

```
pos = Position("Основные", "Команды")
```

Methods:

add(button[, size])	Добавляет кнопку текущее положение.

add(button, size=2)
Добавляет кнопку текущее положение.

Параметры

- **button** (Union[QAction, Button]) – Кнопка.
- **size** (int) – Размер кнопки. Маленькая кнопка - 1. Большая кнопка - 2.

Примечание: Для выполнения примеров достаточно произвести импорт всего окружения:

```
from axipy import *
```

Содержание модулей Python

a

Алфавитный указатель

M

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z