Аксиома.ГИС API для Python

ООО ЭСТИ

дек. 08, 2020

Оглавление

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

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

Глава 1

Модули Аксиома.ГИС

1.1 axipy

Основной пакет API для взаимодействия с Аксиомой.ГИС.

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

Функции

`axipy.init_axioma()`

Инициализирует ядро Аксиомы.ГИС.

Тип результата `QApplication`

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

Example:

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

`axipy.local_file(`*caller_file*`)`

Создает функцию получения пути к файлу/папку относительно другого файла.

Возвращаемая функция принимает относительный путь и возвращает абсолютный путь.

Параметры `caller_file` (`str`) – Файл, относительно которого будет строиться относительный путь. Чаще всего специальный объект `__file__`.

Тип результата `Callable`[..., `str`]

Результат Функция получения пути.

Example:

```
locate = local_file(__file__)
icon = locate('images', '32px', 'logo.png')
```

axipy.translate(*context*)

Создает функцию перевода строки.

Возвращаемая функция производит поиск строки в загруженных файлах перевода.

Параметры `context` (`str`) – Идентификатор группы для перевода. Чаще всего имя модуля для Аксиомы `__package__`.

Тип результата `Callable`[[`str`], `str`]

Результат Функция получения перевода строки.

Example:

```
tr = translate('ru_mywebsite_myplugin')
tr('Строка для перевода')
```

1.2 axipy.app

Модуль приложения.

Данный модуль является основным модулем приложения.

axipy.app.mainwindow

Готовый экземпляр главного окна Аксиомы.

Type *MainWindow*

1.2.1 Главное окно приложения - MainWindow

class axipy.app.MainWindow

Базовые классы: `PySide2.QtCore.QObject`

Главное окно Аксиомы.

Примечание: Используйте готовый объект *axipy.app.mainwindow*.

---

catalog()

Возвращает хранилище объектов приложения.

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

Тип результата *DataCatalog*

qt_object()

Возвращает Qt5 объект окна.

Тип результата `QMainWindow`

1.3 axipy.cs

Модуль систем координат.

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

1.3.1 Система Координат (СК) - CoordSystem

class axipy.cs.CoordSystem(*_shadow*)

Базовые классы: object

Система координат (СК).

property description
Текстовое описание.

Тип результата str

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

Тип результата Optional[int]

from_degree(*v*)
Переводит из градусов в единицы измерения системы координат.

Тип результата Union[QPointF, List[QPointF], QRectF]

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

Параметры wkt – Стандартное значение EPSG.

Тип результата *CoordSystem*

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

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

Примеры:

```
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.

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.

Пример:

```
crs1 = CoordSystem.from_string('epsg:4326')
crs2 = CoordSystem.from_string('prj:1,104')
```

Тип результата *CoordSystem*

`classmethod from_units(unit)`
Создает декартову систему координат.

Параметры `unit` (*LinearUnit*) – Единицы измерения системы координат.

Пример:

```
ne = CoordSystem.from_units(LinearUnit.kilometer())
```

Тип результата *CoordSystem*

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

Параметры `wkt` (str) – Строка WKT.

Тип результата *CoordSystem*

`property name`
Описание координатной системы.

Тип результата str

`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 QRectF

`to_degree(v)`
Переводит из единиц измерения системы координат в градусы.

Тип результата Union[QPointF, List[QPointF], QRectF]

`property unit`
Единицы измерения.

Тип результата *LinearUnit*

`property wgs84_params`
Коэффициенты преобразования датума к WGS84.

Тип результата List[float]

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

Тип результата str

1.3.2 Единицы измерения

1.3.2.1 Единицы измерения расстояний - LinearUnit

class axipy.cs.LinearUnit(*_shadow*)

Базовые классы: axipy.cs.EarthUnit

Линейные единицы измерения. Используются для работы с координатами объектов или расстояний.

static centimeter()
Сантиметры.

Тип результата *LinearUnit*

static kilometer()
Километры.

Тип результата *LinearUnit*

static meter()
Метры.

Тип результата *LinearUnit*

static mile()
Мили.

Тип результата *LinearUnit*

static millimeter()
Миллиметры.

Тип результата *LinearUnit*

static nautical_mile()
Морские мили.

Тип результата *LinearUnit*

1.3.2.2 Единицы измерения площадей - AreaUnit

class axipy.cs.AreaUnit(*_shadow*)

Базовые классы: axipy.cs.EarthUnit

Единицы измерения площадей.

static sq_centimeter()
Квадратные сантиметры.

Тип результата *AreaUnit*

static sq_kilometer()
Квадратные километры.

Тип результата *AreaUnit*

static sq_meter()
Квадратные метры.

Тип результата *AreaUnit*

static sq_mile()
Квадратные мили.

Тип результата *AreaUnit*

static sq_millimeter()
Квадратные миллиметры.

Тип результата *AreaUnit*

static sq_nautical_mile()
Квадратные морские мили.

Тип результата *AreaUnit*

1.3.3 Трансформация координат - CoordTransformer

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

Базовые классы: object

Класс для преобразования координат из одной СК в другую.

__init__(*cs_from*, *cs_to*)
Конструктор.

Параметры

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

transform(*point*)
Преобразовывает точки из исходной СК в целевую СК.

Параметры point (Union[QPointF, List[QPointF]]) – Входное значение. Может быть точкой или массивом точек.

Тип результата Union[QPointF, List[QPointF]]

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

Исключение RuntimeError – Ошибка выполнения преобразования.

Пример преобразования точки:

```
from axipy import *
from PySide2.QtCore import QPointF

csLL = CoordSystem.from_prj("1, 104")
csMercator = CoordSystem.from_prj("10, 104, 7, 0")
point = QPointF(10, 10)
transformer = CoordTransformer(csLL, csMercator)
outPoint = transformer.transform(point)
print('Result:', outPoint)
```

1.4 axipy.da

Модуль источников данных.

В данном модуле содержатся классы и методы для работы с источниками данных.

1.4.1 Каталог данных - DataCatalog

class axipy.da.DataCatalog(*shadow=None*)

Базовые классы: PySide2.QtCore.QObject

Хранилище источников данных.

add(*data_object*)
Добавляет объект данных в хранилище.

Параметры data_object (*DataObject*) – Объект данных для добавления.

count()
Возвращает количество объектов данных.

Тип результата int

find(*name*)
Производит поиск объект данных по имени.

Параметры name (str) – Имя объекта данных.

Тип результата Optional[*DataObject*]

Результат Искомый объект данных или None.

objects()
Возвращает список объектов.

Тип результата List[*DataObject*]

remove(*data_object*)
Удаляет объект данных.

Параметры data_object (*DataObject*) – Объект данных для удаления.

tables()
Возвращает список таблиц.

Тип результата List[*Table*]

1.4.2 Источник данных - DataObject

class axipy.da.DataObject(*shadow*, *object_id*)

Базовые классы: object

Абстрактный объект данных.

Открываемые объекты из источников данных представляются объектами этого типа. Возможные реализации: таблица, растр, грид, чертеж, панорама, и так далее.

property name
Название объекта данных.

Тип результата str

property provider
Провайдер изначального источника данных.

Тип результата str

1.4.2.1 Таблица - Table

class axipy.da.Table(*shadow*, *object_id*, *is_editable=False*)

Базовые классы: *axipy.da.DataObject*

Абстрактный класс таблицы.

commit()
Сохраняет изменения в таблице.

Из временного файла транзакций все изменения переносятся в основную таблицу.

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[QRectF, tuple, None]) – Ограничивающий прямоугольник.

Тип результата int

Результат Количество записей.

insert(*features*)
Вставляет записи в таблицу.

Параметры features (Union[Iterator[*Feature*], *Feature*]) – Записи для вставки.

Тип результата None

property is_editable
Признак того, что таблица является редактируемой.

Тип результата bool

property is_spatial
Признак того, что таблица является пространственной.

Тип результата bool

items(*bbox=None*, *ids=None*)
Запрашивает записи, удовлетворяющие параметрам.

Параметры

- bbox (Union[QRectF, tuple, None]) – Ограничивающий прямоугольник.
- ids (Optional[List[int]]) – Список идентификаторов.

Результат Итератор по записям.

remove(*ids*)
Удаляет записи из таблицы.

Параметры ids (Union[int, Iterator[int]]) – Идентификаторы записей для удаления.

schema()
Возвращает схему таблицы.

Тип результата *Schema*

Схема таблицы - Schema

class axipy.da.Schema(**attributes*, *coordsystem=None*, *_dictionary=None*)

Базовые классы: dict

Схема таблицы. Представляет собой перечень атрибутов.

__init__(**attributes*, *coordsystem=None*, *_dictionary=None*)
Конструктор.

Параметры

- *attributes – Список атрибутов.
- coordsystem (Optional[*CoordSystem*]) – Система координат.

property coordsystem
Система координат.

Тип результата Optional[str]

Запись в таблице - Feature

class axipy.da.Feature(*properties={}*, *geometry=None*, *style=None*, *shadow=None*, ***kwargs*)

Базовые классы: object

Запись в таблице.

__contains__(*key*)
Проверяет наличие атрибута у записи.

Параметры key (Union[int, str]) – Имя или номер атрибута.

__getitem__(*index*)
Возвращает значение атрибута по индексу или ключу.

Параметры index (Union[int, str]) – Индекс атрибута или его имя.

Тип результата Any

Результат Значение атрибута.

__init__(*properties={}*, *geometry=None*, *style=None*, *shadow=None*, ***kwargs*)
Конструктор.

Параметры

- properties (dict) – Значения атрибутов.
- geometry (Optional[*Geometry*]) – Геометрия.
- style (Optional[*Style*]) – Стиль.

__len__()
Возвращает количество атрибутов в записи.

Тип результата int

__setitem__(*index*, *value*)
Присваивает атрибуту значение по индексу или ключу.

Параметры

- index (Union[int, str]) – индекс атрибута или его имя.
- value (Any) – Присваиваемое значение.

property id
Идентификатор записи в таблице.

Тип результата int

to_geojson()
Представляет запись в виде строки в формате json.

Тип результата str

1.4.2.2 Атрибут записи - Attribute

class axipy.da.Attribute(*iterable=()*, */*)

Базовые классы: list

Атрибут схемы таблицы.

Используется для создания и инспектирования атрибутов и схем.

Примечание: Используйте более короткий псевдоним этого класса axipy.da.attr

---

Пример создания схемы:

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

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 schema(**attributes*, *coordsystem=None*)
Возвращает схему, на основе которой сформирован атрибут.

Параметры coordsystem (Optional[*CoordSystem*]) – Система координат.

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*

Геометрия - Geometry

class axipy.da.Geometry(*shadow*)

Базовые классы: object

Геометрический объект (геометрия).

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 area
Рассчитывает площадь, если объект площадной. В противном случае возвращает 0.

Тип результата float

boundary()
Возвращает границы геометрии в виде полилинии.

Тип результата *Geometry*

property bounds
Возвращает минимальный ограничивающий прямоугольник.

Тип результата QRectF

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*

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
Возвращает Систему Координат (СК) геометрии.

Тип результата *CoordSystem*

covers(*other*)
Возвращает True, если геометрия охватывает геометрию *other*.

Тип результата bool

crosses(*other*)
Возвращает True, если у геометрий некоторые, но не все внутренние точки являются общими.

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

distance(*other*)
Производит расчет расстояния до объекта *other*. Результат возвращает в СК текущего объекта.

envelope()
Возвращает полигон, описывающий заданную геометрию.

Тип результата *Geometry*

equals(*other*)
Производит сравнение с другой геометрией.

Тип результата bool

Результат Возвращает True, если геометрии равны.

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. В данном случае система координат

- создаваемой геометрии будет установлено исходя их этого значения. (*для*) –
- coordsystem (Optional[*CoordSystem*]) – Система координат, которая будет установлена для геометрии. Если указано значение SRID, игнорируется.

Пример:

```
from axipy import *
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)
```

intersection(*other*)
Возвращает область пересечения с другой геометрией.

Тип результата *Geometry*

intersects(*other*)
Возвращает True, если геометрии пересекаются.

Тип результата bool

property length
Рассчитывает периметр геометрии.

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

reproject(*cs*)
Перепроецирует геометрию в другую систему координат.

Параметры cs (*CoordSystem*) – СК, в которой требуется получить объект.

Тип результата *Geometry*

symmetric_difference(*other*)
Возвращает логический XOR областей геометрий (объединение разниц).

Параметры other (*Geometry*) – Геометрия для анализа.

Тип результата *Geometry*

touches(*other*)
Возвращает True, если геометрии соприкасаются.

Тип результата bool

property type
Возвращает тип геометрического элемента.

Таблица 1: Возможные значения

Значение	Наименование
Unknown	Не определен
Point	Точка
LineString	Полилиния
Polygon	Полигон
MultiPoint	Коллекция точек
MultiLineString	Коллекция полилиний
MultiPolygon	Коллекция полигонов
MultiGeometry	Смешанная коллекция
Arc	Дуга
Ellipse	Эллипс
Rectange	Прямоугольник
RoundedRectange	Скругленный прямоугольник
Text	Текст

Пример:

```
point = Geometry.from_wkt('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 wkt`
Возвращает WKT строку для геометрии.

Тип результата str

Стиль - Style

`class axipy.da.Style`(*shadow*)

Базовые классы: object

Стиль геометрического объекта. Определяет как будет отрисован геометрический объект.

`classmethod from_mapinfo`(*mapbasic_string*)
Получает стиль из строки формата MapBasic.

Параметры mapbasic_string (str) – Строка в формате MapBasic.

Пример:

```
style = Style.from_mapinfo("Pen (1, 2, 0) Brush (8, 255)")
```

Тип результата *Style*

to_mapinfo()
Возвращает строковое представления в формате MapBasic.

Пример:

```
>>> style.to_mapinfo()
Pen (1, 2, 0) Brush (8, 255)
```

Тип результата str

to_ogr()
Возвращает строкового представления в формате OGR.

Тип результата str

1.5 axipy.gui

Модуль пользовательского интерфейса.

В данном модуле содержатся классы связанные с пользовательским интерфейсом.

axipy.gui.viewservice
Экземпляр менеджера содержимого окон.

Type *ViewService*

axipy.gui.selection
Экземпляр доступа к выделенным объектам.

Type *SelectionService*

1.5.1 Инструмент окна карты - MapTool

class axipy.gui.MapTool

Базовые классы: axipy.cpp_gui.MapTool

Инструмент окна карты.

При создании своего инструмента новый инструмент наследуется от этого класса, и переопределяет необходимые обработчики событий.

Example:

```
MyTool(MapTool):

    def mousePressEvent(self, event):
        print('mouse pressed')
        return self.PassEvent
```

keyPressEvent(*event*)
Обрабатывает событие нажатия клавиши клавиатуры.

Параметры event (QKeyEvent) – Событие нажатия клавиши клавиатуры.

Тип результата bool

Результат PassEvent, чтобы пропустить событие дальше по цепочке обработчиков. Или BlockEvent, чтобы блокировать дальнейшую обработку события.

keyReleaseEvent(*event*)
Обрабатывает событие отпускания клавиши клавиатуры.

Параметры event (QKeyEvent) – Событие отпускания клавиши клавиатуры.

Тип результата bool

Результат PassEvent, чтобы пропустить событие дальше по цепочке обработчиков. Или BlockEvent, чтобы блокировать дальнейшую обработку события.

mouseDoubleClickEvent(*event*)
Обрабатывает событие двойного клика мыши.

Параметры event (QMouseEvent) – Событие двойного клика мыши.

Тип результата bool

Результат PassEvent, чтобы пропустить событие дальше по цепочке обработчиков. Или BlockEvent, чтобы блокировать дальнейшую обработку события.

mouseMoveEvent(*event*)
Обрабатывает событие перемещения мыши.

Параметры event (QMouseEvent) – Событие перемещения мыши.

Тип результата bool

Результат PassEvent, чтобы пропустить событие дальше по цепочке обработчиков. Или BlockEvent, чтобы блокировать дальнейшую обработку события.

mousePressEvent(*event*)
Обрабатывает событие нажатия клавиши мыши.

Параметры event (QMouseEvent) – Событие нажатия клавиши мыши.

Тип результата bool

Результат PassEvent, чтобы пропустить событие дальше по цепочке обработчиков. Или BlockEvent, чтобы блокировать дальнейшую обработку события.

mouseReleaseEvent(*event*)
Обрабатывает событие отпускания клавиши мыши.

Параметры event (QMouseEvent) – Событие отпускания клавиши мыши.

Тип результата bool

Результат PassEvent, чтобы пропустить событие дальше по цепочке обработчиков. 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

Тип результата None

to_scene(*device*)
Переводит точки из координат окна(пикселей) в координаты на карте.

Параметры device (Union[QRectF, QPointF, QLineF, QPolygonF]) – Точки в координатах окна.

Тип результата Union[QRectF, QPointF, QLineF, QPolygonF]

Результат Точки в координатах карты.

property view
Отображение данных в окне.

Тип результата *View*

wheelEvent(*event*)
Обрабатывает событие колеса мыши.

Параметры event (QWheelEvent) – Событие колеса мыши.

Тип результата bool

Результат PassEvent, чтобы пропустить событие дальше по цепочке обработчиков. Или BlockEvent, чтобы блокировать дальнейшую обработку события.

1.5.2 Базовый класс для отображения данных в окне - View

class axipy.gui.View(*shadow*, *object_id*)

Базовые классы: PySide2.QtCore.QObject

Базовый класс для отображения данных в окне.

widget()
Возвращает виджет, соответствующий содержимому окна.

Тип результата QWidget

Результат Qt5 виджет содержимого.

1.5.3 Таблица просмотра атрибутивной информации - TableView

class axipy.gui.TableView(*shadow*, *object_id*)

Базовые классы: *axipy.gui.View*

Таблица просмотра атрибутивной информации.

data_object()
Возвращает таблицу, на основании которой создается данное окно просмотра.

Тип результата *DataObject*

Результат Таблица.

1.5.4 Окно просмотра карты - MapView

class axipy.gui.MapView(*shadow*, *object_id*)

Базовые классы: *axipy.gui.View*

Окно просмотра карты.

property coordsystem
Система координат карты.

Тип результата *CoordSystem*

device_rect()
Возвращает видимую область в координатах окна(пиксели).

Тип результата QRectF

Результат Прямоугольник в координатах окна.

device_to_scene_transform()
Возвращает объект трансформации из координат окна в координаты карты.

Тип результата QTransform

Результат Объект трансформации.

editable_layer()
Возвращает редактируемый слой.

Тип результата Optional[*Layer*]

Результат Редактируемый слой.

map()
Возвращает объект карты.

Тип результата *Map*

Результат Карта.

scene_rect()
Возвращает видимую область в координатах карты.

Тип результата QRectF

Результат Прямоугольник в координатах карты.

scene_to_device_transform()
Возвращает объект трансформации из координат карты в координаты окна.

Тип результата QTransform

Результат Объект трансформации.

1.5.5 Доступ к выделенным объектам - SelectionService

class axipy.gui.SelectionService

Базовые классы: PySide2.QtCore.QObject

Класс доступа к выделенным объектам.

Примечание: Получить экземпляр сервиса можно в атрибуте *axipy.gui.selection*.

---

add(*table*, *ids*)
Добавляет выборку записи таблицы по их идентификаторам.

Параметры

- table (*Table*) – Таблица.
- ids – Идентификаторы записей.

property changed
[signal] Выделение было изменено.

Type Callable[]

clear()
Очищает выборку.

count()
Возвращает размер выделения, то есть количество выделенных записей (количество элементов в списке идентификаторов).

ids()
Возвращает список идентификаторов выделенных записей.

remove(*table*, *ids*)
Удаляет из выборки записи таблицы по их идентификаторам.

Параметры

- table (*Table*) – Таблица.
- ids (Union[List[int], int]) – Идентификаторы записей.

set(*table*, *ids*)
Создать выборку из записей таблицы по их идентификаторам.

Параметры

- table (*Table*) – Таблица.
- ids (Union[List[int], int]) – Идентификаторы записей.

table()
Возвращает таблицу, являющуюся источником текущего выделения.

Тип результата *Table*

1.5.6 Менеджер содержимого окон - ViewService

class axipy.gui.ViewService

Базовые классы: PySide2.QtCore.QObject

Менеджер содержимого окон.

Примечание: Используйте готовый экземпляр этого класса *viewservice*.

__contains__(*key*)
Проверяет существование окна с заданным именем.

Параметры key (str) – Имя окна.

Тип результата bool

__getitem__(*key*)
Возвращает существующее окно по имени.

Исключение KeyError – Окно с заданным именем не найдено.

Тип результата *View*

activate(*view*)
Делает заданное окно активным.

Параметры view (*View*) – Содержимое окна.

active()
Возвращает текущее активное окно.

Тип результата Optional[*View*]

property activeChanged
[signal] Активное окно изменилось.

Type Callable[]

close(*view*)
Закрывает окно.

Параметры view (*View*) – Содержимое окна.

count()
Количество окон.

Тип результата int

property countChanged
[signal] Количество окно изменилось.

Type Callable[]

create_view(*obj*)
Создает окно из объекта данных.

Параметры obj (Union[*Map*, *Table*]) – Карта или таблица.

Тип результата Union[*MapView*, *TableView*]

Результат Окно карты или окно таблицы.

views()
Список всех известных окон.

Тип результата List[*View*]

1.6 axipy.io

Модуль открытия/создания источников данных.

Пример:

```
from axipy import io
table = io.openfile('../path/to/datadir/table.tab')
```

axipy.io.create(*definition*)

Создает 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.

Параметры definition (dict) – Описание объекта данных.

Пример:

```
definition = {
    'src': '../path/to/datadir/edit/table.tab',
    'schema': attr.schema(
        attr.string('field1'),
        attr.int('field2'),
    ),
}
table = io.create(definition)
```

Тип результата *DataObject*

axipy.io.createfile(*filepath*, *schema*)

Создает таблицу.

create() выполняет ту же функцию, но в более обобщенном виде.

Параметры

- filepath (str) – Путь к создаваемой таблице.
- schema (*Schema*) – Схема таблицы.

axipy.io.loaded_providers()

Возвращает список всех загруженных провайдеров данных.

Тип результата dict

Результат Провайдеры в виде пар (Идентификатор : Описание).

```
print(io.loaded_providers())
```

axipy.io.open(*definition*)
Открывает данные по описанию.

Формат описания объектов данных индивидуален для каждого провайдера данных, однако многие элементы используются для всех провайдеров данных.

Параметры `definition` (`dict`) – Описание объекта данных.

Пример открытия GPKG файла:

```
definition = { 'src': '../path/to/datadir/example.gpkg',
               'dataobject': 'tablename',
               'provider': 'SqliteDataProvider' }
table = io.open(definition)
```

Пример открытия таблицы базы данных:

```
definition = {"src": "localhost",
              "db": "sample",
              "user": "postgres",
              "password": "postgres",
              "dataobject": "public.world",
              "provider": "PgDataProvider"}
table = io.open(definition)
```

Тип результата *DataObject*

axipy.io.openfile(*filepath*, *dataobject=''*, *provider=''*)
Открывает данные из файла.

Параметры

- `filepath` (`str`) – Путь к открываемому файлу.
- `dataobject` (`str`) – Имя открываемого объекта данных.
- `provider` (`str`) – Провайдер, с помощью которого будет произведено открытие. Если не указан, выбирается наиболее подходящий. Перечень можно получить с помощью *loaded_providers()*

Пример:

```
table = io.openfile('../path/to/datadir/example.gpkg', dataobject='tablename', provider=
↪'SqliteDataProvider')
```

Тип результата *DataObject*

axipy.io.query(*query_text*, **tables*)
Выполняет SQL-запрос к перечисленным таблицам.

Параметры

- `query_text` (`str`) – Текст запроса.
- `*tables` – Список таблиц, к которым выполняется запрос.

Тип результата `Optional`[*Table*]

Результат Таблица, если результатом запроса является таблица.

Пример:

```
query_text = "SELECT * FROM world, caps WHERE world.capital = caps.capital"
joined = io.query(query_text, world, caps)
```

axipy.io.query_aggregate(*query_text*, **tables*)

Выполняет SQL-запрос и возвращает значения первой записи.

Удобен при чтении результата агрегирующих запросов.

Параметры

- query_text (str) – Текст запроса.
- *tables – Список таблиц, к которым выполняется запрос.

Тип результата tuple

Результат Кортеж значений.

Пример:

```
avg, max, sum = io.query_aggregate(f'SELECT AVG(myvalue), MAX(myvalue), SUM(myvalue) FROM
↪{table.name}', table)
```

axipy.io.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:

```
>>> io.read_contents('../path/to/datadir/example.gpkg')
['world', 'worldcap']

>>> world = io.openfile('../path/to/datadir/example.gpkg',
...         dataobject='world')
```

1.7 axipy.menubar

Модуль меню главного окна Аксиомы.ГИС.

class axipy.menubar.Position(*tab*, *group*)

Базовые классы: object

Положение кнопки в меню.

Параметры

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

add(*button*, *size=2*)

Добавляет кнопку текущее положение.

Параметры

- **button** (Union[QAction, *ToolDefinition*]) – Кнопка.
- **size** (int) – Размер кнопки. Маленькая кнопка - 1. Большая кнопка - 2.

class axipy.menubar.ToolDefinition(*action*, *factory*)

Базовые классы: object

Кнопка с инструментом для добавления в меню.

axipy.menubar.create_button(*title*, *on_click*, *icon=''*)

Создает кнопку с заданными параметрами.

Параметры

- **title** (str) – Текст.
- **on_click** (Optional[Callable]) – Действие на нажатие. Любой функтор без параметров.
- **icon** (Union[str, QIcon]) – Иконка. Может быть путем к файлу или адресом ресурса.

Тип результата QAction

Результат Кнопка.

axipy.menubar.create_tool(*title*, *on_click*, *icon=''*)

Создает инструмент с заданными параметрами.

Параметры

- **title** (str) – Текст.
- **on_click** (Callable[[], MapTool]) – Фабрика инструмента. Любой функтор без параметров, возвращающий новый инструмент. Чаще всего само объявление инструмента.
- **icon** (Union[str, QIcon]) – Иконка. Может быть путем к файлу или адресом ресурса.

Тип результата *ToolDefinition*

Результат Кнопка с инструментом.

Example:

```
tool = create_tool("Мой инструмент", on_click=MyTool)
```

axipy.menubar.get_position(*tab*, *group*)

Возвращает положение в меню. Может заранее не существовать.

Параметры

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

Результат Положение для кнопки.

Example:

```
pos = get_position("Основные", "Команды")
```

axipy.menubar.remove(*action*)

Удаляет кнопку из меню.

Параметры action (Union[QAction, *ToolDefinition*]) – Удаляемая кнопка.

1.8 axipy.render

Модуль отрисовки.

Данный модуль содержит инструменты, предназначенные для отрисовки геопространственных и прочих данных.

Карта

1.8.1 Карта - Map

class axipy.render.Map(*layers=[]*, *shadow=None*)

Базовые классы: PySide2.QtCore.QObject

Класс карты. Рассматривается как единая группа слоев.

__init__(*layers=[]*, *shadow=None*)
Конструктор.

Параметры layers (List) – Список слоев, с которым будет создана карта.

property area_unit
Единицы измерения площадей карты.

Тип результата *AreaUnit*

best_coordsystem()
Определяет координатную системы карты, наиболее подходящую исходя из содержимого перечня слоев.

Тип результата *CoordSystem*

best_rect(*coordsystem=None*)
Определяет ограничивающий прямоугольник карты.

Параметры coordSystem – Координатная система, в которой необходимо получить результат. Если отсутствует, будет выдан результат для наиболее подходящей координатной системы.

Тип результата QRectF

draw(*context*)
Рисует карту в контексте.

Параметры context (Context) – Контекст рисования.

Пример получения карты как растра:

```
map = Map([ worldcap, world ])
image = QImage(1600, 800, QImage.Format_ARGB32_Premultiplied)
image.fill(Qt.white)
painter = QPainter(image)
context = Context(painter)
map.draw(context)
```

property layers
Список слоев.

```
>>>len(map.layers)
2
```

Тип результата *ListLayers*

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[QRectF]) – Ограничивающий прямоугольник. Если не задан, берется у карты.

Тип результата QImage

Результат Выходной растр.

property unit
Единицы измерения координат карты.

Тип результата *LinearUnit*

1.8.1.1 Список слоев карты - ListLayers

class axipy.render.ListLayers(*shadow*)

Базовые классы: object

Перечень слоев карты.

__getitem__(*index*)
Возвращает слой по его индексу или имени.

Параметры **index** (Union[int, str]) – Индекс или имя слоя.

Тип результата *Layer*

Результат Искомый слой.

__len__()
Возвращает количество элементов в списке.

Тип результата int

add(*layer*)
Добавляет слой в карту.

Параметры **layer** (*Layer*) – Добавляемый слой.

at(*idx*)
Возвращает слой по его индексу.

Параметры **idx** (int) – Индекс слоя в списке.

Тип результата *Layer*

by_name(*name*)
Возвращает слой по его имени. Если не найден, возвращает None.

Параметры name (str) – Наименование слоя.

Тип результата *Layer*

count()
Количество слоев.

Тип результата int

move(*from_idx*, *to_idx*)
Перемещает слой в списке слоев по его индексу.

Параметры

- from_idx (int) – Индекс слоя для перемещения.
- to_idx (int) – Целевой индекс.

remove(*idx*)
Удаляет слой по индексу.

Параметры idx (int) – Индекс удаляемого слоя.

Слои карт

1.8.2 Слой - Layer

class axipy.render.Layer(*object_id*, *nameMethod='Layer.create method'*)

Базовые классы: PySide2.QtCore.QObject

Абстрактный базовый класс для слоя карты. Для создания нового экземпляра необходимо использовать axipy.Layer.create().

bounds()
Область в которую попадают все данные, которые могут быть отображены на слое.

Тип результата QRectF

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*.

Пример создания слоя на базе таблицы:

```
table = io.openfile('world.tab')
world = Layer.create(table)
```

data_object()
Источник данных для слоя.

property name
Наименование слоя.

Тип результата str

property opacity

Прозрачность слоя в составе карты. Доступные значения от 0 до 100.

Тип результата int

1.8.2.1 Векторный слой - VectorLayer

class axipy.render.VectorLayer(*shadow*, *_object_id_*)

Базовые классы: *axipy.render.Layer*

Слой, основанный на базе векторных данных.

Примечание: Создание слоя производится посредством метода вызова *Layer.create()*

class Label(*_outer_*)

Базовые классы: object

Метки слоя. Доступны через свойство label.

Зададим в качестве формулы метки атрибут «Страна» и запретим перекрытие меток друг другом:

```
world.label.text = "Страна"
world.label.placementPolicy = VectorLayer.Label.DISALLOW_OVERLAP
```

property placementPolicy

Принцип наложения меток на слой карты.

Допустимые значения: ALLOW_OVERLAP: Допускать перекрытие меток (по умолчанию). DISALLOW_OVERLAP: Не допускать перекрытие меток. TRY_OTHER_POSITION: Пробовать найти для метки новую позицию.

Тип результата int

property text

Наименование атрибута таблицы либо выражение для метки, которое может основываться на одном или нескольких атрибутах..

Тип результата str

property overrideStyle

Переопределяемый стиль слоя. Если задан как None (по умолчанию), объекты будут отображены на основании оформления источника данных.

Пример:

```
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
```

Тип результата *Style*

property showCentroid
Показ центроидов на слое.

Тип результата bool

property thematic
Перечень тематик для данного слоя.

Тип результата *ListThematic*

Перечень тематик для векторного слоя - ListThematic

class axipy.render.ListThematic(*shadow*)

Базовые классы: object

Список тематических слоев (тематик) карты

add(*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*

by_name(*name*)
Получение тематики по ее наименованию.

Параметры name (str) – Наименование запрашиваемой тематики.

Тип результата *ThematicLayer*

Результат Если не найдено, возвращает None.

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.

1.8.2.2 Растровый слой - RasterLayer

class axipy.render.RasterLayer(*_shadow_*, *_object_id_*)

Базовые классы: *axipy.render.Layer*

Класс, который должен использоваться в качестве базового класса для тех слоев, в которых используются свойства отрисовки растрового изображения.

Создание слоя производится посредством метода вызова *Layer.create()*:

```
raster = io.openfile('TrueMarble.tif')
rasterLayer = Layer.create(raster)
```

property transparentColor

Цвет растра, который отображается прозрачным.

Тип результата QColor

Тематика

1.8.3 Тематика - ThematicLayer

class axipy.render.ThematicLayer(*shadow*, *_object_id_*)

Базовые классы: *axipy.render.Layer*

class ShadowFacadeThematic

Базовые классы: axipy.cpp_render.ShadowFacadeLayer

dataChanged = <PySide2.QtCore.Signal object>

init_impl()

staticMetaObject = <PySide2.QtCore.QMetaObject object>

classmethod create(*shadow*)

Создает слой на базе открытой таблицы или растра.

Параметры dataObject – Таблица или растр. В зависимости от переданного объекта будет создан *VectorLayer* или *RasterLayer*.

Пример создания слоя на базе таблицы:

```
table = io.openfile('world.tab')
world = Layer.create(table)
```

1.8.3.1 Интервалы - RangeThematicLayer

class axipy.render.RangeThematicLayer(*expression*, *shadow=None*)

Базовые классы: *axipy.render.ThematicLayer*

Тематическое оформление слоя с распределением значений по интервалам.

__init__(*expression*, *shadow=None*)

Конструктор.

Параметры expression (str) – Наименование атрибута или выражение.

Пример создания тематики с последующим добавлением ее к базовому слою:

```
range = RangeThematicLayer("Население")
range.ranges = 6
range.splitType = RangeThematicLayer.EQUAL_COUNT
world.thematic.add(range)
```

assign_two_colors(*colorMin*, *colorMax*)
Равномерно распределяет оформление по заданным крайним цветам.

Параметры

- colorMin (QColor) – Цвет нижнего диапазона.
- colorMax (QColor) – Цвет верхнего диапазона.

property ranges
Количество интервалов.

Тип результата int

property splitType
Тип распределения значений по интервалам.

Допустимые значения:

EQUAL_INTERVAL: Распределение исходя из равномерности интервалов (по умолчанию).

EQUAL_COUNT: Распределение исходя их равного количества объектов в каждом интервале.

MANUAL: Ручное распределение значений путем задания пределов вручную.

Тип результата int

1.8.3.2 Круговые диаграммы - PieThematicLayer

class axipy.render.PieThematicLayer(*expressions*, *shadow=None*)

Базовые классы: *axipy.render.ThematicLayer*

Тематика в виде круговых диаграмм.

__init__(*expressions*, *shadow=None*)
Конструктор.

Параметры expressions (List) – Наименования атрибутов или выражений.

Пример создания тематики с последующим добавлением ее к базовому слою:

```
pie = PieThematicLayer(["Население", "Мужское", "Женское"])
world.thematic.add(pie)
```

property allocation_type
Тип распределения значений.

Допустимые значения:

LINEAR: Линейное (по умолчанию).

SQRT: Квадратичное.

LOG10: Логарифмическое.

Тип результата int

Легенда

1.8.4 Легенда слоя - Legend

class axipy.render.Legend(*lay*)

Базовые классы: object

Легенда слоя. Позволяет получить информацию об условных обозначениях на слое.

__init__(*lay*)

Конструктор.

Параметры lay (*Layer*) – Слой, для которого создается легенда.

Пример создания легенды для слоя world:

```
legend = Legend(world)
legend.position = QPointF(100, 10)
```

draw(*context*)

Рисует легенду в контексте. Легенду также можно отрисовать совместно с картой в одном контексте (см. *Map.draw()*).

Параметры context (Context) – Контекст рисования.

```
context = Context(painter)
context.rect = QRectF(0,0, 1000, 1000)
legend.draw(context)
```

property position

Положение легенды в контексте рисования.

Тип результата QPointF

to_image(*width*, *height*)

Возвращает легенду в виде растра.

Параметры

- width (int) – Ширина выходного растра.
- height (int) – Высота выходного растра.

Тип результата QImage

Классы, относящиеся к работе с отчетами

1.8.5 Отчет - Report

class axipy.render.Report(*printer*)

Базовые классы: object

План отчета для последующей печати.

draw(*context*)

Выводит отчета в заданном контексте.

Параметры context (Context) – Контекст, в котором будет отрисован отчет.

Пример создания отчета с геометрическим элементом и вывод его в pdf:

```
printer = QPrinter()
printer.setPaperSize(QPrinter.A4)
printer.setOutputFormat(QPrinter.PdfFormat)
printer.setOutputFileName('report.pdf')
painterReport = QPainter(printer)
contextReport = Context(painterReport)

report = Report(printer)
geometryReportItem = GeometryReportItem()
geometryReportItem.geometry = Geometry.from_wkt('POLYGON ((10 10, 10 100, 100 100, 10␣
↪10))')
geometryReportItem.style = Style.from_mapinfo(mi.brush(45, 255, 65535))
report.items.add(geometryReportItem)
report.draw(contextReport)
```

property horisontalPages
Количество страниц отчета по горизонтали.

Тип результата int

property items
Элементы отчета.

property name
Наименование отчета.

property pageSize
Размеры одного листа отчета.

Тип результата QSizeF

property unit
Единицы измерения в отчете.

Тип результата PaperUnit

property verticalPages
Количество страниц отчета по вертикали.

Тип результата int

1.8.5.1 Список элементов отчета - ReportItems

class axipy.render.ReportItems(*_shadow_)

Базовые классы: object

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.

`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.

1.8.6 Элемент отчета - ReportItem

`class axipy.render.ReportItem`

Базовые классы: object

Базовый класс элемента отчета.

`property rect`
Размер (ограничивающий прямоугольник) элемента отчета в единицах измерения отчета.

Тип результата QRectF

1.8.6.1 Элемент отчета: геометрия - GeometryReportItem

`class axipy.render.GeometryReportItem`

Базовые классы: *axipy.render.ReportItem*

Элемент отчета типа геометрия.

`__init__()`
Конструктор.

`property geometry`
Геометрическое представление объекта.

Тип результата *Geometry*

`property style`
Стиль геометрического представления объекта.

Тип результата *Style*

1.8.6.2 Элемент отчета: карта - MapReportItem

`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.

```
mapReportItem = MapReportItem(QRectF(10, 10, 200, 100), map)
mapReportItem.scale = 200000000
report.items.add(mapReportItem)
```

__init__(*rect*, *map*)
Конструктор.

Параметры

- rect (QRectF) – Размер элемента отчета в единицах измерения отчета.
- map (*Map*) – Карта, на базе которой будет создан элемент отчета.

property center
Центр карты в координатах карты.

Тип результата QPointF

map()
Возвращает элемент типа карта, на основании которой создается элемент отчета.

Тип результата *Map*

property scale
Текущее значение масштаба карты.

Тип результата float

1.8.6.3 Элемент отчета: растр - RasterReportItem

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.

Растр на базе URL:

```
rasterReportItem = RasterReportItem(QRectF(10, 10, 140, 70), 'https://upload.wikimedia.org/
↪wikipedia/en/7/72/World_Map_WSF.svg.png')
report.items.add(rasterReportItem)
```

Растр на базе локального файла:

```
rasterReportItem = RasterReportItem(QRectF(10, 10, 140, 70), 'TrueMarble.tif')
report.items.add(rasterReportItem)
```

__init__(*rect*, *data*)
Конструктор.

Параметры

- rect (QRectF) – Размер элемента отчета в единицах измерения отчета.
- data (str) – Путь к растровому файлу или его URL.

- genindex

- modindex
- search

Содержание модулей Python

a

Символы

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

модуль