Разбор домашнего задания

Роман Булгаков
Спикер курса

Skillbox

Инкапсуляция и сокрытие данных. Геттеры и сеттеры

Роман Булгаков
Спикер курса

Skillbox

Первый принцип ООП

```python
class Person:
    __count = 0

    def __init__(self, name, age):
        self.name = name
        self.age = age
        Person.__count += 1
```

Объединение атрибутов и методов объекта **в одну сущность.**

__count — приватный атрибут **(сокрытие данных).**

Инкапсуляция — это механизм языка, позволяющий **объединить данные и методы**, работающие с этими данными, в единый объект и **скрыть детали реализации** от пользователя.

Сокрытие данных в Python

```python
class Person:
    __count = 0

    def __init__(self, name, age):
        self.__name = name
        self.__age = age
        Person.__count += 1
```

“__” перед именем — это соглашение между программистами. Такое имя предусмотрено для использования внутри класса.

```python
print(Person.__count)
```

→ выводится ошибка

```python
print(Person._Person__count)
```

→ выводится 0
(но так делать не стоит)

Для изменения или получения приватных данных используйте геттеры и сеттеры!

Итоги урока

- _count = 0 #сокрытие данных
- Инкапсуляция — объединение данных и методов в единый объект и сокрытие реализации от пользователя
- ```
  def get_name(self): #геттер
      return self.__name
  def set_age(self, name): #сеттер
      self.__name = name
  ```
- Person._Person__count #лучше не надо
- def __do_action(self): #приватный метод


  ```

Наследование

Роман Булгаков
Спикер курса

Skillbox

Задача «Животные»

Входные данные:

- класс «Кот»
- класс «Собака»

Атрибуты класса «Кот»:

1. 4 лапы
2. Есть хвост

Методы класса Кот:

1. Издаёт звук «Мяу»

Выходные данные:

- экземпляры классов

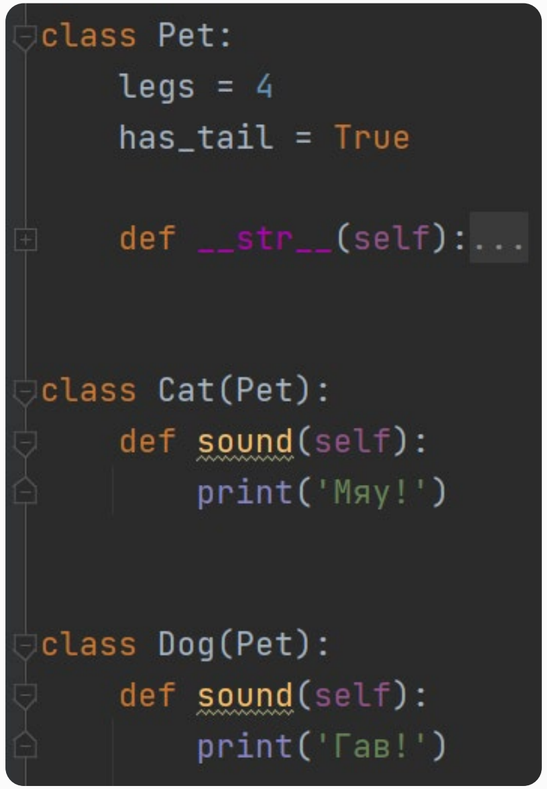
Атрибуты класса «Собака»:

1. 4 лапы
2. Есть хвост

Методы класса Собака:

1. Издаёт звук «Гав»

Второй принцип ООП



```python
class Pet:
    legs = 4
    has_tail = True

    def __str__(self):...


class Cat(Pet):
    def sound(self):
        print('Мяу!')


class Dog(Pet):
    def sound(self):
        print('Гав!')
```

Класс Pet называется **базовым классом** (родительским классом, а также суперклассом).

Классы Cat и Dog называются **подклассами** (дочерними классами).

Наследование — это механизм языка, позволяющий создавать **новый класс на основе уже существующего** класса.

Используется для выделения общих атрибутов и методов объектов.

Задача «Корабли»

Входные данные:

- класс «Грузовой корабль»
- класс «Военный корабль»

Выходные данные:

- экземпляры классов

Атрибуты класса «Кот»:

1. Модель
2. Заполненность (0)

Методы класса «Кот»:

1. Сказать модель
2. Идти по воде
3. Погрузить груз
4. Выгрузить груз

Атрибуты класса «Собака»:

1. Модель
2. Оружие

Методы класса «Собака»:

1. Сказать модель
2. Идти по воде
3. Атаковать

Итоги урока

- class Pet:...
  class Cat(Pet):...

- Pet — базовый класс
  Cat — подкласс

- cat = Cat()
  print(cat.legs)

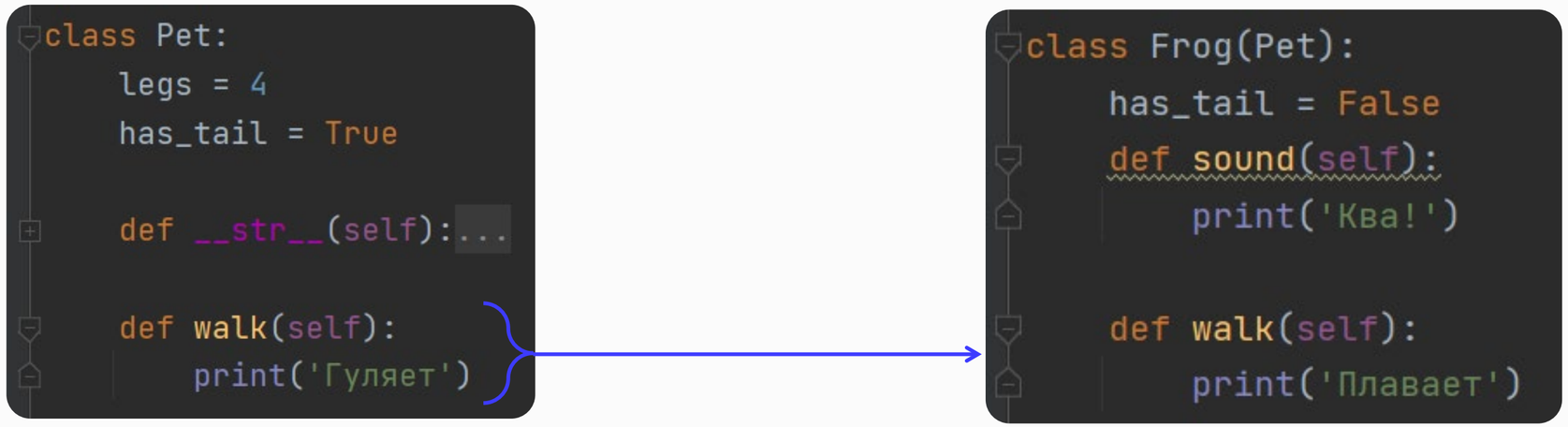
- class CargoShip(Ship):
  def __init__(self, model):
  super().__init__(model)
  self.tonnage_load = 0

Полиморфизм

Роман Булгаков
Спикер курса

Skillbox

Третий принцип ООП



Базовый класс

Подкласс

Полиморфизм — принцип, предполагающий **способность к изменению функционала**, унаследованного от базового класса.

Документация. Описание классов и методов

Роман Булгаков
Спикер курса

Skillbox

Документация

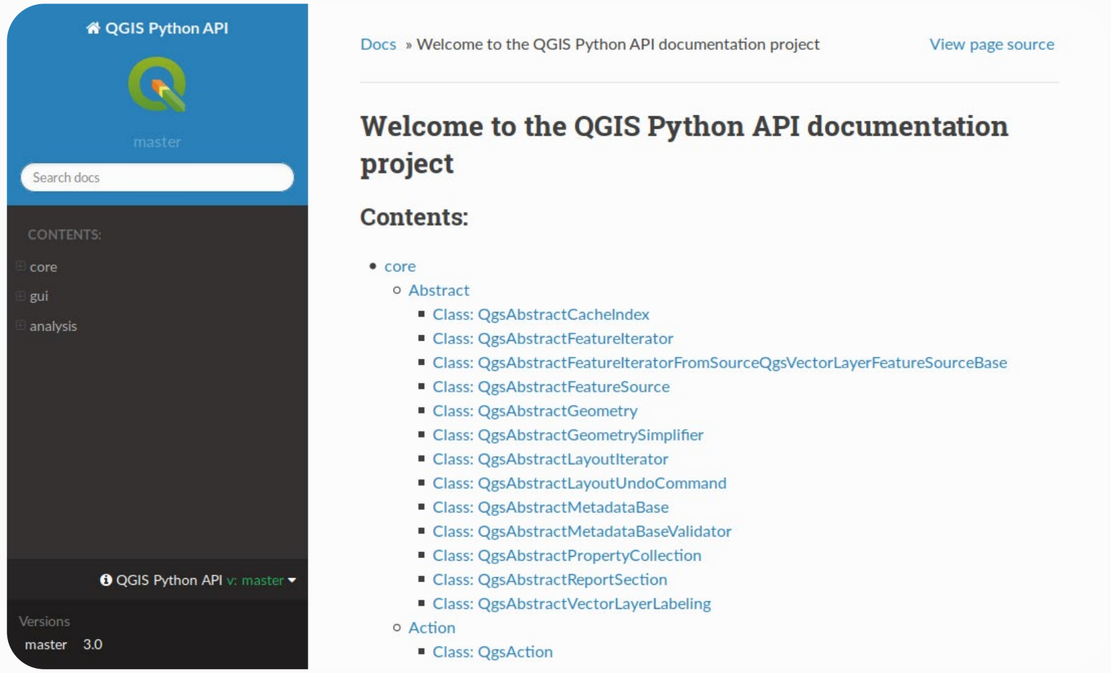
SAMSUNG

МИКРОВОЛНОВАЯ ПЕЧЬ

Инструкция для пользователя и руководство по приготовлению пищи

GE87KR

Инструкция к технике



QGIS Python API

master

Search docs

CONTENTS:

core

gui

analysis

QGIS Python API v: master

Versions

master 3.0

Docs » Welcome to the QGIS Python API documentation project View page source

Welcome to the QGIS Python API documentation project

Contents:

- core
  - Abstract
    - Class: QgsAbstractCacheIndex
    - Class: QgsAbstractFeatureIterator
    - Class: QgsAbstractFeatureIteratorFromSourceQgsVectorLayerFeatureSourceBase
    - Class: QgsAbstractFeatureSource
    - Class: QgsAbstractGeometry
    - Class: QgsAbstractGeometrySimplifier
    - Class: QgsAbstractLayoutIterator
    - Class: QgsAbstractLayoutUndoCommand
    - Class: QgsAbstractMetadataBase
    - Class: QgsAbstractMetadataBaseValidator
    - Class: QgsAbstractPropertyCollection
    - Class: QgsAbstractReportSection
    - Class: QgsAbstractVectorLayerLabeling
  - Action
    - Class: QgsAction

Документация к коду

Итоги модуля

- _count = 0 #сокрытие данных

- def get_name(self): #геттер
  def set_age(self, name): #сеттер

- class Person:
  def __init__(self, name):
  self._name = name

- class Pet:...
  class Cat(Pet):...

- def walk(self): # есть в Pet
  print('Бегает в колесе')

- docstring