Занятие 9.
ООП. Классы

1

ООП. Классы

ООП

Что это такое?

- Python — мультипарадигмальный язык программирования. Он поддерживает разные подходы к программированию.
- Один из популярных подходов к решению проблем — создание объектов. Это называется объектно-ориентированным программированием (ООП).
- У объекта есть две характеристики:
  - атрибуты;
  - поведение.

Класс

Что это такое?

- **class -** фабрика по производству объектов (людей), наш собственный тип данных
- **self -** ссылка на экземляр класса
- **экземпляр -** объект класса (конкретный человек)
  -

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

```
person_1 = Person("Lena", 24)
```

```
person_1.name
```

```
'Lena'
```

метод __init__

Что это такое?

- это магический метод, который самостоятельно вызывается при создании экземпляра класса
- в нем создаются атрибуты экземпляра

```
class Person:
    def __init__(self, name, age):
        print("Я вызываюсь каждый раз при создании экземпляра класса")
        self.name = name
        self.age = age
```

```
person_1 = Person("Lena", 24)
```

```
Я вызываюсь каждый раз при создании экземпляра класса
```

Методы

Что это такое?

- функция, относящаяся к экземпляру класса (например, человек умеет говорить)
- методы определяют **поведение**

```
class Person:
    def __init__(self, name, age=None):
        self.name = name
        self.age = age

    def say_hi(self):
        print(f"Hi! My name is {self.name}")

person_1 = Person(name="Lena")
person_1.say_hi()
```

```
Hi! My name is Lena
```

1

Четыре принципа ООП

Наследование

Что это такое?

- дочерний класс содержит в себе все атрибуты родительского, при этом некоторые атрибуты могут быть переопределены или добавлены в дочернем

Зачем?

- Чтобы сократить количество дублированного кода

```
class Animal:

    def __init__(self, name):
        self.name = name

    def make_sound(self):
        print("**animal sound**")
```

```
class Cat(Animal):

    def make_sound(self):
        print("meow")

cat = Cat("Каспер")
cat.make_sound()
```

```
meow
```

Полиморфизм

- используем единственную сущность для разного поведения

Родительский класс

```
class Animal:

    def __init__(self, name):
        self.name = name

    def make_sound(self):
        print("**animal sound**")
```

```
class Cat(Animal):

    def make_sound(self):
        print("meow")

class Dog(Animal):

    def make_sound(self):
        print("wow")

cat = Cat("Базилио")
cat.make_sound()
```

```
meow
```

```
dof = Dog("Стрелка")
dof.make_sound()
```

```
wow
```

Инкапсуляция

- ограничение доступа к составляющим объект компонентам (методам и переменным). Инкапсуляция делает некоторые из компонент доступными только внутри класса
- в python она работает только на уровне соглашений

```
class A:
    def _private(self):
        print("Это приватный метод!")
a = A()
a._private()
```

```
Это приватный метод!
```

```
class B:
    def __private(self):
        print("Это приватный метод!")

b = B()
b.__private()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[87], line 9
      6         print("Это приватный метод!")
      8 b = B()
----> 9 b.__private()

AttributeError: 'B' object has no attribute '__private'
```

Абстракция

- мы хорошо знакомы с основными функциями смартфона - камера, диктофон и тд, но мы зачастую не знаем, как они работают, мы знаем, как ими пользоваться
- Зачем? - для сокрытия излишней сложности - пользователю необязательно знать, как что-то реализовано, важно только то, что функция делает

3

Магические методы

Магические методы

- это методы, которые вызываются при каком-то действии, например, при создании экземпляра класса вызывается уже знакомый нам метод __ init __

```
# __del__ вызывается при удалении экземпляра класса

class Example:
    def __del__(self):
        print("Я вызываюсь перед тем как мы удалим объект")
        del self
```

```
ex = Example()
```

```
del ex
```

```
Я вызываюсь перед тем как мы удалим объект
```

4

DataClass

Датаклассы

- классы для хранения данных
- в нем нет такой логики как в обычных классах (say_hi, make_sound и тп)
- по умолчанию есть методы __ init __, __ repr __, __ eq __

```python
from dataclasses import dataclass


@dataclass
class Coordinate:
    x: int
    y: int
    z: int
```

```python
a = Coordinate(4, 5, 3)
a
```

```
Coordinate(x=4, y=5, z=3)
```

```python
# можно задавать значения по умолчанию
@dataclass
class Coordinate:
    x: int = 0
    y: int = 0
    z: int = 0
```

```python
a = Coordinate()
a
```

```
Coordinate(x=0, y=0, z=0)
```

5

Домашнее задание

Домашнее задание (часть 1)

Задача 1. Создать родительский класс машина, у которого есть атрибуты model, age, color и weight, из них обязательный только model. Также у класса должны быть методы move, stop, birthday, методы move и stop выводят сообщение на экран "move", "stop", а birthday увеличивает атрибут age на 1. Если атрибут age = None, то выбрасывает исключение с сообщением "атрибут age не задан".

Задача 2. Есть csv файл со списком людей, нужно прочитать его и преобразовать в список датаклассов. То есть нужно создать датакласс с атрибутами name, age, при этом тип age : Optional[int]. У датакласса должно быть property birth_year, которое считает возраст

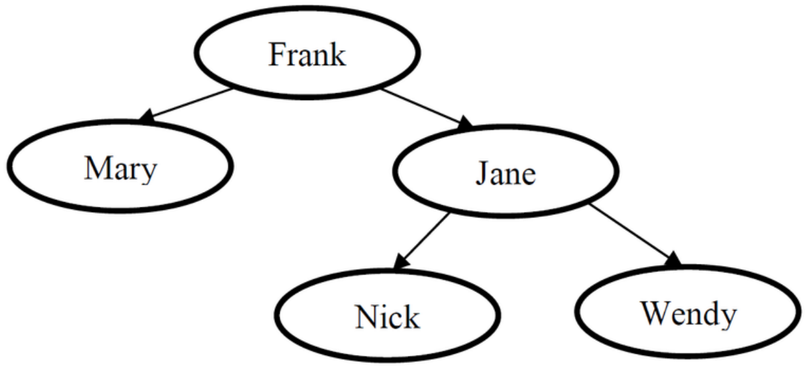
name	age
lena	24
dima	
vova	35

Задача 3. Пройти тест в чатике

Домашнее задание (часть 2)

Задача 4*. Доработать созданный на уроке класс Family следующим образом:

- создать методы, которые умеют возвращать детей по роли в семье (список FamilyMember), а также мать, отца и домашних животных
- доработать класс FamilyMember таким образом, чтобы у члена семьи могли быть mother и father - тоже экземпляры FamilyMember, эти атрибуты опциональны, то есть у человека может отсутствовать mother и father
- добавить метод print_family_tree, который будет выводить семейное древо (формат вывода неважен):



Подсказка: вначале создаем father, mother, потом детей, в которых передадим father и mother и тд. И так сформировать хотя бы 3 поколения. Также все эти father, mother и children должы войти в Family. При этом можно использовать методы FamilyMember для печати родителей, так как он знает о них

Спасибо за внимание!