ООП
методы
наследование

Повторение

1. Выберите дверь
2. Нажмите на нее, чтобы войти







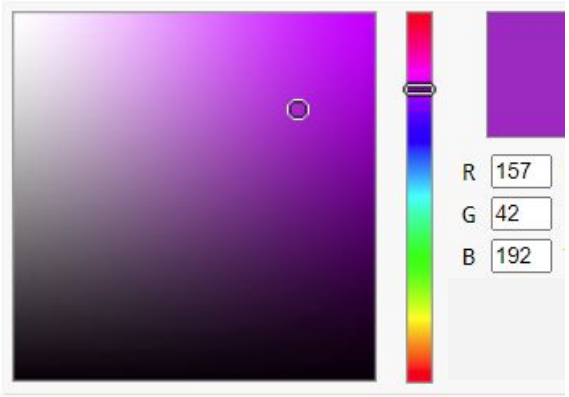

Красная дверь

Что такое ООП?

Фиолетовая дверь

Напишите код создания объекта класса Color.

```
class Color:
    def __init__(self, r, g, b):
        self.r = r
        self.g = g
        self.b = b
```



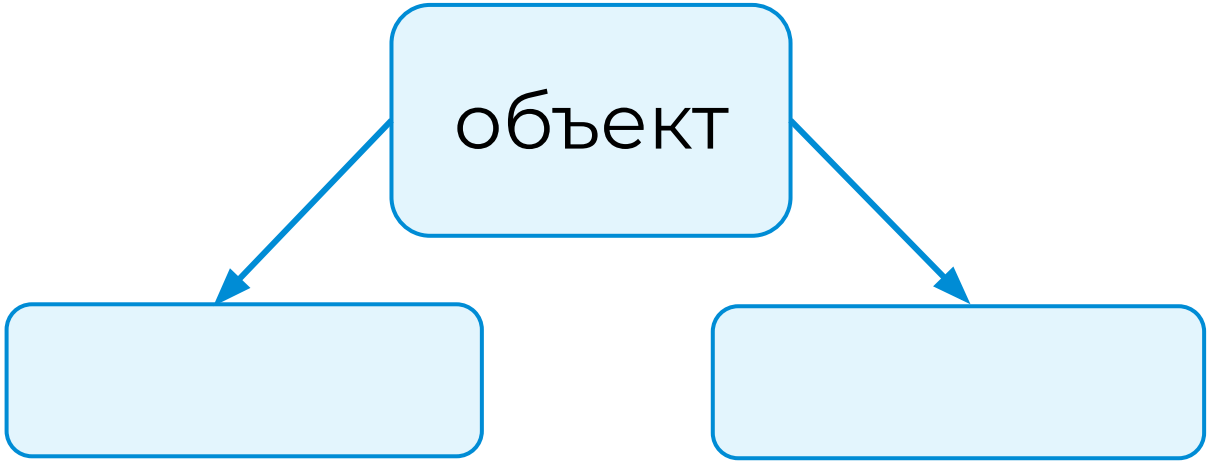
Голубая дверь

Что такое объект?

Приведите пример.

Бирюзовая дверь

Заполните схему и приведите пример на каждую ветку.



Розовая дверь

Разделите на категории эти слова:

- длина клюва
- птица
- пеликан
- цвет оперения
- летать
- плавать
- страус
- возраст
- живое существо

класс
объект
атрибут
действие

Зеленая дверь

Найдите все ошибки в коде и исправьте их.

```
def Pet:
    clas init (my_name)
        name = my_name
        type = ""
        weight = 0
```

Коричневая дверь

Что такое init в классе?

Зачем функция init нужна?

Как передать в init значения?

Оранжевая дверь

Что такое класс?

Приведите пример.

Домашнее задание

Сегодня

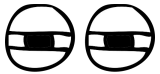
Научим наши объекты действовать

Узнаем, как объекту получить наследство от родителя

Научимся экономить время при создании класса

Что такое метод?

У каких типов данных мы уже видели методы?

их минимум 4 штуки 

Какие методы мы уже знаем?

(примеры в студию)

Можно ли применить метод строки к списку?

```
s1 = "Привет"
s2 = ["a", "b", "c"]
s1.replace("и", "ы")
s2.replace("a", "e")
```

Работа в парах

7 минут

Обсудите и запишите ответы

1. Что такое метод класса?
2. Приведите пример метода класса

   (можно из жизни, а не в коде)
3. Как вызвать метод класса?

   (можно кодом)

```
class Color:
    def __init__(self, r, g, b):
        self.r = r
        self.g = g
        self.b = b
```

4. Дан класс Color. Напишите код метода **add_red**, который:

   - увеличивает значение параметра red на 20

   - если число превышает 255, то red устанавливается = 255

Ответы

Кто готов представить свои ответы группе?

Зачем нужны методы?

Зачем нужны методы?

- Чтобы класс был не только хранилищем свойств, как словарь
- Чтобы объект мог быстро и удобно вызывать действия для работы с самим собой
  Например, добавление элемента в список. Это действие, которое повторяется очень часто
- Экономим время и длину кода на повторяющихся фрагментах

Важно! self

```
class Color:
    def __init__(self, r, g, b):
        self.r = r
        self.g = g
        self.b = b
    def add_red(self):
        self.r += 20
        if self.r > 255:
            self.r = 255
```

Обязательно передаем **self**! Чтобы обращаться к тому объекту, который вызвал метод

Практика

15 минут

Метод класса

Метод в ООП — функция, принадлежащая какому-то классу.

Метод имеет доступ к данным объекта, к которому будет применяться.

Пример метода класса

Объект: котик

Свойства:

- цвет: рыжий
- вес: 10 кг

Действия:

- есть
- спать

```
действие есть(кг):
# кг - сколько съел
    вес = вес + кг
```

Вызов метода

На примере строки или списка:

```
s1 = "ПрИвЕт"
s2 = [5, 2, 4]
s1 = s1.lower()
s2.sort()
```

На примере класса:

```
p = Pet("Мурзик", "кот", 4)
p.eat()
```

Как написать метод?

```
class Color:
    def __init__(self, r, g, b):
        self.r = r
        self.g = g
        self.b = b
    def print_info(self):
        print("Ваш цвет =", self.r, self.g, self.b)
```

Привет, я метод!

Как написать метод?

```
class Color:
    def __init__(self, r, g, b):
        self.r = r
        self.g = g
        self.b = b
    def add_red(self):
        self.r += 20
        if self.r > 255:
            self.r = 255
```

Привет, я метод!

Теория: наследование

7 минут

Ситуация:
Вы с командой работаете над компьютерной игрой по мотивам Marvel
В игре есть персонажи двух типов:

1. Обычные люди
2. Супергерои

У обычных людей много методов и атрибутов: имя, адрес, возраст, есть, спать, говорить, ходить....

Супергерои имеют всё тоже самое + некоторые супер действия (летать, стрелять паутиной и т.д.) или атрибуты (оружие: молот)

Как избежать дублирования кода?
Как оптимизировать идею классов в данной ситуации?

Идея наследования

Класс: **человек**

Свойства:

- имя
- возраст

Действия:

- спать

Класс: **человек-паук**

Взять всё у человека

+

Действия:

- пускать паутину

Наследование

Наследование в ООП — это концепция, при которой дочерний класс перенимает (наследует) у родительского класса атрибуты и методы.

Родительский - исходный, первоначальный

Дочерний - тот, который копирует, появляется после родительского

Наследование классов (код)

```
class Human():
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def sleep(self):
        print("Сплю! Не будите", self.name)
class SpiderMan(Human):
    def make_spidernet(self):
        print("Выпускаю паутину!")
```

```
p1 = Human("Олег", 12)
p2 = SpiderMan("Паркер",18)
p1.sleep()
p2.sleep()
p2.make_spidernet()
```

Результат:

```
Сплю! Не будите Олег
Сплю! Не будите Паркер
Выпускаю паутину!
```

Наследование классов

```
class Human():
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def sleep(self):
        print("Сплю! Не будите", self.name)
class SpiderMan(Human):
    def make_spidernet(self):
        print("Выпускаю паутину!")
```

Практика

20 минут

Итоги

Умеем писать методы классов

Узнаем делать наследование

Умеем вызывать методы класса

В блоке учебника вас ждет секретик

Домашнее задание:

вопросы на понимание

и задача о том, как вас спасли с острова!