Теоретический материал к занятию Основы ООП 1 занятие.

Python - объектно-ориентированный язык, а соответственно все, что мы не видим является объектом какого либо класса. Узнать тип объекта нам поможет функция type()

```
def func():
    pass

class Test:
    pass
print(type(5))
print(type('a'))
print(type(5.5))
print(type(True))
print(type([1, 2]))
print(type((5, 4)))
print(type({1, 2}))
print(type({'a': 2}))
print(type(Test))
print(type(func))
```

Результат

```
<class 'int'>
<class 'str'>
<class 'float'>
<class 'bool'>
<class 'list'>
<class 'tuple'>
<class 'set'>
<class 'dict'>
<class 'type'>
<class 'function'
```

Класс можно представить, как некий шаблон описывающий общие свойства и возможные действия. Например, в любой игре огромное количество персонажей и писать под каждого из них отдельный код, было бы очень накладно и ресурсоемко. Все персонажи имеют какие то общие характеристики(переменные), такие как например жизни, имя и т.п, а также набор каких то методов, например ходить, говорить, стрелять.
Метод - та же функция, но написанная внутри класса.
Объект же, если брать в пример игру, то это собранный на основе класса персонаж.

Для создания класса используется ключевое слово class. Давайте создадим простой класс некого персонажа.

```
class Person:
    name = 'Иван'
    age = 'Иванов'

    def say():
        print('Hello')
```

Создадим экземпляр класса и попробуем выведем его имя и вызовем метод say()

```
class Person:
    name = 'Иван'
    age = 'Иванов'

    def say():
        print('Hello')

person1 = Person
print(person1.name)
person1.say()
```

Чтобы создать экземпляр класса необходима всего одна строка. Если бы мы создавали наших персонажей без использования классов, то это заняло бы на порядок больше строк, а если бы потребовалось изменить какую либо общую характеристику, то пришлось бы менять у каждого персонажа по отдельности.

В метод __init__ мы можем передать необходимые значения и сохранить их в локальные переменные, значения которых будут разные для разных объектов.

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


person1 = Person('Иван', 15)
person2 = Person('Петр', 14)
print(person1.name)
print(person1.age)
print(person2.name)
print(person2.age)
```

Теперь у каждого объекта свое имя и свой возраст. Помочь программе, понять какое значение принадлежит объекту, позволяет self, которые ссылается именно на объект, который обратился к данному свойству.

Метод init, называют конструктором класса или же методом инициализации.