ДЕТСКАЯ ОНЛАЙН-ШКОЛА ПРОГРАММИРОВАНИЯ
HELLO WORLD

#ШПАРГАЛОЧКИ

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ PYTHON

Начальный уровень

Материалы подготовлены отделом методической разработки

Больше полезных материалов и общения в нашем комьюнити в Telegram: https://t.me/hw_school

Классы

Класс – это пользовательский тип данных, который описывает какой-то объект (персонажа игры, человека, животное, машину и т.д.)

Чтобы создать класс, нужно написать ключевое слово class и имя класса. Имя должно начинаться с большой буквы:

class Person:

Класс может описывать как свойства объекта (имя, рост и возраст человека, вид животного, марку машины), так и его способности (что объект умеет делать - говорить, бегать, сражаться).

Созданные внутри класса переменные называются свойствами, а функции - методами. У каждого класса есть специальный метод **__init__** - это **конструктор** класса, он автоматически вызывается при создании объекта и нужен для указания его первоначальных свойств.

Объект - это экземпляр класса, чтобы создать его, нужно создать переменную, а после знака равно написать имя класса и круглые скобки:

student = Person()

Чтобы созданные внутри одного метода свойства были доступны внутри всего класса, их нужно связать с объектом класса с помощью слова self. Имя свойства пишется через точку - **self.name**. Кстати, self - обязательный аргумент любого метода. Но его нужно указывать только при объявлении метода, при вызове self передается автоматически.

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

Чтобы вызвать метод объекта или получить доступ к свойству, нужно написать имя объекта и имя метода или свойства через точку:

```
student.say_hello() # вызов метода say_hello объекта student
```

Наследование позволяет не прописывать одинаковые свойства и методы классам со схожим поведением, а наследовать их от общего родителя. **Класс-родитель** указывается при создании **класса-наследника** в скобках после его имени:

```
class Dog(Animal):
```

Важно учитывать, что прописав в классе-наследнике метод с тем же именем, что и у класса-родителя, мы **переопределим** его, т.е. новый метод заменит метод родителя. Чтобы сохранить поведение родительского метода, нужно вызвать его в методе класса-наследника с помощью команды super():

```
def __init__(self):
    super().__init__()
```

`class Car:` `  speed = 100`	Создание класса
`porshe = Car()`	Создание экземпляра (объекта) класса
`super()`	Получение экземпляра класса-родителя
`__init__()`	Конструктор класса, задает первоначальные значения
`self`	Делает созданное свойство доступным во всех методах класса