

Python

22 урок. Классы

Что такое Классы?

Python - это объектно-ориентированный язык программирования.

Создание нового класса создает новый тип объекта, позволяя создавать новые экземпляры этого типа. К каждому экземпляру класса могут быть прикреплены атрибуты для поддержания его состояния. Экземпляры классов также могут иметь методы (определяемые их классом) для изменения их состояния.

Некоторые моменты в классе Python:

Классы создаются по ключевому слову `class`.

Атрибуты - это переменные, принадлежащие классу.

Атрибуты всегда являются общедоступными, и к ним можно получить доступ с помощью оператора точка (`.`).

Например: `Myclass.Myattribute`

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

p1 = Person("John", 36)

print(p1.name)
print(p1.age)
# John
# 36
```

Создать класс

```
class MyClass:
```

```
    x = 5
```

```
print(MyClass)
```

```
# <class '__main__.MyClass'>
```

Создайте класс с именем MyClass
со свойством с именем x

Создать объект

```
class MyClass:
```

```
    x = 5
```

```
p1 = MyClass()
```

```
print(p1.x)
```

```
# 5
```

Теперь мы можем использовать класс MyClass для создания объектов.

Создайте объект с именем p1 и распечатайте значение x

Функция `__init__()`

Приведенные выше примеры представляют собой классы и объекты в их простейшей форме и не очень полезны в реальных приложениях. Используйте функцию `__init__()` для присвоения значений свойствам объекта или других операций, которые необходимо выполнять при создании объекта.

```
class Person:
```

```
    def __init__(self, name, age):
```

```
        self.name = name
```

```
        self.age = age
```



Создайте класс с именем Person



Используйте функцию `__init__()` для присвоения значений имя и возраст

```
p1 = Person("John", 36)
```

```
print(p1.name)
```

```
print(p1.age)
```

Методы

Объекты также могут содержать методы. Методы в объектах - это функции, принадлежащие объекту.

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



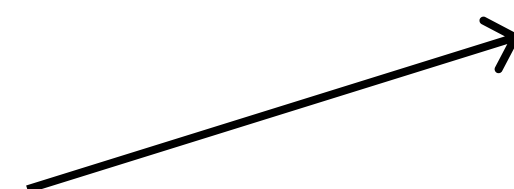
Создадим метод в классе Person

```
def myfunc(self):
    print("Hello my name is " + self.name)
```



Вставьте функцию, которая печатает приветствие, и выполните ее для объекта p1

```
p1 = Person("John", 36)
p1.myfunc()
```



Параметр self

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

    def myfunc(abc):
        print("Hello my name is " + abc.name)
```

```
p1 = Person("John", 36)
p1.myfunc()
```

Параметр self является ссылкой на текущий экземпляр класса и используется для доступа к переменным, принадлежащим этому классу. Его не обязательно называть self, вы можете называть его как хотите, но он должен быть первым параметром любой функции в классе.

Изменить свойства объекта

Объекты также могут содержать методы. Методы в объектах - это функции, принадлежащие объекту.

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

    def myfunc(self):
        print("Hello my name is " + self.name)
```

→ Создадим класс Person

```
p1 = Person("John", 36)
p1.age = 40
print(p1.age)
```

→ Вы можете изменять свойства таких объектов. Установите возраст p1 на 40.

Удалить свойства объекта Оператор pass

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

    def myfunc(self):
        print("Hello my name is " + self.name)
```

```
p1 = Person("John", 36)
```

```
del p1.age
```

```
print(p1.age)
```



Удалите свойство age из объекта p1

Определения классов не могут быть пустыми, но если по какой-то причине у вас есть определение класса без содержимого, вставьте оператор pass, чтобы избежать ошибки.

```
class Person:
    pass
```

Функция `isinstance()`

Функция `isinstance()` возвращает значение `True`, если указанный объект имеет указанный тип, иначе `False`.

Если параметр типа является кортежем, эта функция вернет значение `True`, если объект является одним из типов в кортеже.

`isinstance(object, type)`

```
x = isinstance(5, int)
```

```
x = isinstance("Hello", (float, int, str, list, dict, tuple))
```

Проверьте, является ли у экземпляром `myObj`:

```
class myObj:
```

```
    name = "John"
```

```
y = myObj()
```

```
x = isinstance(y, myObj)
```