# ООП - Классы, атрибуты, методы, конструктор

**№ урока:** 1 **Курс:** Python Essential

**Средства обучения:** PyCharm

## Обзор, цель и назначение урока

После завершения урока обучающиеся будут иметь представление о парадигме объектно-ориентированного программирования, смогут создавать классы и объекты в программах на Python.

## Изучив материал данного занятия, учащийся сможет:

- Понимать основы парадигмы объектно-ориентированного программирования
- Понимать назначение и различие между классами и объектами
- Создавать классы
- Создавать экземпляры классов
- Создавать методы

## Содержание урока

1. Что такое парадигма программирования?
2. Что такое ООП?
3. Что такое класс?
4. Что такое метод?
5. Что такое объект?
6. Реализация в Python

## Резюме

Программирование - это сфера, где одну и ту же задачу можно решить несколькими способами. Вот лишь несколько примеров, как можно найти сумму чисел:

```
FIRST_NUMBER = 8
SECOND_NUMBER = 2
THIRD_NUMBER = 10

# Variant №1
print(FIRST_NUMBER + SECOND_NUMBER + THIRD_NUMBER)

# Variant №2
numbers_to_sum = [FIRST_NUMBER, SECOND_NUMBER, THIRD_NUMBER]
print(sum(numbers_to_sum))

# Variant №3
def sum_numbers_list(numbers_list, current_value=0):
    new_number = numbers_list.pop()
    new_value = current_value + new_number

        if len(numbers_list) != 0:
        return sum_numbers_list(numbers_list, new_value)
    else:
        return new_value

print(sum_numbers_list(numbers_to_sum))
```

Все 3 варианта выведут значение "20". Но получено значение разными способами.
Вариант 1 - самый простой: с помощью операции "+" сложили все числа.
Вариант 2 использует встроенную функцию sum, чтобы найти сумму списка чисел.



CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

И наконец-то вариант 3, что демонстрирует частую проблему программистов - изобретение собственного колеса. Этот вариант самый объёмный и бессмысленный, ведь использует лишние конструкции.

**Совет**

**Не пишите лишний код, если есть возможность воспользоваться встроенной функцией Python. Умение решать задачу лаконично - отличная черта программиста.**

3 примера - три разных решения, один результат. Программисты думали по-разному, когда решали одинаковую задачу - у каждого свой подход. Именно такой подход и называется **парадигма**.

**Парадигма программирования** — это совокупность идей и понятий, определяющих стиль написания компьютерных программ (**подход** к программированию). Это способ концептуализации, **определяющий организацию** вычислений и структурирование кода.

Если к вам попадает проект другого программиста, иногда приходится потратить немало времени, чтобы разобраться в его **парадигме**. А если в месяц вам пригодится работать с 6-ю такими проектами? Каждый раз привыкать к новой парадигме другого программиста?

К счастью, нет. В программировании есть две лидирующие парадигмы: **объектно-ориентированная** и **функциональная парадигмы**. Так же есть множество других парадигм, которые используются программистами, но существенно реже.

Зачастую, чтобы в полной мере воплотить в коде ту или иную парадигму, нужна поддержка со стороны синтаксиса языка. Например, чтобы решить задачу в стиле функционального программирования, язык обязательно должен поддерживать создание функций (**не только**).

**Объектно-ориентированная парадигма** - это подход к программированию, когда все в коде представляется в виде сущностей (объектов) и их методов. Например, вы сейчас используете объект компьютер: у этого объекта есть интерфейс для пользователя и этот интерфейс предоставляет доступ к ряду функционала (методов). А задумывались ли вы, что этот компьютер кто-то придумал: функция доступа в интернет, возможность посмотреть фильм - все эти методы были кем-то спроектированы. Другими словами, ваш компьютер был создан по плану\эскизу.

Если бы мы хотели написать свой компьютер на несуществующем языке, это выглядело бы примерно так:

```
схема Компьютер:
    метод зайти_в_интернет():
        вернуть окно_баузера

    метод открыть_урок_itvdn():
        окно_браузера = зайти_в_интернет()
        окно_браузера.перейти_на("itvdn.com")

        вернуть окно_баузера

    # ...
    # ...
    # ...
    # ... и так другие функции компьютера
```

Мы создали с вами схему компьютера. Схема — это только записанная идея, но не реальный компьютер. Чтобы создать "реальный" компьютер и им воспользоваться надо создать объект:

мой_компьютер = Компьютер()

С помощью вызова схемы, как обычной функции Python, мы можем создавать реальные объекты. Но как теперь открыть урок на ITVDN? Для этого нам необходимо "вызвать" нужный метод **на объекте**:

мой_компьютер.открыть_урок_itvdn()

**Важные аспекты ООП: наследование, полиморфизм, инкапсуляция**. О них мы поговорим на следующих занятиях.

Это самые базовые принципы, которые помогут вам думать в стиле ООП.

**Класс** - это схема будущего объекта. Возвращаясь к компьютеру, запишем код с классом:

```
class Компьютер:
    ИМЯ_КОМПЬЮТЕРА = "CBS"
```



CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

```
    метод зайти_в_интернет():
        вернуть окно_баузера

    метод открыть_урок_itvdn():
        окно_браузера = зайти_в_интернет()
        окно_браузера.перейти_на("itvdn.com")

        вернуть окно_баузера
```

Мы заменили слово "схема" на конструкцию class, что предоставляет нам Python. В классе мы можем описать методы объекта и поля данных.

**Что такое метод?**

Мы уже увидели, что в схемах (class) мы **описываем**, какой функционал должен иметь наш будущий объект с помощью методов. В Python это делается почти так же, как в нашем псевдокоде. Каждый метод класса объявляется с помощью ключевого слов def как функция:

```
class Компьютер:
    ИМЯ_КОМПЬЮТЕРА = "CBS"

    def зайти_в_интернет(self):
        return окно_баузера

    def открыть_урок_itvdn(self):
        окно_браузера = зайти_в_интернет()
        окно_браузера.перейти_на("itvdn.com")

        return окно_баузера
```

Что изменилось:

- Мы изменили слово "метод" на конструкцию Python def.
- Слово "вернуть" изменили на знакомое нам return
- Каждый метод-функция принимает в качестве первого аргумента self (об этом чуть позже)

**Поля данных** - это переменные объявленные внутри класса. Эти переменные получают все его объекты. Иногда поля данных называют свойствами объекта.

**Объект** - это готовая сущность созданная по классу. Иногда объекты называют "**инстансами**". Раньше мы уже создавали и использовали объекты:

```
мой_компьютер = Компьютер()
мой_компьютер.открыть_урок_itvdn()
```

Мы создали объект мой_компьютер, и вызвали у него метод открыть_урок_itvdn. Обратите внимание, что методы мы вызываем на конкретном объекте, а не схеме.

Когда вы вызываете на объекте метод мой_компьютер.открыть_урок_itvdn() на самом деле происходит следующая операция:

```
Компьютер.открыть_урок_itvdn(мой_компьютер)
```

При внимательном исследовании записи, оказывается, что методы вызываются на классе, но в качестве первого аргумента передается уже ранее созданный объект. Это сравнимо, если мы говорим Python'ну: "Используя схему **Компьютер**, открой на **моем компьютере** урок ITVDN". Поэтому в качестве первого аргумента большинство методов принимают объект self, то есть сами себя.

К счастью, нам не надо каждый раз использовать такую длинную запись с классом. Достаточно вызвать метод на самом объекте. Он сам вызовет свой класс и передаст себя в качестве первого аргумента self.

Если вы придумаете революционную схему космической ракеты (class), вы никуда не улетите на ней, пока не реализуете реальную ракету по этой схеме (object). Как только вы создадите реальную ракету по своей схеме, вы сможете вызвать на ней метод "улететь_на_луну".

Реализация в Python

Перепишем наш "Компьютер" на Python'не.



CyberBionic systematics

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

```
class BrowserWindow:
    def open(self, link):
        self.current_link = link
        print(f"Link '{link}' was opened!")

class Computer:
    def go_online(self):
        return BrowserWindow()

    def open_itvdn_lesson(self):
        browser_window = self.go_online()
        browser_window.open("itvdn.com")

        return browser_window

my_own_computer = Computer()
browser = my_own_computer.open_itvdn_lesson()
```

Обратите внимание, что названия классов пишутся с большой буквы. Это соглашения между Python-разработчиками (PEP 8).

## Закрепление материала

- Что такое парадигма программирования?
- Что такое класс?
- Что такое объект?
- Что в Python не является объектом?
- Что такое атрибуты класса?
- Что такое атрибуты экземпляров класса?

## Дополнительное задание

Задание
Создайте класс, описывающий автомобиль. Создайте класс автосалона, содержащий в себе список автомобилей, доступных для продажи, и функцию продажи заданного автомобиля.

## Самостоятельная деятельность учащегося

Задание 1
Создайте класс, описывающий книгу. Он должен содержать информацию об авторе, названии, годе издания и жанре. Создайте несколько разных книг. Определите для него операции проверки на равенство и неравенство, методы __repr__ и __str__.

Задание 2
Создайте класс, описывающий отзыв к книге. Добавьте в класс книги поле – список отзывов. Сделайте так, что при выводе книги на экран при помощи функции print также будут выводиться отзывы к ней.

Задание 3
Ознакомьтесь со специальными методами в Python, используя ссылки в конце урока, и научитесь использовать те из них, назначение которых вы можете понять. Возвращайтесь к этой теме на протяжении всего курса и изучайте специальные методы, соответствующие темам каждого урока.

## Рекомендуемые ресурсы

Документация по Python
https://docs.python.org/3/tutorial/classes.html – ООП в Python
https://docs.python.org/3/reference/datamodel.html#special-method-names – методы со специальными именами



CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Обзор специальных методов в Python
http://rafekettler.com/magicmethods.html

Статьи в Википедии о ключевых понятиях, рассмотренных на этом уроке
https://ru.wikipedia.org/wiki/Объектно-ориентированное_программирование
https://ru.wikipedia.org/wiki/Объектно-ориентированное_программирование_на_Python
https://ru.wikipedia.org/wiki/Класс_(программирование)
https://ru.wikipedia.org/wiki/Объект_(программирование)
https://ru.wikipedia.org/wiki/Свойство_(программирование)
http://www.programiz.com/python-programming/property
https://docs.python.org/3/library/functions.html#property



CyberBionic systematics

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com