# ООП — Класи, атрибути, методи, конструктор

**№ уроку:** 1 **Курс:** Python Essential

**Засоби навчання:** PyCharm

## Огляд, ціль та призначення уроку

Після завершення уроку учні матимуть уявлення про парадигму об'єктно-орієнтованого програмування, зможуть створювати класи та об'єкти у програмах на Python.

## Вивчивши матеріал даного заняття, учень зможе:

- Розуміти основи парадигми об'єктно-орієнтованого програмування
- Розуміти призначення та різницю між класами і об'єктами
- Створювати класи
- Створювати екземпляри класів
- Створювати методи

## Зміст уроку

1. Що таке парадигма програмування?
2. Що таке ООП?
3. Основні принципи ООП.
4. Що таке клас?
5. Що таке спосіб?
6. Що таке об'єкт?
7. Реалізація у Python

## Резюме

Програмування - це сфера, де одне й те саме завдання можна вирішити декількома способами. Ось лише кілька прикладів, як можна знайти суму 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))
```

Усі 3 варіанти виведуть значення "20". Але отримано значення у різний спосіб.



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

Варіант 1 – найпростіший: за допомогою операції "+" склали всі числа.
Варіант 2 використовує вбудовану функцію sum, щоб знайти суму списку чисел.
І, нарешті, варіант 3, що демонструє часту проблему програмістів - винайдення власного колеса. Цей варіант найбільш об'ємний та безглуздий, адже використовує зайві конструкції.

**Порада**
**Не пишіть зайвий код, якщо можна скористатися вбудованою функцією Python.**
**Вміння вирішувати завдання лаконічно – відмінна риса програміста.**
3 приклади — три різні рішення, один результат. Програмісти думали по-різному, коли вирішували однакове завдання – у кожного свій підхід. Саме такий підхід і називається **парадигма**.
**Парадигма програмування** - це сукупність ідей та понять, що визначають стиль написання комп'ютерних програм (**підхід** до програмування). Це спосіб концептуалізації, **що визначає організацію** обчислень та структурування коду.
Якщо до вас потрапляє проект іншого програміста, іноді доводиться витрачати чимало часу, щоб розібратися у його **парадигмі**. А якщо в місяць вам знадобиться працювати з 6 такими проектами? Щоразу звикати до нової парадигми іншого програміста?
На щастя ні. У програмуванні є дві лідируючі парадигми: **об'єктно-орієнтована** та **функціональна парадигми**. Також є безліч інших парадигм, які використовуються програмістами, але суттєво рідше.
Найчастіше, щоб втілити в коді ту чи іншу парадигму, потрібна підтримка з боку синтаксису мови. Наприклад, щоб вирішити завдання у стилі функціонального програмування, мова обов'язково повинна підтримувати створення функцій (**і не тільки**).

**Об'єктно-орієнтована парадигма** — це підхід до програмування, коли все в коді представляється як сутності (об'єкти) та їх методи. Наприклад, ви зараз використовуєте об'єкт комп'ютер: цей об'єкт має інтерфейс для користувача і цей інтерфейс надає доступ до ряду функціоналу (методів). А чи замислювалися ви, що цей комп'ютер хтось вигадав: функція доступу до інтернету, можливість подивитися фільм – усі ці методи були кимось спроектовані. Іншими словами, ваш комп'ютер був створений за планом/ескізом.
Якби ми хотіли написати свій комп'ютер неіснуючою мовою, це виглядало би приблизно так:

```
схема Комп’ютер:
    метод зайти_до_інтернету():
        повернути_вікно_браузера

    метод відкрити_урок_itvdn():
        вікно_браузера = зайти_до_інтернету()
        вікно_браузера.перейти_на("itvdn.com")

        повернути вікно_браузера

    # ...
    # ...
    # ...
    # ... і так інші функції комп’ютера
```

Ми створили із вами схему комп'ютера. Схема — це лише записана ідея, але не реальний комп'ютер. Щоб створити "реальний" комп'ютер і ним скористатися, треба створити об'єкт:
мій_комп'ютер = Комп'ютер()
За допомогою виклику схеми, як звичайної функції Python, ми можемо створювати реальні об'єкти. Але як тепер відкрити урок на ITVDN? Для цього нам необхідно "викликати" потрібний метод **на об'єкті**:
мій_комп’ютер.відкрити_урок_itvdn()
**Важливі аспекти ООП: наслідування, поліморфізм, інкапсуляція, абстракція.** Про них ми поговоримо на наступних заняттях.
Це найбільш базові принципи, які допоможуть вам думати в стилі ОВП.

**Клас** – це схема майбутнього об'єкту. Повертаючись до комп'ютера, запишемо код із класом:

```
class Комп'ютер:
    ІМ'Я_КОМП'ЮТЕРА = "CBS"
    метод зайти_в_інтернет():
```



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

```
        повернути вікно_баузера

    метод відкрити_урок_itvdn():
        вікно_браузера = зайти_в_інтернет()
        вікно_браузера.перейти_на("itvdn.com")

        повернути вікно_баузера
```

Ми замінили слово "схема" на конструкцію class, що надає нам Python. У класі ми можемо описати методи об'єкта та поля даних.

**Що таке метод?**

Ми вже побачили, що в схемах (class) ми **описуємо**, який функціонал повинен мати наш майбутній об'єкт за допомогою методів. У Python це робиться майже так, як у нашому псевдокоді. Кожен метод класу оголошується за допомогою ключового слова def як функція:

```
class Комп'ютер:
    ІМ'Я_КОМП'ЮТЕРА = "CBS"

def зайти_в_інтернет(self):
        return вікно_баузера

def відкрити_урок_itvdn(self):
        вікно_браузера = зайти_в_інтернет()
        вікно_браузера.перейти_на("itvdn.com")

        return вікно_баузера
```

Що змінилося:

- Ми змінили слово «метод» на конструкцію Python def.
- Слово "повернути" змінили на знайоме нам return
- Кожен метод-функція приймає як перший аргумент self (про це трохи пізніше)

**Поля даних** - це змінні, оголошені всередині класу. Ці змінні одержують усі його об'єкти. Іноді поля даних називають властивостями об'єкта.

**Об'єкт** - це готова сутність, створена по класу. Іноді об'єкти називають "**інстансами**". Раніше ми вже створювали та використовували об'єкти:

```
мій_комп'ютер = Комп'ютер()
мій_комп'ютер.відкрити_урок_itvdn()
```

Ми створили об'єкт мій_комп'ютер, і викликали у нього метод відкрити_урок_itvdn. Зауважте, що методи ми викликаємо на конкретному об'єкті, а не схемі.

Коли ви викликаєте на об'єкті метод мій_комп'ютер.відкрити_урок_itvdn(), насправді відбувається наступна операція:

Комп'ютер.відкрити_урок_itvdn(мій_комп'ютер)

При уважному дослідженні запису виявляється, що методи викликаються на класі, але як перший аргумент передається вже раніше створений об'єкт. Це так, неначе ми говоримо Python'ну: "Використовуючи схему **Комп'ютер**, відкрий на **моєму комп'ютері** урок ITVDN". Тому як перший аргумент більшість методів приймають об'єкт self, тобто самі себе.

На щастя, нам не треба щоразу використовувати такий довгий запис із класом. Досить викликати метод на самому об'єкті. Він сам викличе свій клас і передасть себе в якості першого аргументу self.

Якщо ви придумаєте революційну схему космічної ракети (class), ви нікуди не полетите на ній, доки не реалізуєте реальну ракету за цією схемою (object). Як тільки ви створите реальну ракету за своєю схемою, ви зможете викликати на ній метод "полетіти_на_місяць".

Реалізація у Python

Перепишемо наш "Комп'ютер" мовою Python.



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()
```

Зверніть увагу, що назви класів пишуться з великої літери. Це угоди між Python-розробниками (PEP 8).

## Закріплення матеріалу

- Що таке парадигма програмування?
- Що таке клас?
- Що таке об'єкт?
- Що в Python не є об'єктом?
- Що таке атрибути класу?
- Що таке атрибути екземплярів класу?

## Додаткове завдання

Завдання
Створіть клас, який описує автомобіль. Створіть клас автосалону, що містить в собі список автомобілів, доступних для продажу, і функцію продажу заданого автомобіля.

## Самостійна діяльність учня

Завдання 1
Створіть клас, який описує книгу. Він повинен містити інформацію про автора, назву, рік видання та жанр. Створіть кілька різних книжок. Визначте для нього операції перевірки на рівність та нерівність, методи _repr_ та _str_.

Завдання 2
Створіть клас, який описує відгук до книги. Додайте до класу книги поле – список відгуків. Зробіть так, щоб при виведенні книги на екран за допомогою функції print також виводилися відгуки до неї.

Завдання 3
Ознайомтеся зі спеціальними методами в Python, використовуючи посилання в кінці уроку, і навчіться використовувати ті з них, призначення яких ви можете зрозуміти. Повертайтеся до цієї теми протягом усього курсу та вивчайте спеціальні методи, що відповідають темам кожного уроку.

## Рекомендовані ресурси

Документація з Python
https://docs.python.org/3/tutorial/classes.html – ООП в Python
https://docs.python.org/3/reference/datamodel.html#special-method-names – методи зі спеціальними іменами



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

Огляд спеціальних методів у Python
https://uk.wikibooks.org/wiki/Пориньте_у_Python_3/Імена_магічних_методів

Статті у Вікіпедії про ключові поняття, розглянуті на цьому уроці
https://uk.wikipedia.org/wiki/Об'єктно-орієнтоване_програмування
https://uk.wikipedia.org/wiki/Клас_(програмування)
https://uk.wikipedia.org/wiki/Об'єкт_(програмування)
https://uk.wikipedia.org/wiki/Властивість_(програмування)
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