Реферат

Тема: Робота з класами та об'єктами

В [об'єктно-орієнтованому програмуванні](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F" \o "Об'єктно-орієнтоване програмування), клас — це спеціальна конструкція, яка використовується для групування пов'язаних змінних та [функцій](https://uk.wikipedia.org/wiki/%D0%9F%D1%96%D0%B4%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0" \o "Підпрограма). При цьому, згідно з термінологією [ООП](https://uk.wikipedia.org/wiki/%D0%9E%D0%9E%D0%9F" \o "ООП), глобальні змінні класу (члени-змінні) називаються полями даних (також властивостями або атрибутами), а члени-функції називають [методами](https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)" \o "Метод (програмування)) класу. Створений та ініціалізований екземпляр класу називають об'єктом класу. На основі одного класу можна створити безліч об'єктів, що відрізнятимуться один від одного своїм станом (значеннями полів).

Клас можна порівнювати з формою для випічки печива — форма одна, а печива можна випекти безліч. Печиво — це конкретні об'єкти, екземпляри класу печиво, яке може бути з різною начинкою. Поля дозволяють вмістити дані про певний реальний об'єкт, а методи здійснювати обробку цих даних. Наприклад, можна створити загальний клас Людина з полями Ім'я та Прізвище, рік народження, професія, зарплата. При створенні ж на основі класу конкретного екземпляру, дані поля заповнюються конкретними даними про певну людину. Обробкою цих даних можуть займатися відповідні методи. Наприклад, можна створити метод для обчислення віку людини, тощо.

На основі класів можна створювати підкласи, які [успадковують](https://uk.wikipedia.org/wiki/%D0%A3%D1%81%D0%BF%D0%B0%D0%B4%D0%BA%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)" \o "Успадкування (програмування)) властивості та поведінку батьківських класів. Таким чином можна створити цілу ієрархію класів. Різні мови дещо по різному реалізовують механізм успадкування. Існує множинне та одинарне успадкування. Множинне — це, коли підклас створюється на основі кількох безпосередніх батьків (як то в мові програмування [C++](https://uk.wikipedia.org/wiki/C%2B%2B" \o "C++)). Одинарне успадкування — це коли клас може мати одного безпосереднього батька (мова програмування [Java](https://uk.wikipedia.org/wiki/Java" \o "Java)). Надкласи можуть мати свої надкласи, підкласи можуть також бути надкласами для певних класів.

Python підтримує об'єктно-орієнтовану парадигму програмування, це означає, що ми можемо визначити компоненти програми у вигляді класів.

Клас є шаблоном чи формальним описом об'єкта, а об'єкт представляє екземпляр цього, його реальне втілення. Можна провести наступну аналогію: у всіх у нас є деяке уявлення про людину – наявність двох рук, двох ніг, голови, травної, нервової системи, головного мозку тощо. Є певний шаблон – цей шаблон можна назвати класом. Реально існуюча людина (фактично екземпляр даного класу) є об'єктом цього класу.

З погляду коду клас поєднує набір функцій та змінних, які виконують певне завдання. Функції класу називають методами. Вони визначають поведінку класу. А змінні класу називають атрибутами – вони зберігають стан класу

Клас визначається за допомогою ключового слова class:

class подкласс (суперкласс):

методы\_подкласса

При визначенні методів будь-якого класу слід враховувати, що всі вони повинні приймати як перший параметр посилання на поточний об'єкт, який відповідно до умов називається self (у ряді мов програмування є свого роду аналог - ключове слово this). Через це посилання всередині класу ми можемо звернутися до методів або атрибутів цього класу. Зокрема, через вираз self.name можна отримати ім'я користувача.

class Person:

name = "Tom"

def display\_info(self):

print("Привет, меня зовут", self.name)

person1 = Person()

person1.display\_info()         # Привет, меня зовут Tom

person2 = Person()

person2.name = "Sam"

person2.display\_info()         # Привет, меня зовут Sam

Клас Person визначає атрибут name, який зберігає ім'я людини, та метод display\_info, за допомогою якого виводиться інформація про людину.

При визначенні методів будь-якого класу слід враховувати, що всі вони повинні приймати як перший параметр посилання на поточний об'єкт, який відповідно до умов називається self (у ряді мов програмування є свого роду аналог - ключове слово this). Через це посилання всередині класу ми можемо звернутися до методів або атрибутів цього класу. Зокрема, через вираз self.name можна отримати ім'я користувача.

Після визначення клас Person створюємо пару його об'єктів - person1 та person2. Використовуючи ім'я об'єкта, ми можемо звернутися до його методів та атрибутів. В даному випадку у кожного з об'єктів викликаємо метод display\_info(), який виводить рядок на консоль, і другий об'єкт також змінює атрибут name. При цьому під час виклику методу display\_info не потрібно передавати значення для параметра self.

Для створення об'єкта класу використається конструктор. Так, коли ми створювали об'єкти класу Person, ми використовували конструктор за замовчуванням, який неявно мають усі класи:

person1 = Person()

person1.display\_info()

Однак ми можемо явно визначити в класах конструктор за допомогою спеціального методу, який називається \_\_init\_\_(). Наприклад, змінимо клас Person, додавши до нього конструктор:

class Person:

# конструктор

def \_\_init\_\_(self, name):

self.name = name  # устанавливаем имя

def display\_info(self):

print("Привет, меня зовут", self.name)

person1 = Person("Tom")

person1.display\_info()         # Привет, меня зовут Tom

person2 = Person("Sam")

person2.display\_info()         # Привет, меня зовут Sam

Як перший параметр конструктор також приймає посилання на поточний об'єкт - self. Нерідко у конструкторах встановлюються атрибути класу. Так, в даному випадку як другий параметр в конструктор передається ім'я користувача, яке встановлюється для атрибута self.name. Причому для атрибута необов'язково визначати у класі змінну name, як це було в попередній версії класу Person. Встановлення значення self.name=name вже неявно створює атрибут name.

Після закінчення роботи з об'єктом ми можемо використовувати оператор del для видалення його з пам'яті

Варто зазначити, що в принципі це необов'язково робити, оскільки після закінчення роботи скрипта всі об'єкти автоматично видаляються з пам'яті.

Крім того, ми можемо визначити в класі деструктор, реалізувавши вбудовану функцію \_\_del\_\_, який буде викликатися або в результаті виклику оператора del, або при автоматичному видаленні об'єкта.

1. Що таке клас?
2. Навіщо потрібні класи?
3. Що поєднує клас?
4. Як розшифрувати ООП?
5. Як створити клас?
6. Яке ключове слово використовується для створення класу?
7. Чи можливо створити клас без ключового слова class?
8. Чи можна використати class для створення класу?
9. Чи використовуються класи у Пайтоні?
10. Що таке конструктор класу?
11. Що таке деструктор класу?
12. Як реалізувати деструктор?
13. Як реалізувати конструктор?
14. Чи можна програмувати без класів?
15. Чи можна отримати доступ до прайвет властивості?
16. Чи можна створити у класах прайвет властивості?
17. Як у класі отримати властивості екземляру класу?
18. Який аналог є до слова this?
19. Як можна унаслідувати клас від іншого класу?
20. Від якого класу наслідуються інші класи?
21. Чи є класи, від якого наслідуються усі класи?
22. Скільки класів наслідуються завжди за замовчуванням?
23. Чому особливий клас object?
24. Чи є обмеження на кількість методів у класах?
25. Чи є обмеження на кількість властивостей у класі?

Джерела:

<https://metanit.com/python/tutorial/7.1.php>

https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81\_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)