РЕФЕРАТ

Тема: Клас object

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).

**Успадкування** (англ. *[inheritance](https://en.wikipedia.org/wiki/Inheritance_(object-oriented_programming)" \l ":~:text=Inheritance_allows_programmers_to_create,via_public_classes_and_interfaces." \o "en:Inheritance (object-oriented programming))*) — це один з принципів [об'єктно-орієнтовного програмування](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)[[1]](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)#cite_note-1), який дає [класу](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)" \o "Клас (програмування)) можливість використовувати програмний код іншого (базового) класу[[2]](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)" \l "cite_note-2), доповнюючи його своїми власними деталями реалізації. Іншими словами, під час успадкування відбувається отримання нового (похідного) класу, який містить програмний код базового класу з зазначенням власних особливостей використання. Успадкування належить до типу [is-a](https://uk.wikipedia.org/wiki/Is-a" \o "Is-a) відношень між класами. При успадкуванні створюється спеціалізована версія вже існуючого класа.

Починаючи з 3-ї версії Python, всі класи неявно мають один загальний суперклас - object і всі класи за замовчуванням успадковують його методи.

Однією з найбільш використовуваних методів класу object є метод \_\_str\_\_(). Коли необхідно отримати рядкове представлення об'єкта або вивести об'єкт у вигляді рядка, то Python якраз викликає цей метод. І щодо класу хорошої практикою вважається перевизначення цього.

Наприклад, візьмемо клас Person і виведемо його рядкове представлення:

class Person:

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

self.\_\_name = name

self.\_\_age = age

@property

def name(self):

return self.\_\_name

@property

def age(self):

return self.\_\_age

@age.setter

def age(self, age):

if age in range(1, 100):

self.\_\_age = age

else:

print("Недопустимый возраст")

def display\_info(self):

print("Имя:", self.\_\_name, "\tВозраст:", self.\_\_age)

tom = Person("Tom", 23)

print(tom)

При запуску програма виведе щось на кшталт наступного:

<\_\_main\_\_.Person object at 0x0000017D2BEBDCF8>

Це не надто інформативна інформація про об'єкт. Тепер визначимо у класі Person метод \_\_str\_\_:

class Person:

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

self.\_\_name = name # устанавливаем имя

self.\_\_age = age # устанавливаем возраст

@property

def name(self):

return self.\_\_name

@property

def age(self):

return self.\_\_age

@age.setter

def age(self, age):

if age in range(1, 100):

self.\_\_age = age

else:

print("Недопустимый возраст")

def display\_info(self):

print(self.\_\_str\_\_())

def \_\_str\_\_(self):

return "Имя: {} \t Возраст: {}".format(self.\_\_name, self.\_\_age)

tom = Person("Tom", 23)

print(tom)

Метод \_\_str\_\_() повинен повертати рядок. І в цьому випадку ми повертаємо базову інформацію про людину. І тепер консольний вивід буде іншим:

Ім'я: Tom Вік: 23

1. Який клас за замовчуванням наслідують всі об’єкти в Пайтоні?
2. Чи можна перевизначати методи об’єкта object?
3. Чи можна викликати методи об’єкта object?
4. Який метод базового класу викликається у випадку необхідності строкового представлення цього об’єкта?
5. Чому клас object використовують при темі наслідування?
6. Чому клас, від якого наслідуються інші класи називається object?
7. Як наслідуватись від класу object?
8. Чи можна не наслідуватись від класу object?
9. Чи можна зовсім не наслідуватись від жодного класу?
10. Що таке клас?
11. Чи можна унаслідуватись від класу, який унаслідуваний від object?
12. Чим особливий object?
13. Яку проблему вирішує наслідування?
14. Як використати клас object?
15. Чи підтримується множинне спадкування у Пайтоні?
16. Чи потрібно передавати у базовий клас object параметри?
17. Чи потрібно викликати явно конструктор класу object?
18. Чи потрібно імпортувати щось для використання object?
19. Чи існує наслідування усіх класів від object у старих версіях Пайтону?
20. Чи можна створити свій клас object?
21. Чи існує ще якийсь клас, окрім object, від якого наслідуються усі інші?
22. Який импорт потрібно використати для роботи з класом object?
23. Чому існує такий клас, від якого наслідуються усі інші класи неявно?
24. Чи явно успадковуються класи від єдиного класу object?
25. Чим являється клас object для всіх класів у Пайтоні?
26. З якої версії Пайтона усі класи неявно успадковуються від object?

Джерела:

<https://metanit.com/python/tutorial>  
  
<https://www.google.com/search?q=%D0%BA%D0%BB%D0%B0%D1%81+object+%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+python&sxsrf=AOaemvJddrTjLlmUUfmaEggpQm5nsIJYFQ%3A1639653569233&ei=wSC7YcLFDYKxrgTOlL7QBg&oq=%D0%BA%D0%BB%D0%B0%D1%81+object%C2%A0+%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+pyt&gs_lcp=Cgdnd3Mtd2l6EAMYADIECCEQFToECAAQR0oECEEYAEoECEYYAFDaA1i_HmCsKGgDcAN4AIABmgKIAZIKkgEFMC41LjKYAQCgAQHIAQjAAQE&sclient=gws-wiz>

<https://www.wiki.uk-ua.nina.az/%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).html>

https://python-scripts.com/python-class