

ІНФОРМАТИКА

8

Поняття об'єкта в мові програмування, його властивостей і методів. Структура програми

За навчальною програмою 2017 року



Урок 38

teach-inf.com.ua

Мова **Python** є об'єктно-орієнтованою, тобто кожна величина є об'єктом певного класу. Окрім вбудованих класів (типів даних), програміст може описувати і використовувати в програмі власні класи.

Добре спроектовані класи — це «будівельні блоки», з яких легше будувати складні програми.

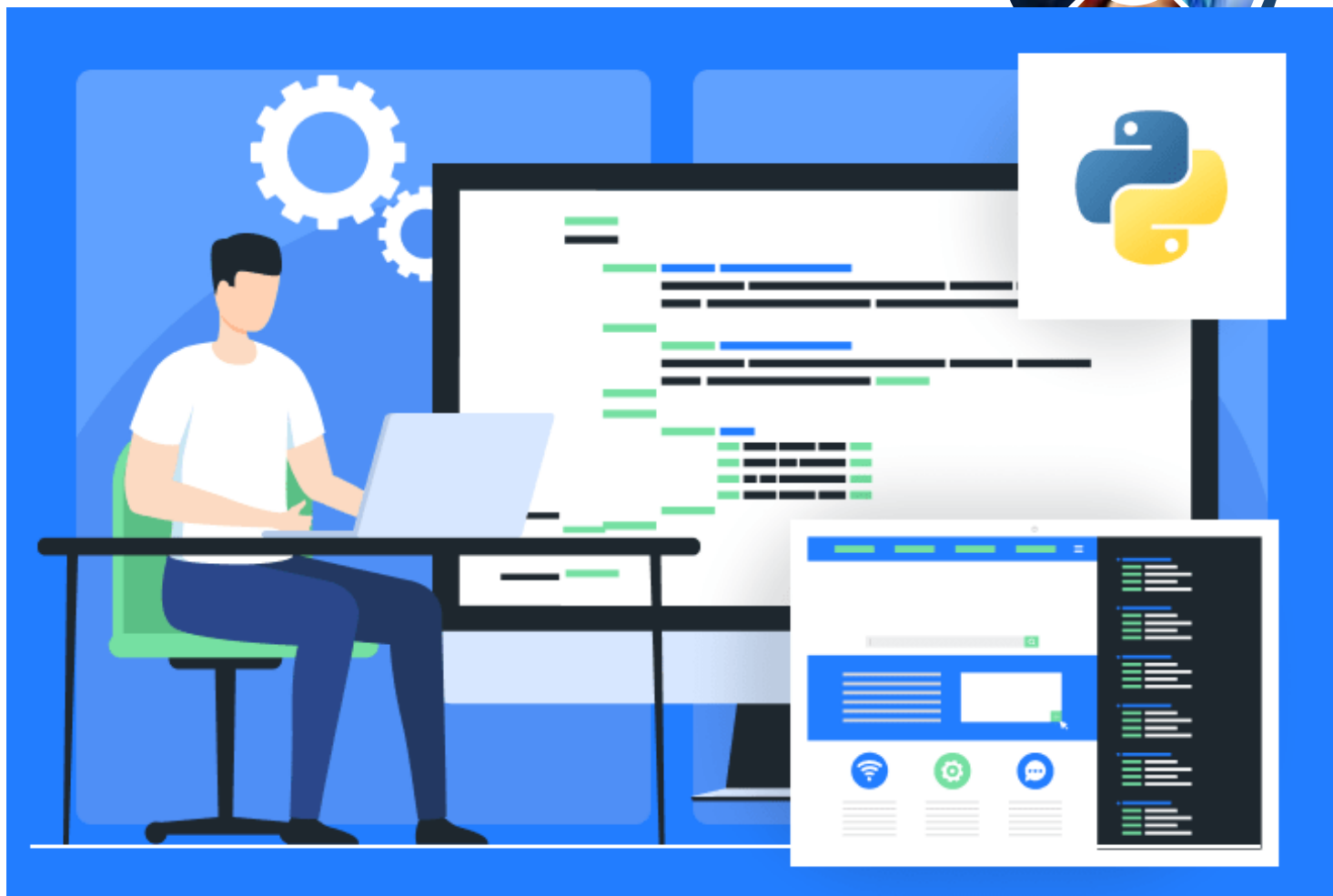


Клас — це опис об'єктів певного типу, **об'єкт** — це екземпляр деякого класу. Опис класу містить **атрибути (змінні)**, які відповідають властивостям об'єктів, і описи **методів класу** — дій, які можуть виконувати об'єкти цього класу.



Якщо опис класу **Animal** (Тварина) містить атрибут маса, то кожному екземпляру цього класу можна надати певне значення маси.

*Згадаємо, як описують класи в програмі. Можна вважати, що **клас** — це своєрідна інструкція зі створення екземплярів. Об'єкт, створений на основі класу, називають екземпляром цього класу.*



Опис класу зручно розташовувати на початку коду програми.

Синтаксис опису класу:

```
class <назва класу>(<базовий клас>):  
    <атрибут класу> = <значення>  
    def __init__(self, <інші параметри>):  
        self.<атрибут екземпляра> = значення
```

Атрибути класу мають однакове значення для всіх екземплярів класу, тоді як атрибути екземплярів — окреме значення для кожного екземпляра.

Метод `__init__` називається **конструктором класу**. За наявності, він автоматично виконується під час створення кожного нового екземпляра класу для початкового налаштування властивостей об'єкта.

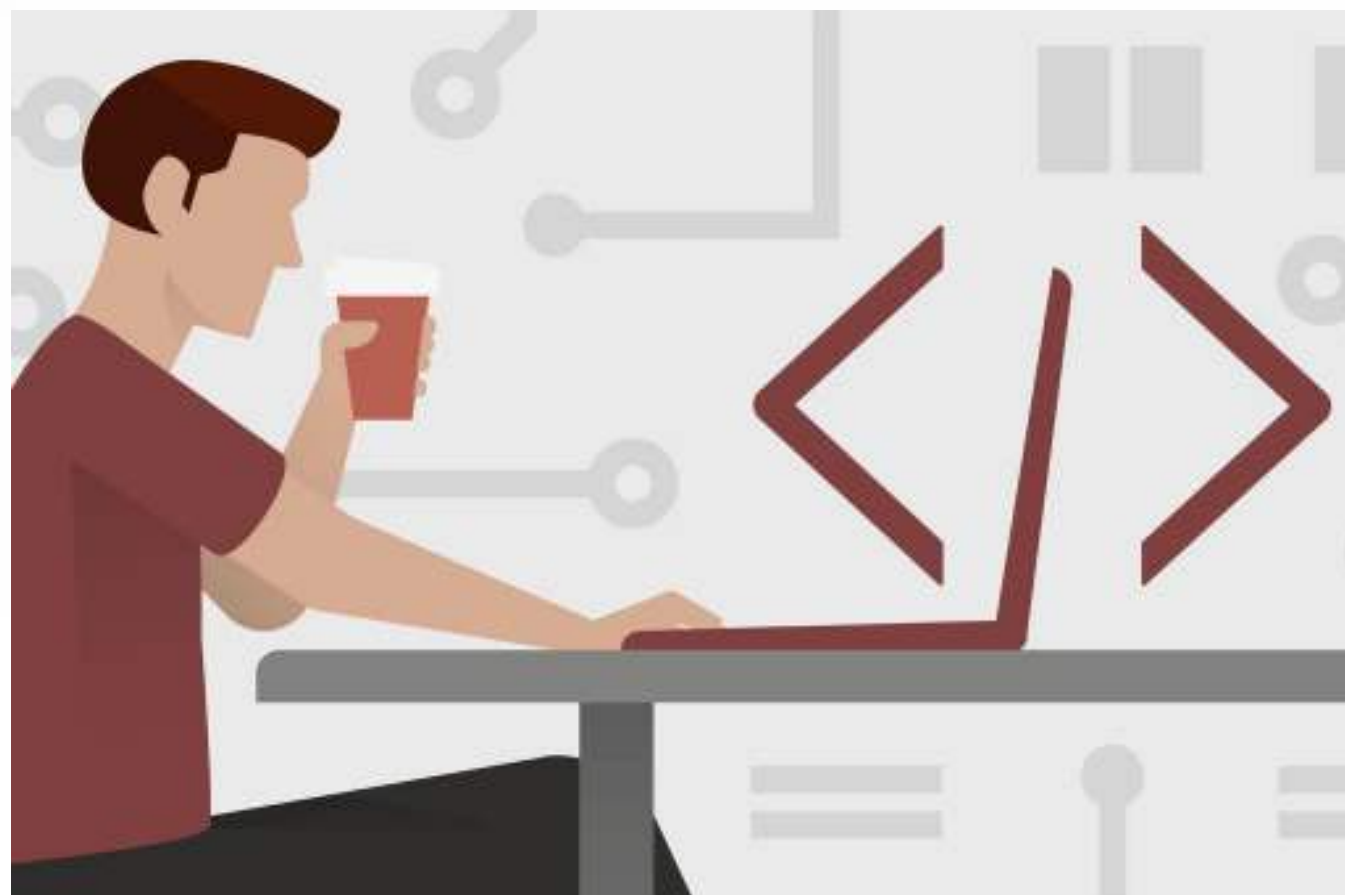
Перший із параметрів в описі конструктора (здебільшого його позначають **`self`**) зберігатиме посилання на створений об'єкт.



Синтаксис команди створення об'єкта:

[змінна =] <назва класу>([<перелік значень параметрів>])

Доступ до створеного об'єкта можна отримати через змінну. Якщо такий доступ не потрібен, назву змінної не вказують.





Опишемо клас, що моделює транспортний засіб:

```
class Transport():  
    def __init__(self, type_vehicle, motor):  
        self.type_vehicle = type_vehicle    # Тип транспортного засобу  
        self.motor = motor                  # Тип двигуна  
tr1 = Transport('вантажівка' , 'дизельний') # Створення екземпляра класу
```

В останньому рядку коду викликається конструктор класу *Transport* і створюється екземпляр класу, відповідні атрибути якого мають значення вантажівка і дизельний. Посилання на цей об'єкт зберігається в змінну *tr1*.

Під час виклику методів можуть змінюватися властивості (значення атрибутів) об'єкта, а також виконуватися інші дії.

Синтаксис заголовка методу класу:

`def <назва методу>(self[, параметри]):`

Код методу виконується у відповідь на виклик методу для конкретного об'єкта.

Виклик методу для об'єкта має такий **синтаксис**:

<об'єкт>.<метод>([значення параметрів])

*У разі виклику методу до нього передається посилання на той об'єкт, для якого викликається метод (параметр **self**).*

Self in Python



```
class Person:
    # name made in constructor
    def __init__(self, n):
        self.name = n;

    def get_person_name(self):
        return self.name
```

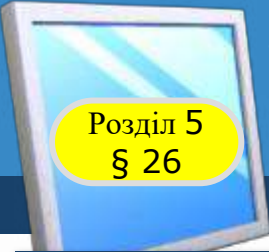


Додамо до опису класу *Transport()* опис методу *fuel()* для виведення повідомлення про вид пального:

```
class Transport():  
    <...>  
    def fuel(self):                # Визначення виду пального  
        print(self.type_vehicle, ':', self.motor, 'двигун')  
tr1 = Transport('вантажівка', 'дизельний')  
tr1.fuel()
```

Буде надруковано:

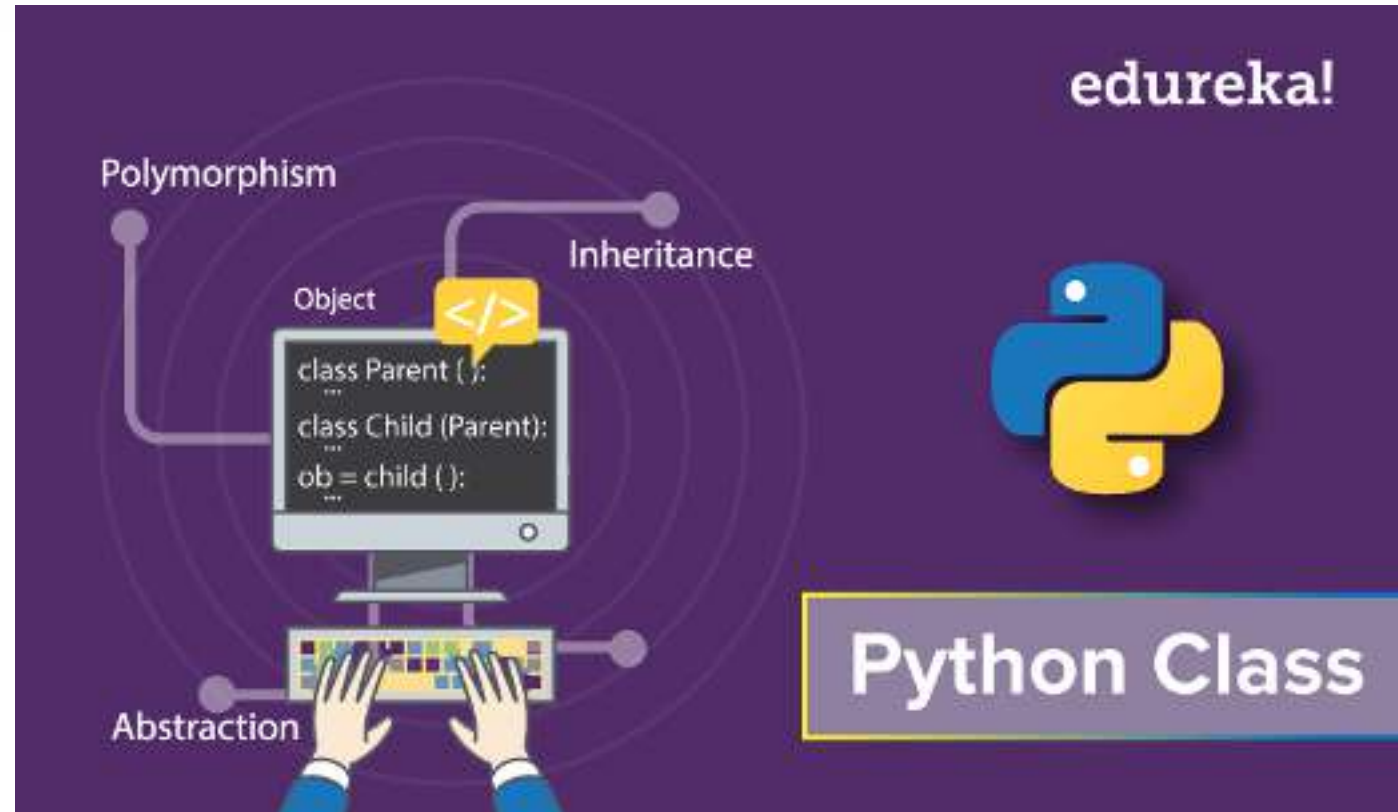
вантажівка: дизельний двигун.



Об'єктно-орієнтоване програмування дозволяє прискорити розробку програми шляхом створення нових класів на основі створених раніше.




Успадкування — це можливість створення класу-нащадка на основі наявного (базового) класу.



Створення класів-нащадків базового класу

8

*Транспортний засіб має такі властивості: тип засобу, тип двигуна, вартість. Автобус, крім цих властивостей, має ще такі: кількість місць, призначення, маршрут. З точки зору успадкування клас **Автобус** є нащадком класу **Транспортний засіб**.*



Клас-нащадок містить усі атрибути та методи базового класу, проте його можна розширити, додавши нові. У класі-нащадку можна не описувати атрибути та методи, успадковані від базового класу.

Створимо на основі базового класу **Transport** два класи-нащадки: **Truck** і **Bus**, що мають власні методи.

```
7 class Truck(Transport):
8     def cargo(self, crg):
9         print(self.type_vehicle, ' перевозить', crg)
10 class Bus(Transport):
11     def passenger(self, destination):
12         print(self.type_vehicle, ' везе учнів', destination)
13 tr1 = Truck('вантажівка', 'дизельний')
14 tr2 = Truck('автоцистерна', 'бензиновий')
15 tr1.cargo('гравій')
16 tr2.cargo('воду')
17 tr3 = Bus('міжміський', 'дизельний')
18 tr3.passenger('на екскурсію')
```

У панелі **Структура** відображається структура опису класів і перелік змінних екземплярів класів.

Зверніть увагу на позначки заголовків:

c

класів

m

методів

f

атрибутів

v

екземплярів класів

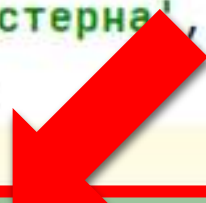
Structure



```
▼ c Transport
  m __init__(self, type_vehicle, motor)
  m fuel(self)
  f motor
  f type_vehicle
▼ c Truck(Transport)
  m cargo(self, crg)
▼ c Bus(Transport)
  m passenger(self, destination)
v tr1
v tr2
v tr3
```

Якщо набрати назву екземпляра класу, автодоповнення запропонує перелік методів, які можна викликати для цього об'єкта із зазначенням класу, в якому метод описано.

```
tr2 = Truck('автоцистерна', 'бензиновий')  
tr1.cargo('гравій')  
tr1.
```



m	cargo(self, crg)	Truck
f	type_vehicle	Transport
f	motor	Transport
m	fuel(self)	Transport
m	__init__(self, type_vehicle, motor)	Transport
f	__annotations__	object
p	__class__	object

Отже, програміст може описати свій тип даних (клас), визначити в класі певні методи.

Разом із тим використання готових класів, описаних у файлах модулів (наприклад, у модулі **tkinter або **easygui**) значно прискорює розробку програми.**





1. Поясніть поняття класу, об'єкта, атрибута класу, методу класу.

2. Поясніть сутність успадкування.

3. Створіть клас *Element* — модель хімічного елемента. У конструкторі класу `__init__` опишіть атрибути *name*, *symbol* і *number*. Створіть екземпляр *elem* класу *Element* зі значеннями атрибутів 'Aurum', 'Au', 79. Виведіть значення атрибутів об'єкта *elem*.



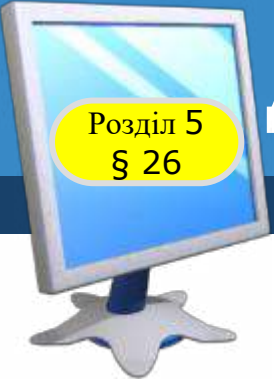


4. Створіть модель країни — клас `Country` з атрибутами `name` (назва країни), `currency` (національна валюта). Опишіть метод класу `print_currency`, який виводить значення атрибутів об'єкта.

5. Створіть екземпляр `my_country` класу `Country` зі значеннями атрибутів 'Україна', 'гривня'. Викличте для об'єкта `my_country` метод `print_currency`.

6. Створіть екземпляри класу `Country`, що описують Польщу, Німеччину, Мексику, Австралію, Танзанію.





Домашнє завдання



**Прийти тестування за посиланням до
10.04**

<https://naurok.com.ua/test/join?gamecode=6861763>

ІНФОРМАТИКА

Дякую за увагу!

8

За навчальною програмою 2017 року



Урок 38

teach-inf.com.ua