



# Урок 14

## Робота з об'єктами

Ініціалізація об'єктів, успадковані функції



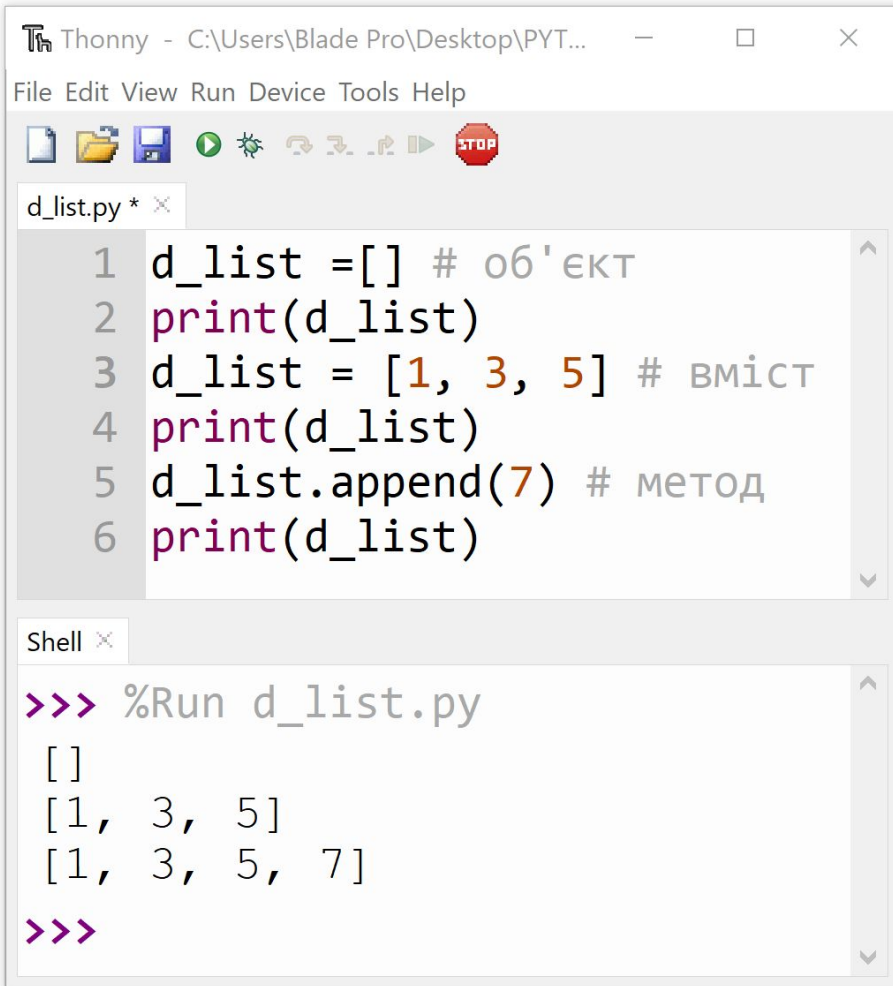
## Об'єкт

це **контейнер**, який має певні характеристики (дані або стан) та виконує певні дії (поведінка).

Приклад:

список **d\_list**, який містить дані - набір чисел **1, 3, 5** та може виконувати певні дії за допомогою методів, наприклад **.append**

Різним об'єктам властива різна поведінка (різні методи).



The screenshot shows the Thonny Python IDE interface. The title bar reads "Thonny - C:\Users\Blade Pro\Desktop\PYT...". The menu bar includes "File", "Edit", "View", "Run", "Device", "Tools", and "Help". The toolbar contains icons for file operations and execution. The editor window, titled "d\_list.py \*", contains the following Python code:

```
1 d_list = [] # об'єкт
2 print(d_list)
3 d_list = [1, 3, 5] # вміст
4 print(d_list)
5 d_list.append(7) # метод
6 print(d_list)
```

Below the editor is a "Shell" window showing the execution output:

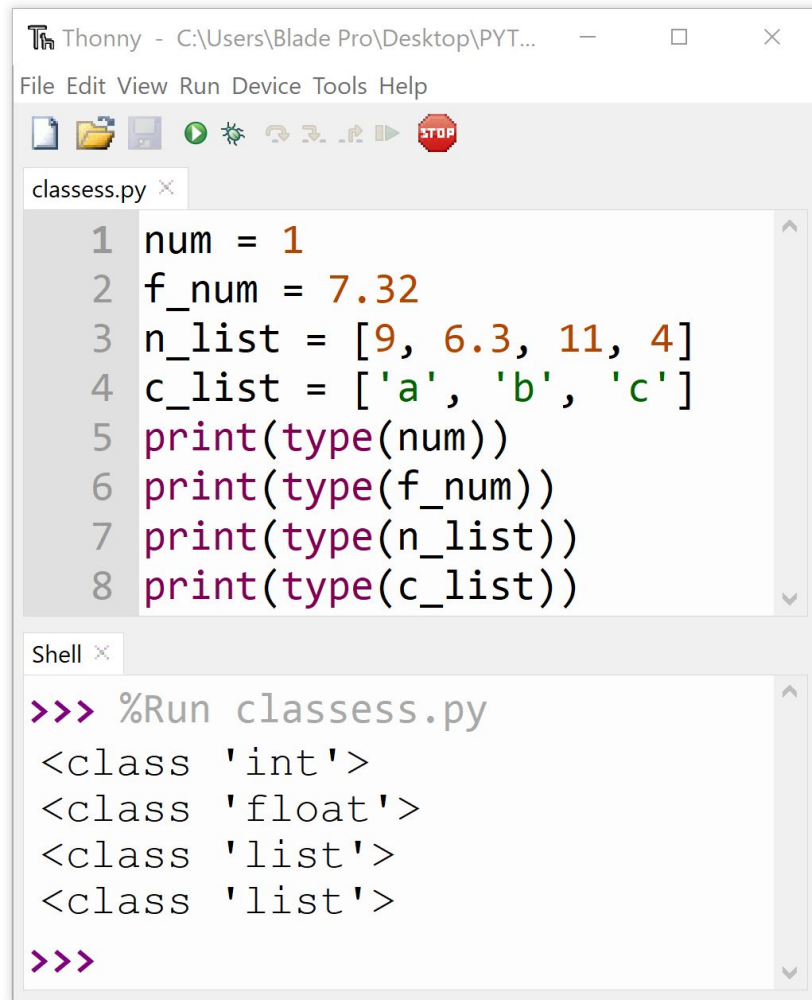
```
>>> %Run d_list.py
[]
[1, 3, 5]
[1, 3, 5, 7]
>>>
```

# Клас

це спосіб групування (класифікація) або шаблон певних характеристик об'єкта.

Дізнатись клас об'єкта можна за допомогою функції **type()**.

Клас можна порівняти з видами тварин: птахи (горобець, орел), ссавці (кішка, жирафа), а представників конкретних видів - з об'єктами.



The screenshot shows the Thonny Python IDE interface. The title bar reads "Thonny - C:\Users\Blade Pro\Desktop\PYT...". The menu bar includes "File", "Edit", "View", "Run", "Device", "Tools", and "Help". The toolbar contains icons for file operations, running, and debugging. The editor window, titled "classess.py", contains the following Python code:

```
1 num = 1
2 f_num = 7.32
3 n_list = [9, 6.3, 11, 4]
4 c_list = ['a', 'b', 'c']
5 print(type(num))
6 print(type(f_num))
7 print(type(n_list))
8 print(type(c_list))
```

The Shell window at the bottom shows the execution output for the command `>>> %Run classess.py`:

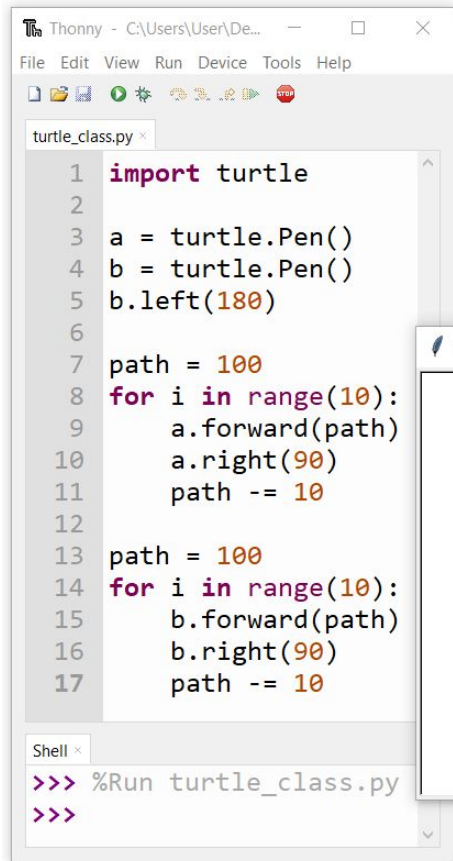
```
<class 'int'>
<class 'float'>
<class 'list'>
<class 'list'>

>>>
```

# Користування класами у Python

Класи допомагають працювати з кількома об'єктами одного класу одночасно. Це спрощує програму та зменшує кількість рядків коду.

Наприклад, можна створити кілька "черепашок" при використанні модулю **turtle**.

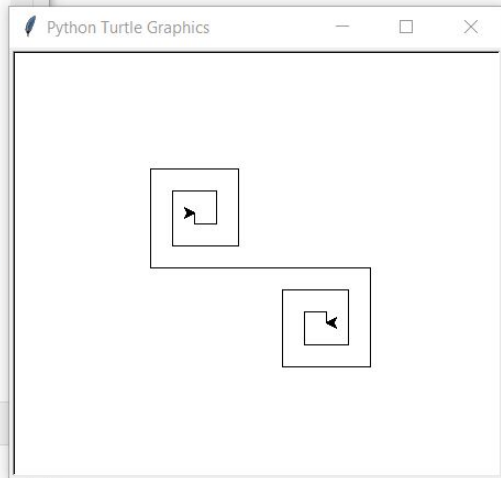


The screenshot shows the Thonny Python IDE interface. The main editor window displays a Python script named `turtle_class.py` with the following code:

```
1 import turtle
2
3 a = turtle.Pen()
4 b = turtle.Pen()
5 b.left(180)
6
7 path = 100
8 for i in range(10):
9     a.forward(path)
10    a.right(90)
11    path -= 10
12
13 path = 100
14 for i in range(10):
15     b.forward(path)
16     b.right(90)
17     path -= 10
```

Below the editor is a Shell window showing the command to run the script:

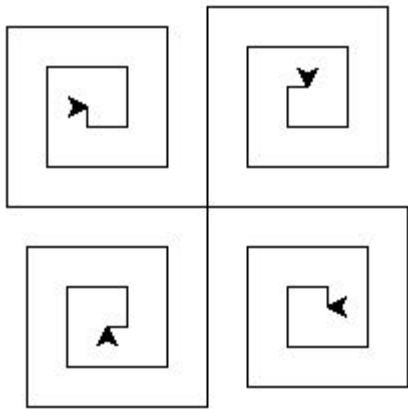
```
>>> %Run turtle_class.py
>>>
```

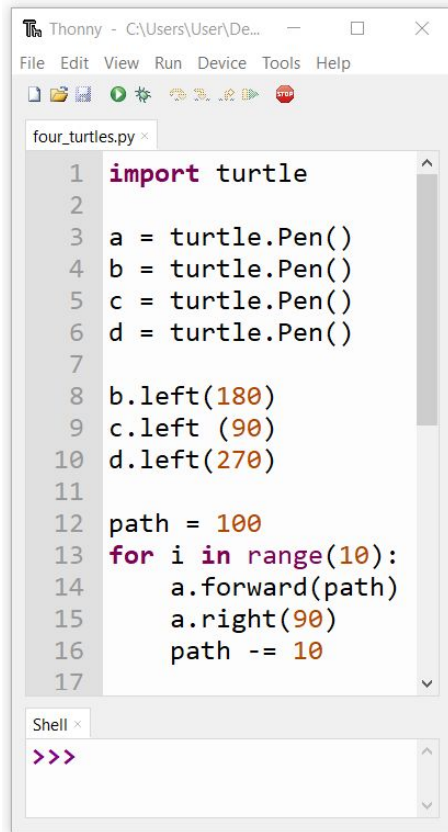




## Практична робота

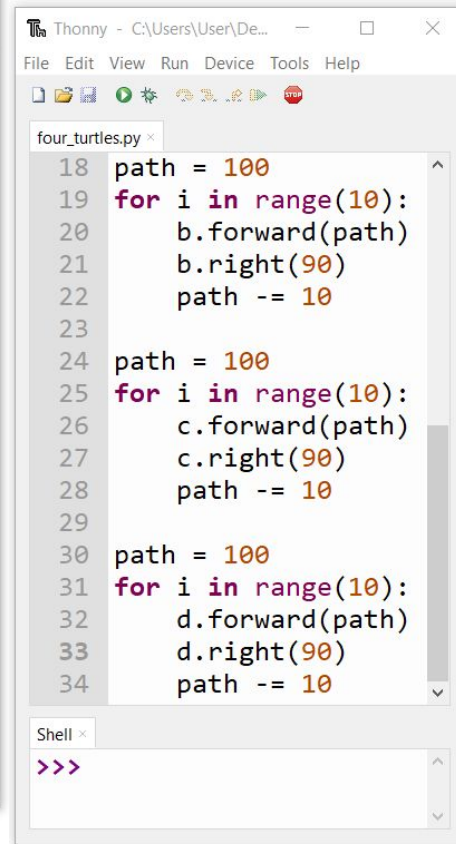
Спробуйте намалювати таку фігуру за допомогою чотирьох “черепашок”.





```
1 import turtle
2
3 a = turtle.Pen()
4 b = turtle.Pen()
5 c = turtle.Pen()
6 d = turtle.Pen()
7
8 b.left(180)
9 c.left(90)
10 d.left(270)
11
12 path = 100
13 for i in range(10):
14     a.forward(path)
15     a.right(90)
16     path -= 10
17
```

Shell >>>



```
18 path = 100
19 for i in range(10):
20     b.forward(path)
21     b.right(90)
22     path -= 10
23
24 path = 100
25 for i in range(10):
26     c.forward(path)
27     c.right(90)
28     path -= 10
29
30 path = 100
31 for i in range(10):
32     d.forward(path)
33     d.right(90)
34     path -= 10
```

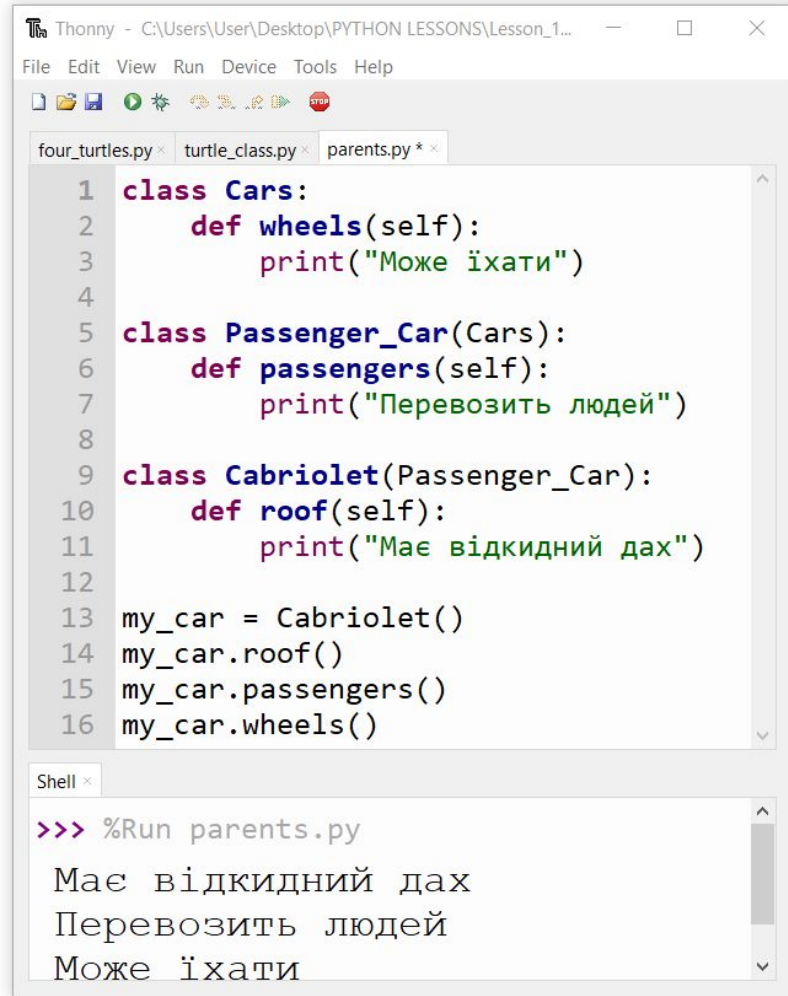
Shell >>>

Якщо все зроблено правильно - ви побачите такий результат

# Успадковані функції

До об'єкта (екземпляра класу) можна застосовувати не тільки ті функції, що визначені в його класі, але й функції батьківських класів.

Наприклад, у об'єкта **my\_car** класу **Cabriolet** можна викликати функцію **wheels()** класу **Cars**, оскільки вона є успадкованою.



The screenshot shows the Thonny Python IDE interface. The main editor window displays a Python script named `parents.py` with the following code:

```
1 class Cars:
2     def wheels(self):
3         print("Може їхати")
4
5 class Passenger_Car(Cars):
6     def passengers(self):
7         print("Перевозить людей")
8
9 class Cabriolet(Passenger_Car):
10    def roof(self):
11        print("Має відкидний дах")
12
13 my_car = Cabriolet()
14 my_car.roof()
15 my_car.passengers()
16 my_car.wheels()
```

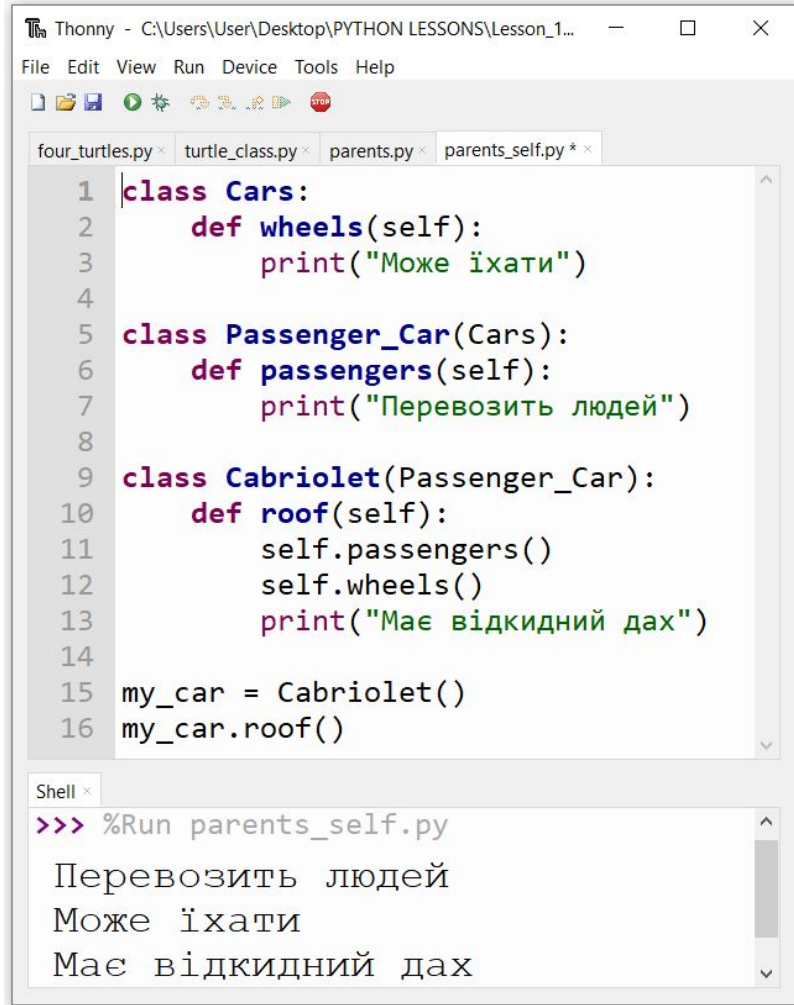
The bottom panel, labeled "Shell", shows the output of running the script:

```
>>> %Run parents.py
Має відкидний дах
Перевозить людей
Може їхати
```

# Виклик функціями інших функцій

Ми можемо викликати функцію класу прямо всередині іншої функції за допомогою параметру **self**.

Завдяки доданим функціям, що викликають інші функції, коли ми створюємо об'єкти цих класів, ми можемо викликати одну функцію, яка робить більше однієї дії.



The screenshot shows the Thonny Python IDE interface. The top menu bar includes File, Edit, View, Run, Device, Tools, and Help. Below the menu is a toolbar with icons for file operations, running, and debugging. The editor window displays a Python script with the following code:

```
1 class Cars:
2     def wheels(self):
3         print("Може їхати")
4
5 class Passenger_Car(Cars):
6     def passengers(self):
7         print("Перевозить людей")
8
9 class Cabriolet(Passenger_Car):
10    def roof(self):
11        self.passengers()
12        self.wheels()
13        print("Має відкидний дах")
14
15 my_car = Cabriolet()
16 my_car.roof()
```

The bottom panel, labeled 'Shell', shows the output of running the script:

```
>>> %Run parents_self.py
Перевозить людей
Може їхати
Має відкидний дах
```

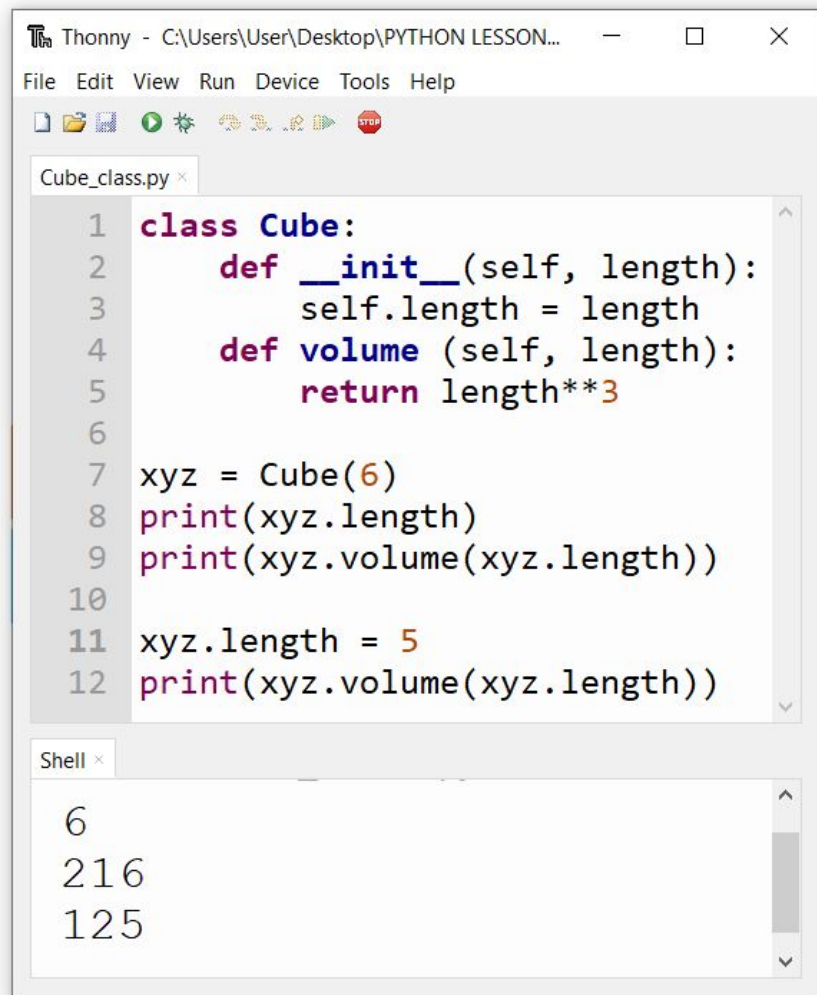


# Ініціалізація об'єкта

це початкова підготовка об'єкта, коли йому надаються певні властивості при створенні.

Для ініціалізації використовується функція `__init__` з параметрами `self` та тими, що відповідають за його властивості. Ця функція викликається автоматично при створенні екземпляру класу.

В подальшому початкові значення можна змінювати.



The screenshot shows the Thonny Python IDE interface. The main editor window displays a Python script named `Cube_class.py` with the following code:

```
1 class Cube:
2     def __init__(self, length):
3         self.length = length
4     def volume (self, length):
5         return length**3
6
7 xyz = Cube(6)
8 print(xyz.length)
9 print(xyz.volume(xyz.length))
10
11 xyz.length = 5
12 print(xyz.volume(xyz.length))
```

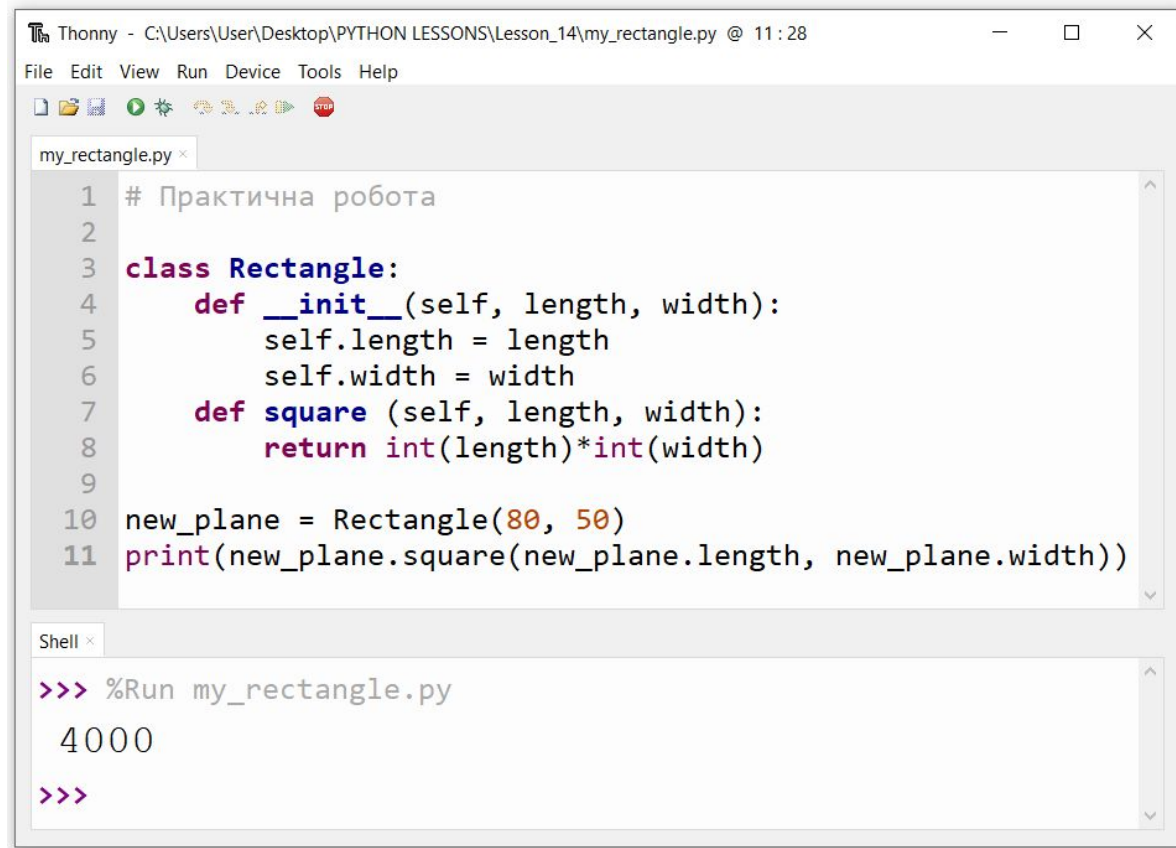
Below the editor is a `Shell` window showing the output of the script:

```
6
216
125
```



## Практична робота

- 1) Створіть новий файл (**New**)
- 2) Створіть клас **Rectangle**, який містить функцію ініціалізації з параметрами довжини та ширини, а також функцію **square** для розрахунку площі прямокутника,
- 3) Використайте цей клас, щоб створити об'єкт **new\_plane** з розмірами **80** та **50**. Використовуючи метод **square** розрахуйте площу об'єкта **new\_plane**
- 4) Збережіть файл (**Save**) під назвою **my\_rectangle.py**
- 5) Виведіть результат у вікно **Shell** за допомогою функції **print()**



The image shows a screenshot of the Thonny Python IDE. The window title is "Thonny - C:\Users\User\Desktop\PYTHON LESSONS\Lesson\_14\my\_rectangle.py @ 11:28". The menu bar includes "File", "Edit", "View", "Run", "Device", "Tools", and "Help". Below the menu bar is a toolbar with icons for file operations and running code. The main editor area displays a Python script in a file named "my\_rectangle.py". The script defines a "Rectangle" class with an "\_\_init\_\_" method and a "square" method. It then creates an instance "new\_plane" and prints the result of "new\_plane.square(new\_plane.length, new\_plane.width)". The bottom panel, labeled "Shell", shows the command "%Run my\_rectangle.py" and the output "4000".

```
1 # Практична робота
2
3 class Rectangle:
4     def __init__(self, length, width):
5         self.length = length
6         self.width = width
7     def square (self, length, width):
8         return int(length)*int(width)
9
10 new_plane = Rectangle(80, 50)
11 print(new_plane.square(new_plane.length, new_plane.width))
```

```
>>> %Run my_rectangle.py
4000
>>>
```

Якщо все зроблено правильно - ви побачите такий результат



## Підсумки

Дізнались, для чого можна використовувати класи

Навчилися використовувати успадковані функції та викликати функції всередині інших функцій

Дізнались, як ініціалізувати об'єкт