

Урок 12

Функції в Python

Повторне використання коду за допомогою функцій і модулів.

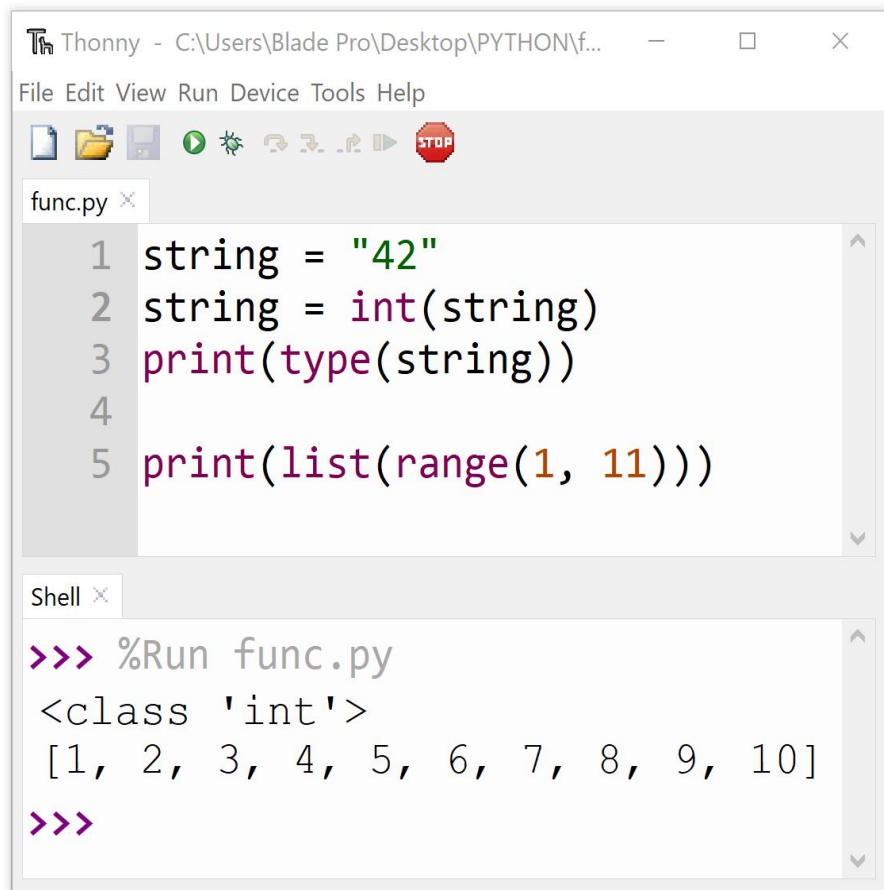


Функція

це **блок коду** для виконання певної дії, який можна застосовувати багато разів у різних частинах своєї програми.

У Python існують вбудовані функції: **print()**, **input()**, **list()**, **range()**, **int()**, **float()**, **str()**, **type()**, **len()** тощо.

Також, функції можна створювати самостійно.



The screenshot shows the Thonny Python IDE interface. The title bar indicates the file path: C:\Users\Blade Pro\Desktop\PYTHON\func.py. The menu bar includes File, Edit, View, Run, Device, Tools, and Help. The toolbar contains icons for opening files, saving, running, and stopping. The main editor window displays the following Python code in func.py:

```
1 string = "42"
2 string = int(string)
3 print(type(string))
4
5 print(list(range(1, 11)))
```

Below the editor is the Shell window, which shows the output of running the script:

```
>>> %Run func.py
<class 'int'>
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>>
```

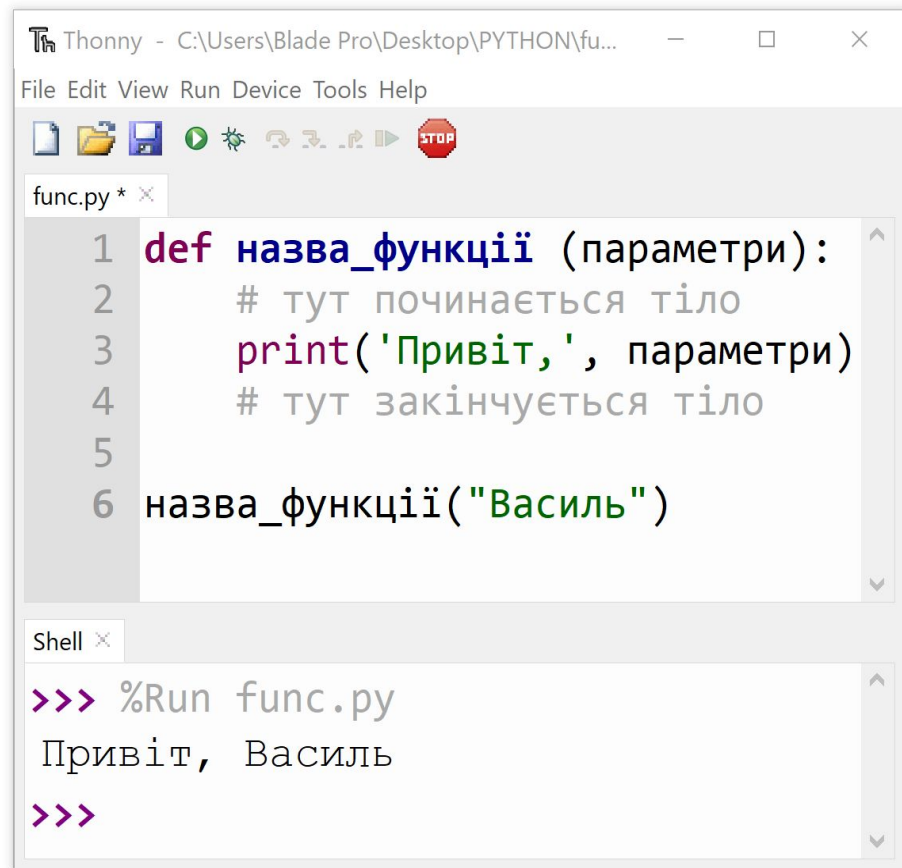
Складові функції

Функція складається з трьох частин:

- назва;
- параметри (змінна);
- тіло.

Для створення функції використовують команду **def**, після якої вказується назва, параметри у дужках та тіло у вкладеному блоці після двокрапки.

Щоб викликати функцію, треба вказати її назву та вказати значення параметрів у дужках.



The screenshot shows the Thonny Python IDE interface. The main editor window displays a Python file named 'func.py' with the following code:

```
1 def назва_функції (параметри):  
2     # тут починається тіло  
3     print('Привіт,', параметри)  
4     # тут закінчується тіло  
5  
6 назва_функції("Василь")
```

Below the editor, the Shell window shows the execution of the script:

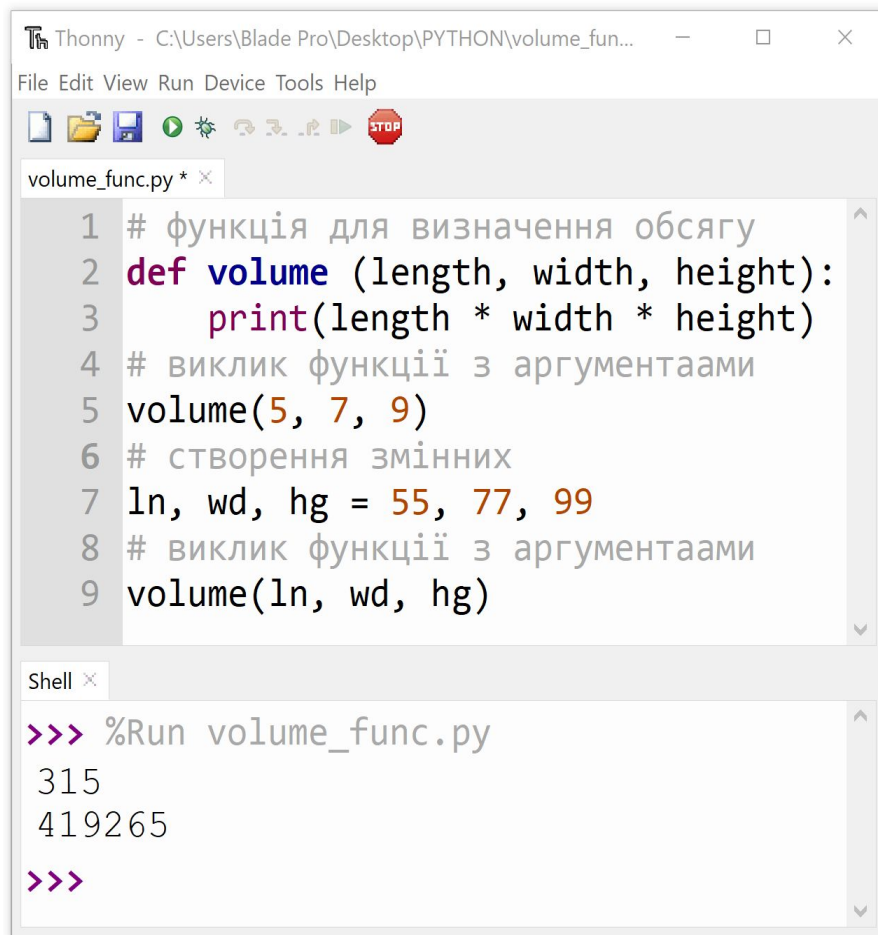
```
>>> %Run func.py  
Привіт, Василь  
>>>
```

Аргументи функції

це **значення параметрів**, які вказуються при користуванні функцією.

Параметрів у функції може існувати декілька. Назви цих параметрів розділяються комою.

Аргументи можна зберігати у змінних, які потім підставляються у функцію.



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

```
1 # функція для визначення обсягу
2 def volume (length, width, height):
3     print(length * width * height)
4 # виклик функції з аргументаами
5 volume(5, 7, 9)
6 # створення змінних
7 ln, wd, hg = 55, 77, 99
8 # виклик функції з аргументаами
9 volume(ln, wd, hg)
```

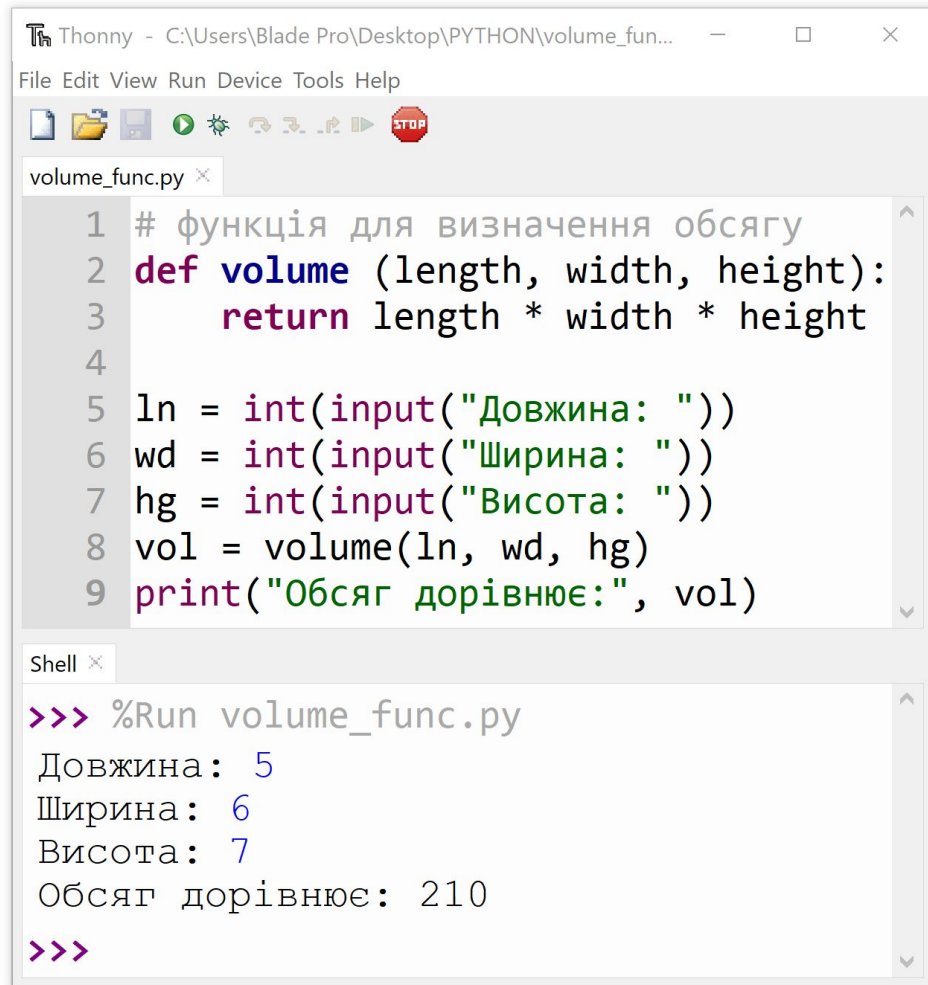
Below the editor, the Shell window shows the execution of the script:

```
>>> %Run volume_func.py
315
419265
>>>
```

Повернення значення функції

За допомогою команди **return** можна повернути результат роботи функції.

Такий результат можна зберігати у змінній для подальшого використання у програмі.



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

```
1 # функція для визначення обсягу
2 def volume (length, width, height):
3     return length * width * height
4
5 ln = int(input("Довжина: "))
6 wd = int(input("Ширина: "))
7 hg = int(input("Висота: "))
8 vol = volume(ln, wd, hg)
9 print("Обсяг дорівнює:", vol)
```

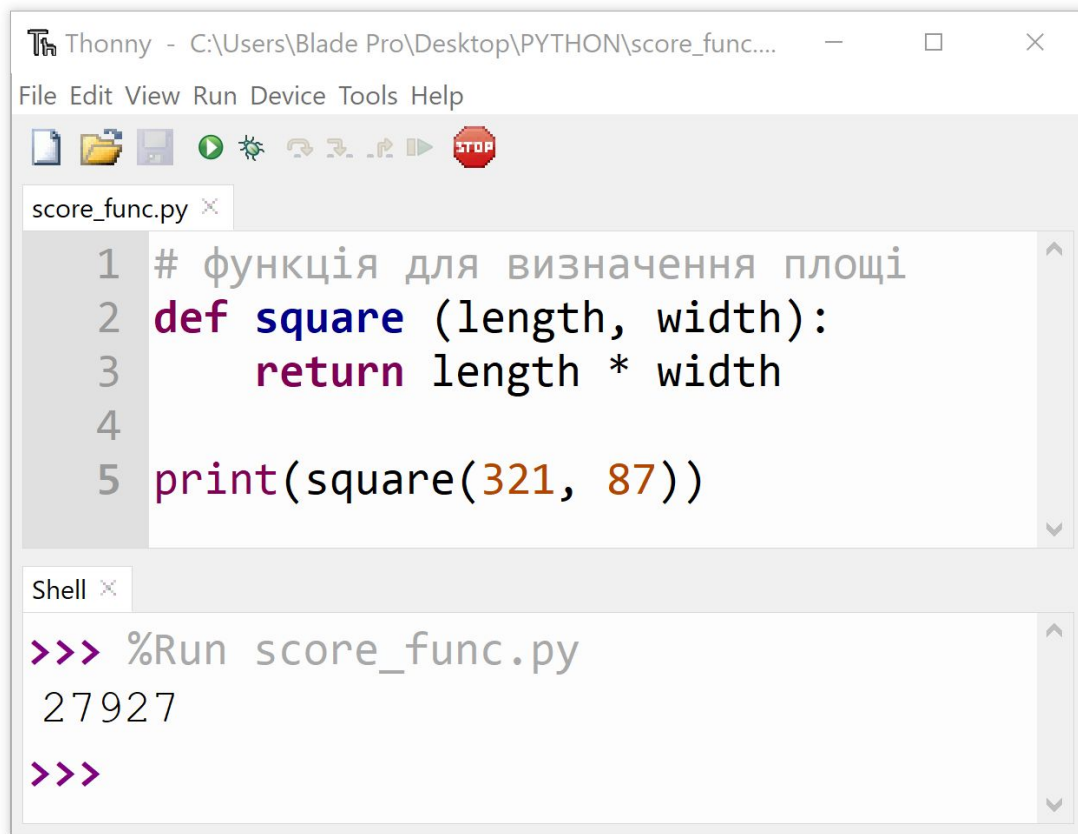
Below the editor is a Shell window showing the execution of the script:

```
>>> %Run volume_func.py
Довжина: 5
Ширина: 6
Висота: 7
Обсяг дорівнює: 210
>>>
```



Практична робота

- 1) Створіть новий файл (**New**)
- 2) Створіть функцію **square**, яка має два параметри - довжину (**length**) та ширину (**width**) та **повертає** результат множення цих параметрів
- 3) Використайте цю функцію для знаходження площі прямокутника з розмірами сторін: **321** і **87** метрів
- 4) Збережіть файл (**Save**) під назвою **square_func.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\Blade Pro\Desktop\PYTHON\score_func....". The menu bar includes "File", "Edit", "View", "Run", "Device", "Tools", and "Help". The toolbar contains icons for opening files, saving, running, and stopping. The editor shows a file named "score_func.py" with the following Python code:

```
1 # функція для визначення площі
2 def square (length, width):
3     return length * width
4
5 print(square(321, 87))
```

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

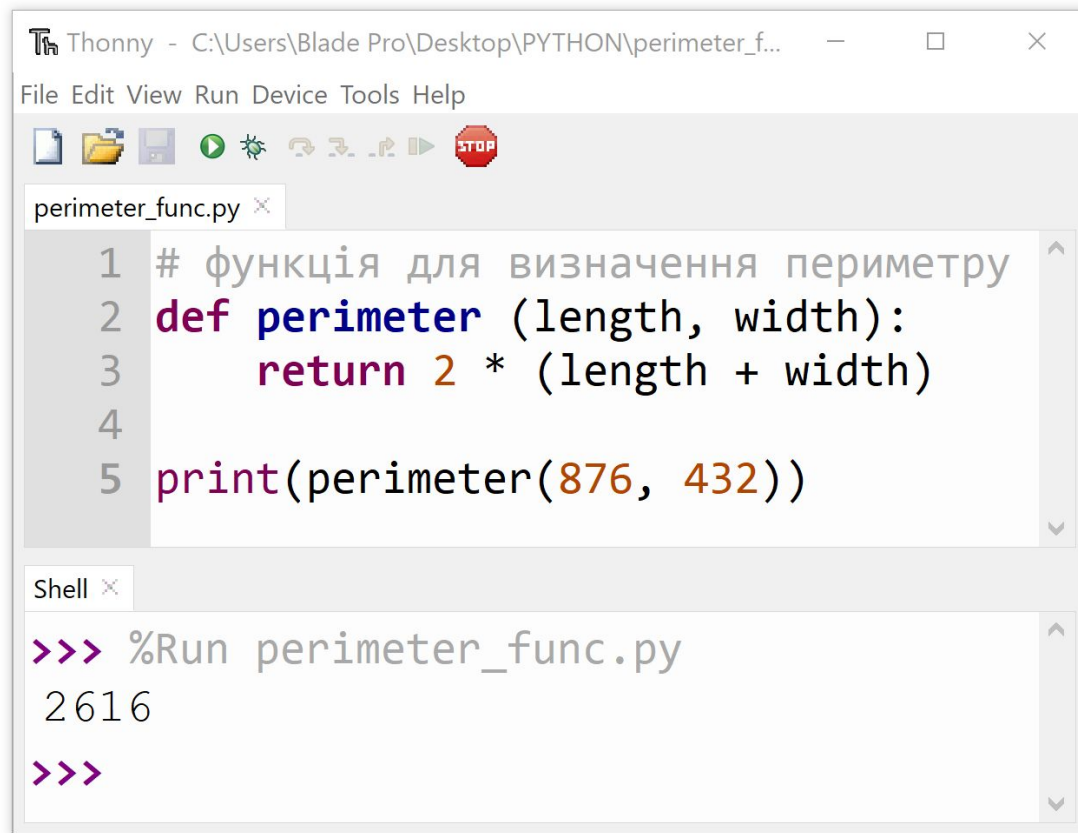
```
>>> %Run score_func.py
27927
>>>
```

Якщо все зроблено правильно - ви побачите такий результат



Практична робота

- 1) Створіть новий файл (**New**)
- 2) Створіть функцію **perimeter**, яка має два параметри - довжину (**length**) та ширину (**width**) та **повертає** подвоєний результат суми цих параметрів
- 3) Використайте цю функцію для знаходження периметру прямокутника з розмірами сторін: **876** і **432** метрів
- 4) Збережіть файл (**Save**) під назвою **perimeter_func.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\Blade Pro\Desktop\PYTHON\perimeter_f...". The menu bar includes "File", "Edit", "View", "Run", "Device", "Tools", and "Help". The toolbar contains icons for opening files, saving, running, and other IDE functions. The main editor window shows a file named "perimeter_func.py" with the following Python code:

```
1 # функція для визначення периметру
2 def perimeter (length, width):
3     return 2 * (length + width)
4
5 print(perimeter(876, 432))
```

Below the editor is a "Shell" window showing the execution of the script:

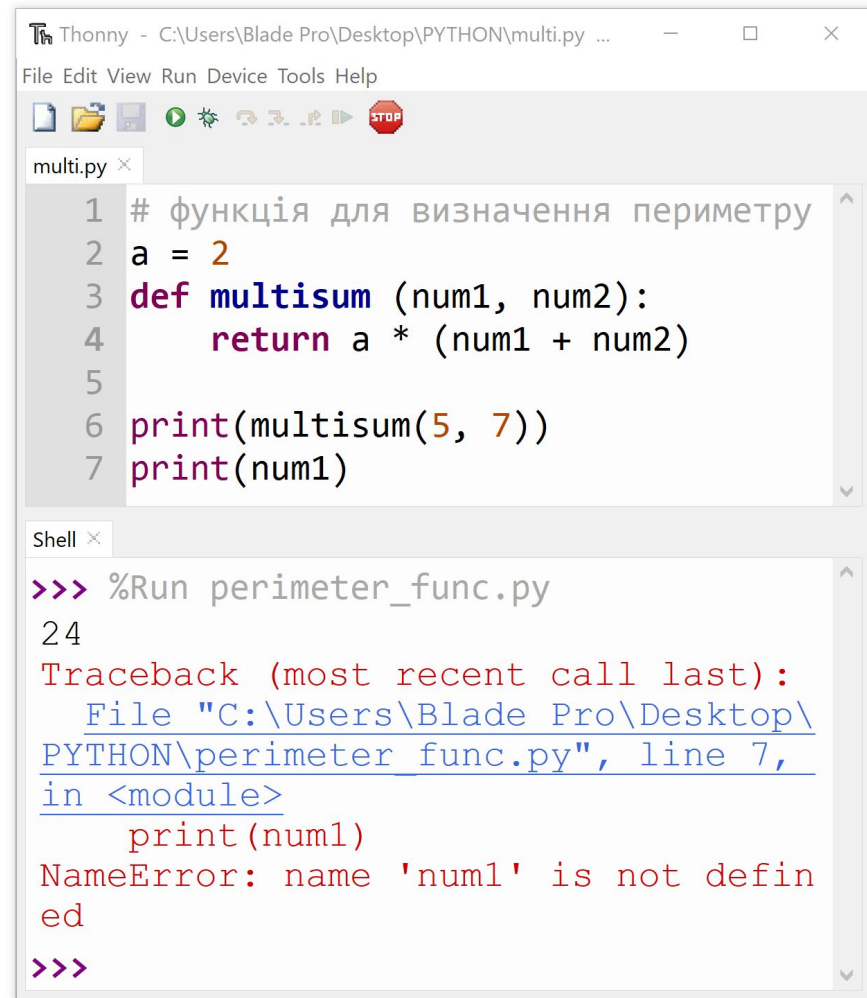
```
>>> %Run perimeter_func.py
2616
>>>
```

Якщо все зроблено правильно - ви побачите такий результат

Область видимості

Змінну, яка перебуває всередині тіла функції, не можна використовувати ще раз, коли функція уже виконала свою дію, бо вона існує тільки в межах функції.

Якщо змінна створена перед функцією, нею можна скористатися всередині функції.



The screenshot shows a Thonny Python IDE window. The title bar reads "Thonny - C:\Users\Blade Pro\Desktop\PYTHON\multi.py ...". 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 "multi.py", contains the following Python code:

```
1 # функція для визначення периметру
2 a = 2
3 def multisum (num1, num2):
4     return a * (num1 + num2)
5
6 print(multisum(5, 7))
7 print(num1)
```

Below the editor is a "Shell" window showing the execution of the script. The prompt is ">>> %Run perimeter_func.py". The output shows a "Traceback (most recent call last):" error. The traceback points to "File 'C:\Users\Blade Pro\Desktop\PYTHON\perimeter_func.py', line 7, in <module>". The code snippet shown in the traceback is "print(num1)". The error message is "NameError: name 'num1' is not defined". The prompt ">>>" is shown at the bottom of the shell window.

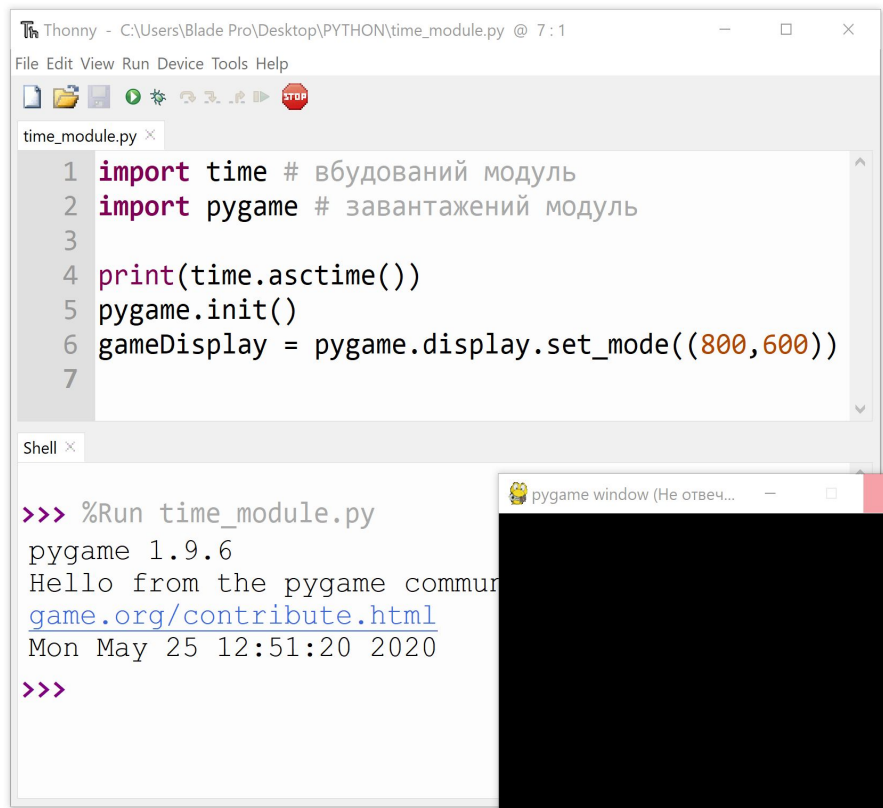
Модуль

це сукупність готових функцій та іншого корисного коду, які можна використовувати в своїй програмі.

Існують вбудовані модулі та ті, що треба завантажувати окремо.

Модуль підключається за допомогою команди **import** у верхній частині програми.

Після підключення модулю його ім'я використовується для доступу до функцій цього модулю.



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

```
1 import time # вбудований модуль
2 import pygame # завантажений модуль
3
4 print(time.asctime())
5 pygame.init()
6 gameDisplay = pygame.display.set_mode((800,600))
7
```

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

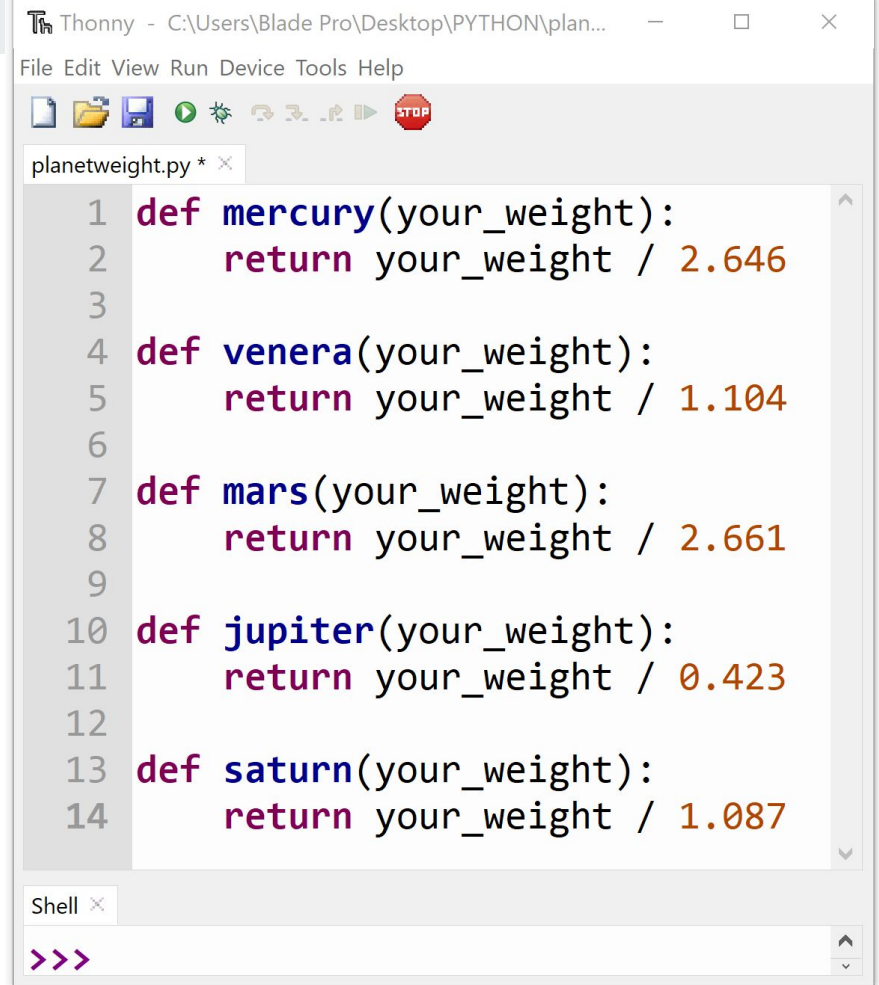
```
>>> %Run time_module.py
pygame 1.9.6
Hello from the pygame community.
www.pygame.org/contribute.html
Mon May 25 12:51:20 2020
>>>
```

To the right of the Shell window, a small window titled "pygame window (He отвеч..." is visible, showing a black screen.

Свій модуль

Цей модуль дозволяє дізнатись вашу вагу на різних планетах Сонячної системи.

- 1) Створюємо новий файл
- 2) Пишемо в ньому код, що наведений справа
- 3) Зберігаємо файл з назвою **planetweight.py**



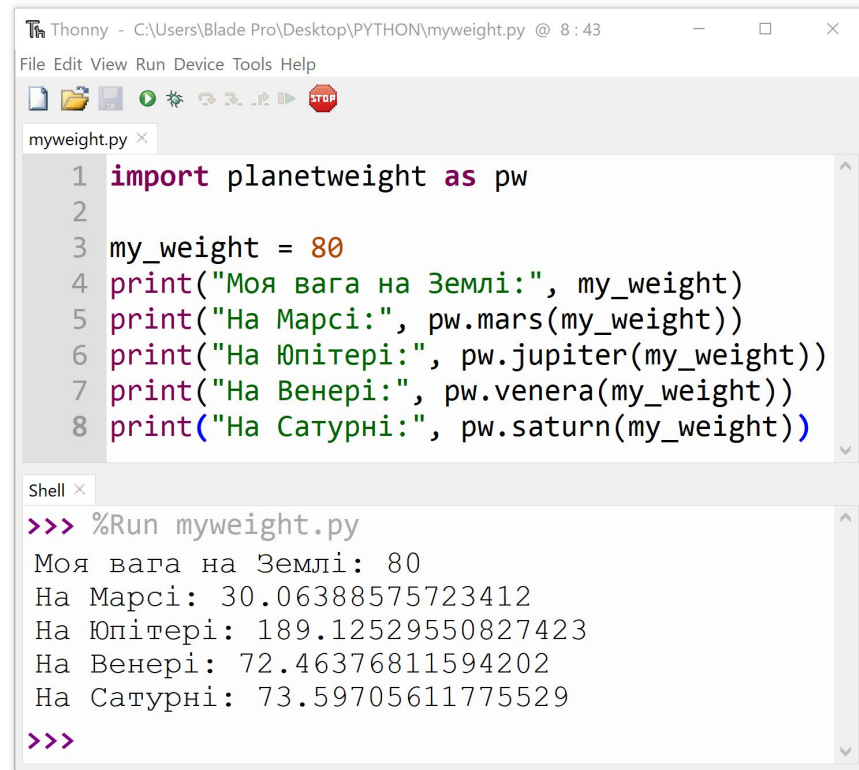
```
Thonny - C:\Users\Blade Pro\Desktop\PYTHON\plan...
File Edit View Run Device Tools Help

planetweight.py * x
1 def mercury(your_weight):
2     return your_weight / 2.646
3
4 def venera(your_weight):
5     return your_weight / 1.104
6
7 def mars(your_weight):
8     return your_weight / 2.661
9
10 def jupiter(your_weight):
11     return your_weight / 0.423
12
13 def saturn(your_weight):
14     return your_weight / 1.087

Shell x
>>>
```

Використання свого модулю

- 1) Створюємо новий файл у тій же директорії, що і модуль
- 2) Підключаємо модуль за допомогою команди **import**
- 3) Використовуємо функції з назвами планет для визначення своєї ваги
- 4) Зберігаємо файл з назвою **myweight.py**



The screenshot shows the Thonny Python IDE interface. The top window displays the code for `myweight.py`, which imports the `planetweight` module and uses its functions to calculate weight on different planets. The bottom window, titled 'Shell', shows the output of running the script, displaying the weight on Earth and other planets.

```
Thonny - C:\Users\Blade Pro\Desktop\PYTHON\myweight.py @ 8 : 43
File Edit View Run Device Tools Help

myweight.py x
1 import planetweight as pw
2
3 my_weight = 80
4 print("Моя вага на Землі:", my_weight)
5 print("На Марсі:", pw.mars(my_weight))
6 print("На Юпитері:", pw.jupiter(my_weight))
7 print("На Венері:", pw.venus(my_weight))
8 print("На Сатурні:", pw.saturn(my_weight))

Shell x
>>> %Run myweight.py
Моя вага на Землі: 80
На Марсі: 30.06388575723412
На Юпитері: 189.12529550827423
На Венері: 72.46376811594202
На Сатурні: 73.59705611775529
>>>
```



Практична робота

Спробуйте створити модуль **sputnikweight**, за допомогою якого можна визначити свою вагу на різних супутниках планет Сонячної системи, у тому числі і на Місяці



Підсумки

Дізнались, що таке **функції**, які вони бувають та для чого вони використовуються

Навчилися створювати **власні функції**

Дізнались, як використовувати **модулі**