CBS Department

Python Functions

Soma Soleiman Zadeh

Object-Oriented Programming (CBS 215)

Fall 2025 - 2026

Week 4

October 29-30, 2025

Outline

- What is **Function?**
- **Built-In Functions** vs. **User-Defined Functions**
- Creating Functions
- **Calling Functions**
- Function **Parameters** and **Arguments**
- The **return** Statement

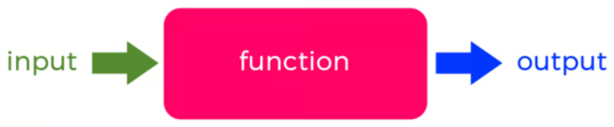
Functions in Mathematics

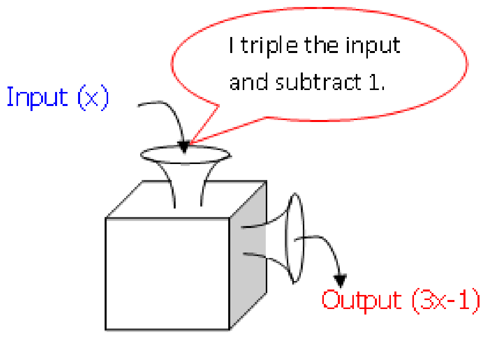
$$f(x) = x^2$$

$$g(x, y) = x^2 + y^2$$

- The concept of functions in programming is similar to mathematical functions.

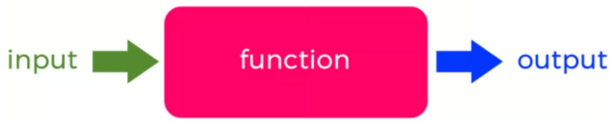
Concept of Function





What is Function?

- A **function** is a reusable block of code that performs a specific task.
- A function **can take input arguments**, **process them**, and **return outputs**.
- Once the function is created, it can be run over and over and over again.
- Whenever we want to run a function, we call it.



When Using Functions

- If you want to **do the same task multiple times**, you can encapsulate the code inside a **Function**.
    - Create the function (including the steps to do the task) ONLY once,
    - Use (call) the function over and over again.

Advantages of Using Functions

- By dividing the large program into small blocks, the code becomes more **readable, organized** and **easy to understand**.
- **Reducing duplication of code**.
- **Reducing the complexity** of a program.

Built-In Functions

- **Built-in functions** are <u>pre-defined functions</u> in any programming language that can be used anytime to perform some common tasks.
- Examples of Built-In functions in Python:

  print( )

  input( )

  range( )

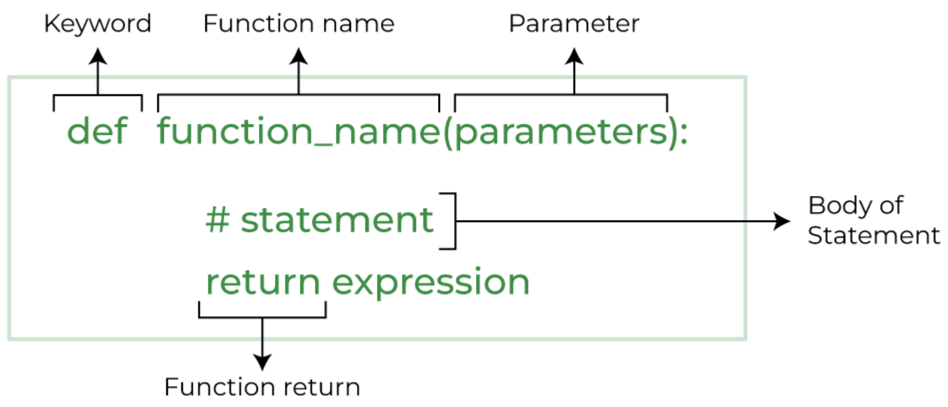
  type( )

  int( )

User-Defined Functions

- **User-defined function** is created by the programmer to perform specific tasks in a program.
- **User-defined functions** allow for **<u>customization</u>**. It means you might need a function to do a specific task, but there is no ready built-in function for it.
  - **What's the solution?** Create a function by yourself.

Syntax of Creating Function in Python



```
def function_name(parameters):
    # statement
    return expression
```

Steps of Defining a Function

Step 1 – Use the keyword **def** to declare the function followed by the **function name**.

Step 2 – Add parameters to the function: they should be within the function's parentheses. End your line with a colon **(:)**.

Step 3 – Add statements that the functions should execute (Function's body).

Step 4 – End your function with a **return** statement if the function should output something. Without the return statement, your function will return an object **None**.

(The **None** object has no value at all.)

Calling Function

- Defining a function using **def** keyword does not execute it.
- Defining a function is for naming the function and specifying what to do when the function is called.
- **Calling the function** executes what we have already specified in the function with the indicated parameters.
- For calling the function, use the **function name** followed by parentheses and pass any required values as inputs.

Function with Parameters and Return Statement

Function Goal: taking two numbers and returning their multiplication.

Step 1 –
Defining the function

```python
def MultiplyTwoNumbers(a , b):
    result = a * b
    return result
```

Step 2 –
Calling the function

```python
mult = MultiplyTwoNumbers(4 , 7)
```

Function with Parameter and No Return Statement

Function Goal: Greet the user by printing ‘Hello’ followed by the user’s name.

Step 1 –
Defining the function

```python
def greeting(firstname):
    print("Hello", firstname)
```

Step 2 –
Calling the function

```python
greeting('Ahmed')
```

Function with No Parameter and No Return Statement

Take an example of such a function, with **NO parameter and NO return statement**.

Step 1 –
Defining the function

```
def greeting():
    print('Hello')
```

Step 2 –
Calling the function

```
greeting()
```

Function with Multiple Return Values

- **Function Goal:** taking two numbers and returning two values: their summation and their multiplication.

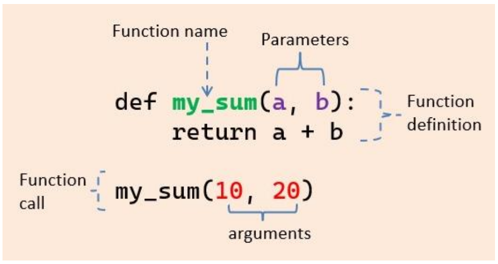
```
def FindSumMultiply (a , b):

    summation = a + b
    multiplication = a * b

    return summation, multiplication

s, m = FindSumMultiply(6 , 5)
print("The summation value is", s)
print("The multiplication value is", m)
```

Two Return Values

Since the function returns two values, when you call the function, you can save the output in two variables.

Parameters vs. Arguments

- A function needs certain information to do its work.
- There is a difference between **parameter** and **argument**.
- A **parameter** is the variable listed inside the parentheses in the function definition. An **argument** is the value that is sent to the function when it is called.



```
def my_sum(a, b):
    return a + b

my_sum(10, 20)
```

Global Variables vs. Local Variables

- **Local Variables:**
  - The **local variable** is declared inside the function blocks.
  - Only statements inside a function can access **local variables**.
  - These variables are destroyed once the function block has finished.
- **Global Variables:**
  - The **global variable** is declared outside of every function of the program.
  - **Global variables** continue to exist until the entire program has been ended.

Example

Function Goal: Create a function that takes in **three numbers** and returns the **<u>smallest number</u>** and the **<u>largest number</u>**.