

What is Function in Python?

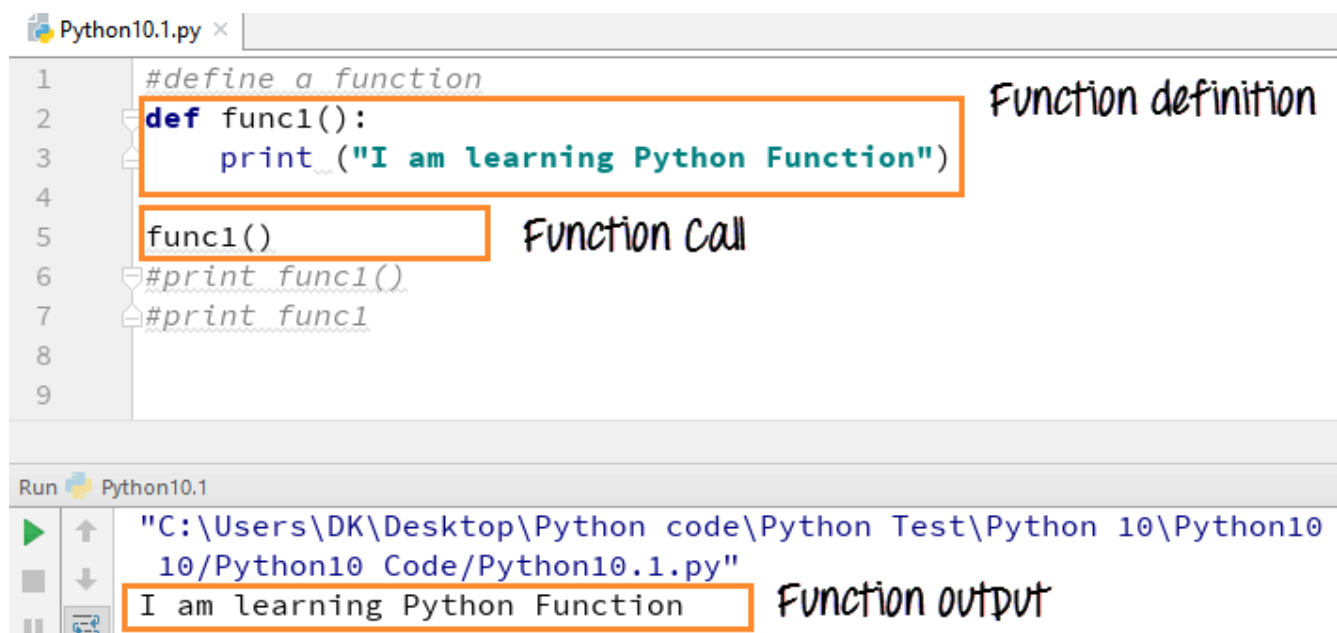
A **Function in Python** is a piece of code which runs when it is referenced. It is used to utilize the code in more than one place in a program. It is also called method or procedure. Python provides many inbuilt functions like `print()`, `input()`, `compile()`, `exec()`, etc. but it also gives freedom to create your own functions.

How to define and call a function in Python

Function in Python is defined by the “**def**” statement followed by the function name and parentheses ())

Example:

Let us define a function by using the command “ **def func1():**” and call the function. The output of the function will be “**I am learning Python function**”.



```
Python10.1.py x
1  #define a function
2  def func1():
3      print("I am learning Python Function")
4
5  func1()
6  #print func1()
7  #print func1
8
9
```

Function definition

Function Call

Run Python10.1

"C:\Users\DK\Desktop\Python code\Python Test\Python 10\Python10 10\Python10 Code\Python10.1.py"

I am learning Python Function

Function output

The function **print func1()** calls our `def func1():` and print the command “ **I am learning Python function.**”

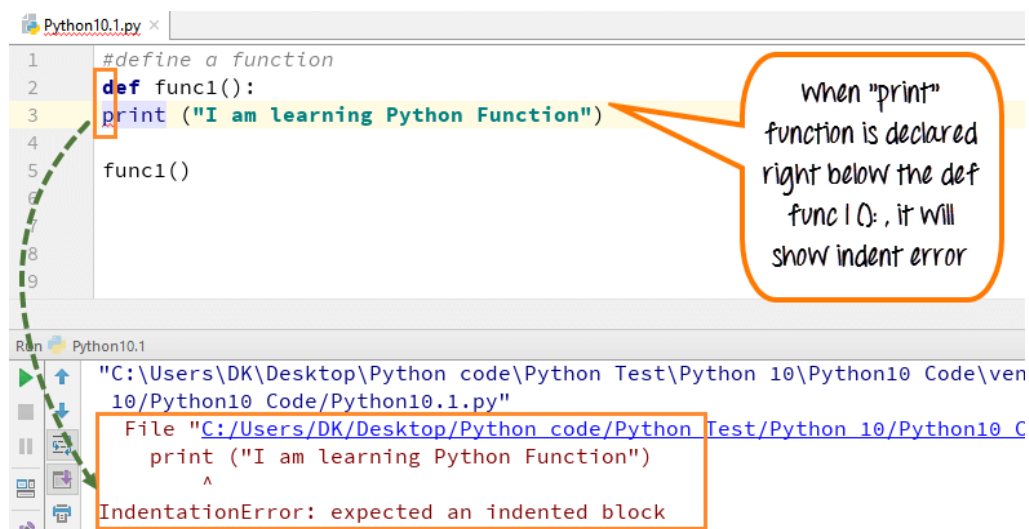
There are set of rules in Python to define a function.

- Any args or input parameters should be placed within these parentheses
- The function first statement can be an optional statement-docstring or the documentation string of the function
- The code within every function starts with a colon (:) and should be indented (space)
- The statement return (expression) exits a function, optionally passing back a value to the caller. A return statement with no args is the same as return None.

Significance of Indentation (Space) in Python

Before we get familiarize with Python functions, it is important that we understand the indentation rule to declare Python functions and these rules are applicable to other elements of Python as well like declaring conditions, loops or variable.

Python follows a particular style of indentation to define the code, since **Python functions don't have any explicit begin or end like curly braces to indicate the start and stop for the function, they have to rely on this indentation.** Here we take a simple example with "print" command. When we write "print" function right below the `def func1():` It will show an **"indentation error: expected an indented block"**.



The screenshot shows a Python IDE with a file named `Python10.1.py`. The code in the editor is as follows:

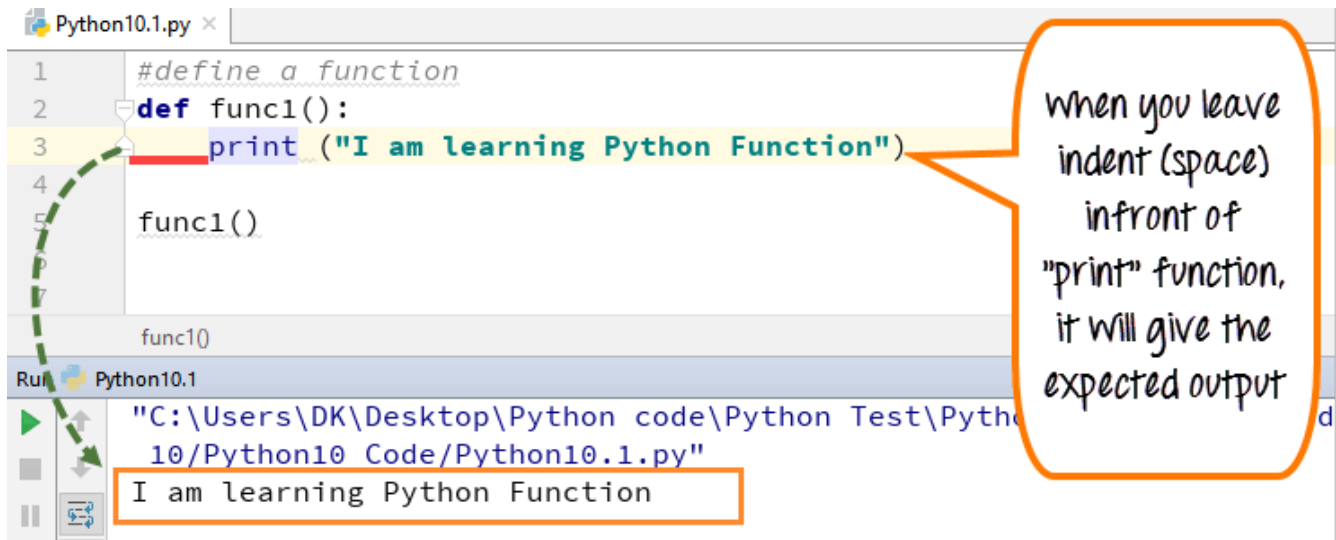
```
1 #define a function
2 def func1():
3 print ("I am learning Python Function")
4
5 func1()
6
7
8
9
```

A callout box points to line 3, stating: "When 'print' function is declared right below the def func1(), it will show indent error".

The bottom of the screenshot shows the command prompt output:

```
Run Python10.1
"C:\Users\DK\Desktop\Python code\Python Test\Python 10\Python10 Code\ven
10\Python10 Code\Python10.1.py"
File "C:/Users/DK/Desktop/Python code/Python Test/Python 10/Python10_C
print ("I am learning Python Function")
^
IndentationError: expected an indented block
```

Now, when you add the indent (space) in front of “`print`” function, it should print as expected.



```
1 #define a function
2 def func1():
3     print ("I am learning Python Function")
4
5 func1()
```

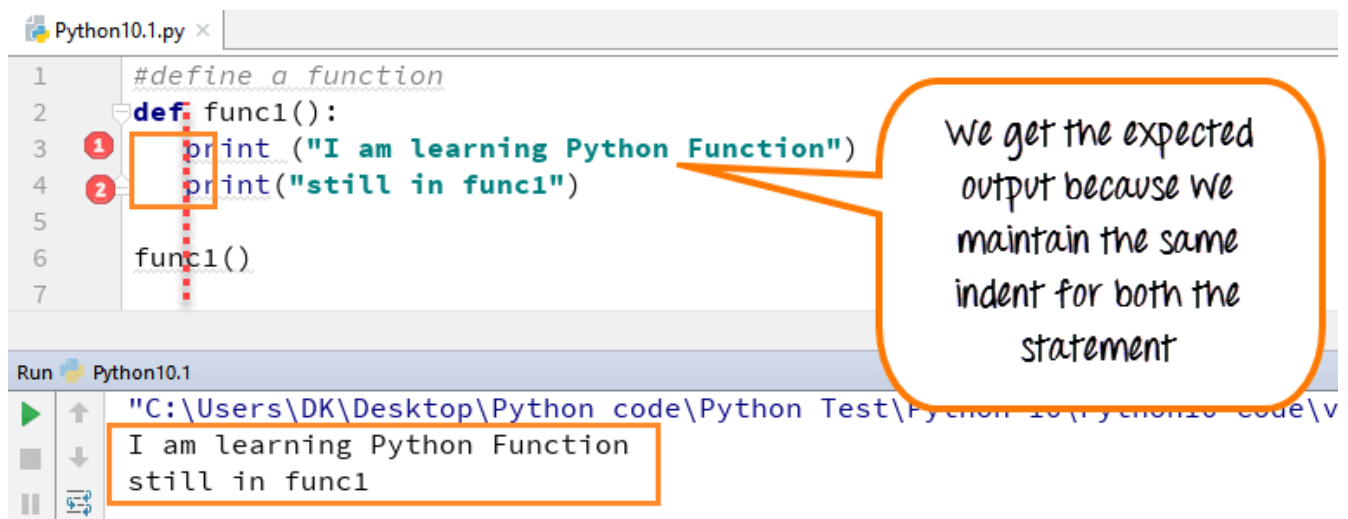
When you leave indent (space) in front of "print" function, it will give the expected output

Run Python10.1

"C:\Users\DK\Desktop\Python code\Python Test\Python 10\Python10 Code\Python10.1.py"

I am learning Python Function

Now, when we apply same indentation incase we have 2 statements to print and align them in the same line, it gives the expected output.



```
1 #define a function
2 def func1():
3     print ("I am learning Python Function")
4     print("still in func1")
5
6 func1()
```

We get the expected output because we maintain the same indent for both the statement

Run Python10.1

"C:\Users\DK\Desktop\Python code\Python Test\Python 10\Python10 Code\Python10.1.py"

I am learning Python Function
still in func1

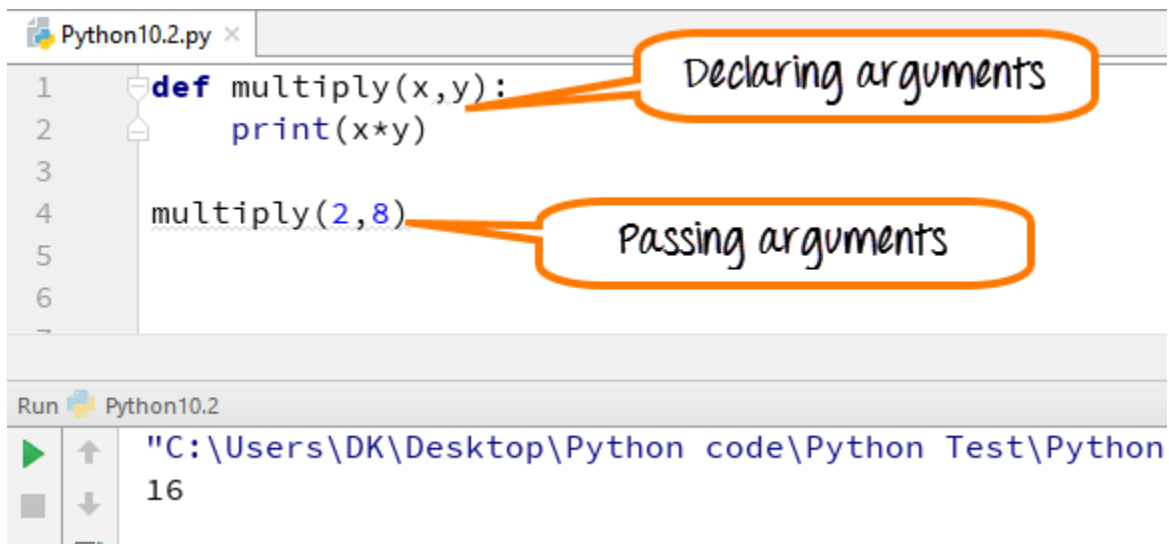
Arguments in Functions

The argument is a value that is passed to the function when it's called.

In other words on the calling side, it is an argument and on the function side it is a parameter.

Let see how Python Args works -

Step 1) Arguments are declared in the function definition. While calling the function, you can pass the values for that args as shown below



The screenshot shows a Python IDE window titled 'Python10.2.py'. The code is as follows:

```
1 def multiply(x,y):
2     print(x*y)
3
4 multiply(2,8)
5
6
```

Two orange callout boxes with handwritten text are present:

- A box labeled "Declaring arguments" points to the function definition `def multiply(x,y):` on line 1.
- A box labeled "Passing arguments" points to the function call `multiply(2,8)` on line 4.

Below the code editor is a 'Run' button and a console output area. The console shows the output of the program:

```
"C:\Users\DK\Desktop\Python code\Python Test\Python
16
```

Read more

<https://www.geeksforgeeks.org/python-functions/>