

---

---

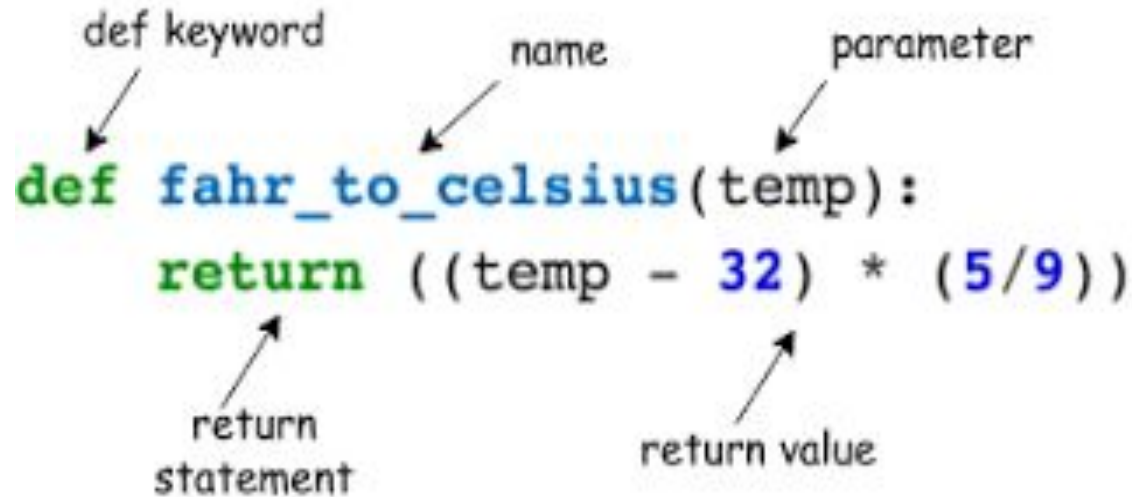
# Python Function

— ODIN Outsourcing —

---

---

# How does function work-1?



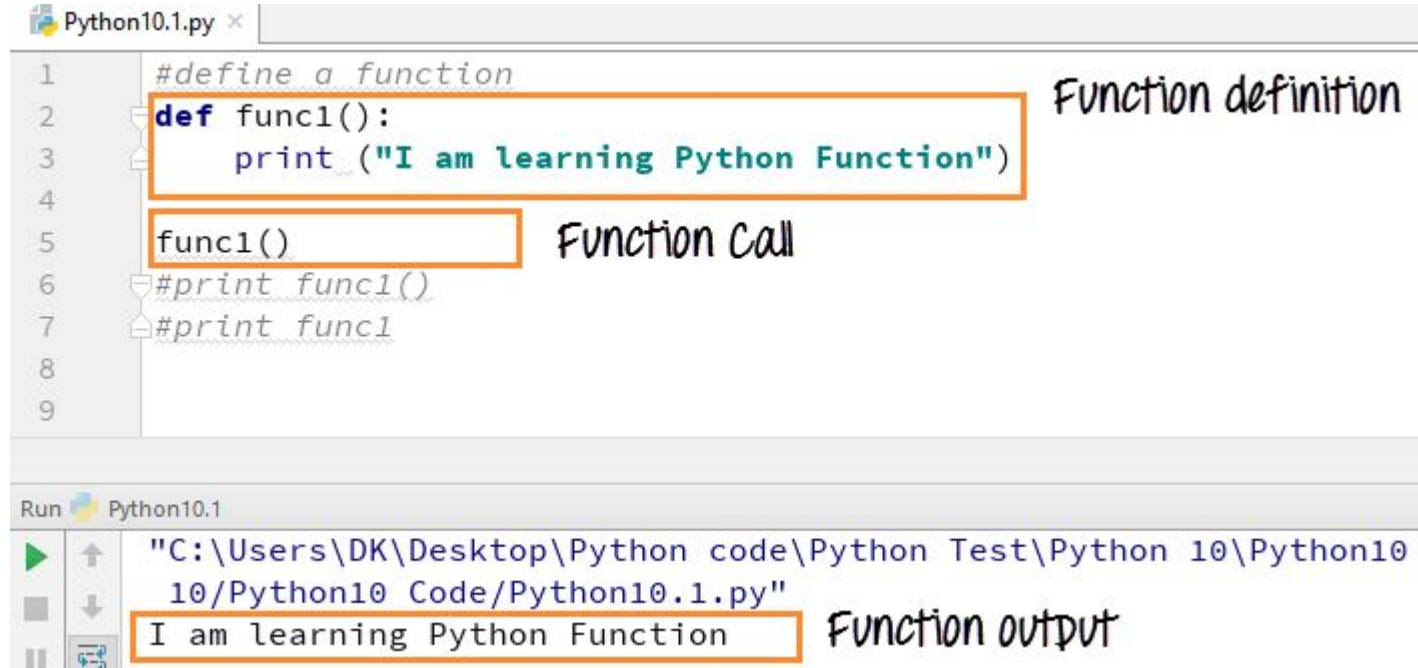
The diagram illustrates the components of a Python function definition. It shows the code `def fahr_to_celsius(temp):` on the first line and `return ((temp - 32) * (5/9))` on the second line. Annotations with arrows point to specific parts: 'def keyword' points to 'def', 'name' points to 'fahr\_to\_celsius', 'parameter' points to 'temp', 'return statement' points to 'return', and 'return value' points to the expression '((temp - 32) \* (5/9))'.

```
def fahr_to_celsius(temp):  
    return ((temp - 32) * (5/9))
```

Annotations:

- def keyword
- name
- parameter
- return statement
- return value

# How does function work-2?



The screenshot shows a Python IDE with a file named 'Python10.1.py'. The code is as follows:

```
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
```

Annotations on the code:

- Function definition:** Points to the `def func1():` line.
- Function Call:** Points to the `func1()` line.

The Run console at the bottom shows the execution path and output:

```
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
```

Annotations on the console:

- Function output:** Points to the output text `I am learning Python Function`.

# How does function work-3?

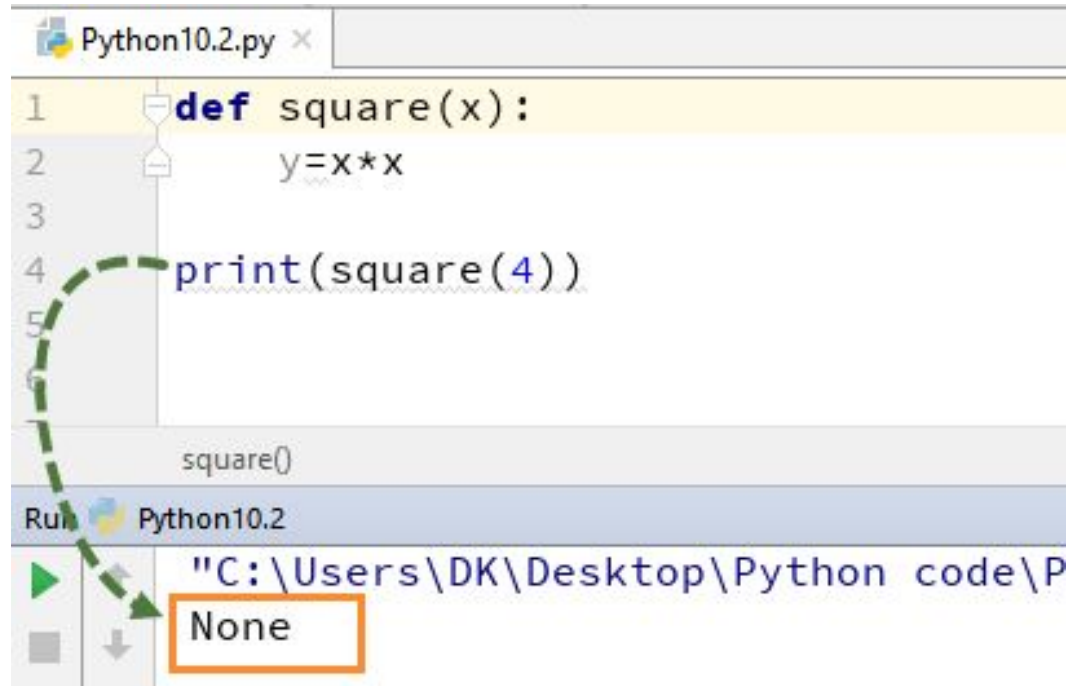
```
Python10.2.py x
1  #define return function
2  def square(x):
3      print(x*x)
4
5  print(square(4))
6
7
```

Run Python10.2

"C:\Users\DK\Desktop\Python 10.2.py" 16  
None

The function does not return anything. Hence output is None

# How does function work-4?



```
Python10.2.py x
1 def square(x):
2     y=x*x
3
4 print(square(4))
5
6
7 square()
Run Python10.2
"C:\Users\DK\Desktop\Python code\Python10.2.py"
None
```

The screenshot shows a Python IDE window titled "Python10.2.py". The code contains a function definition `def square(x):` on line 1, followed by `y=x*x` on line 2, and a call to the function `print(square(4))` on line 4. A green dashed arrow originates from the `square(4)` part of the print statement and points to the `def square(x):` line, illustrating the function call. Below the code editor, the output console shows the command `Run Python10.2` and the resulting output `"C:\Users\DK\Desktop\Python code\Python10.2.py"`. The output `None` is highlighted with an orange box, indicating that the function does not return a value.

# How does function work-5?

```
1 def square(x):  
2     return x*x  
3  
4 print(square(4))  
5  
6
```

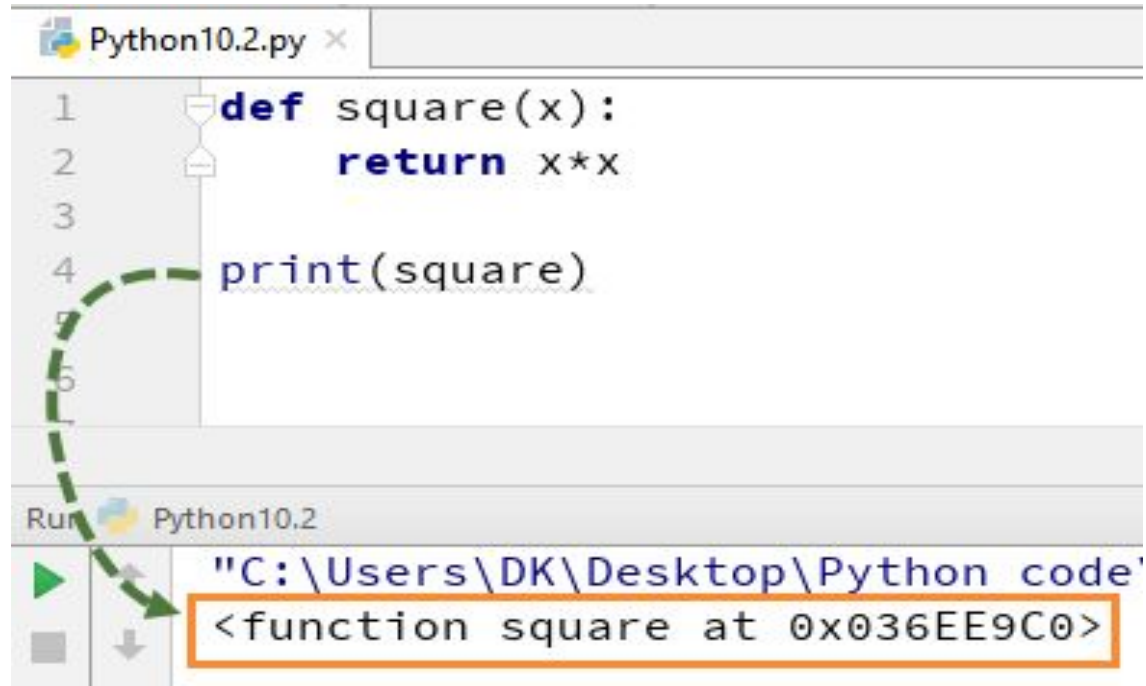
Run Python10.2

"C:\Users\DK\Desktop\Pyth... \P

16

Here we have used "return command" to return the value of function, which is square of (4) i.e 16

# How does function work-6?



The screenshot shows a Python IDE window titled "Python10.2.py". The code editor contains the following Python code:

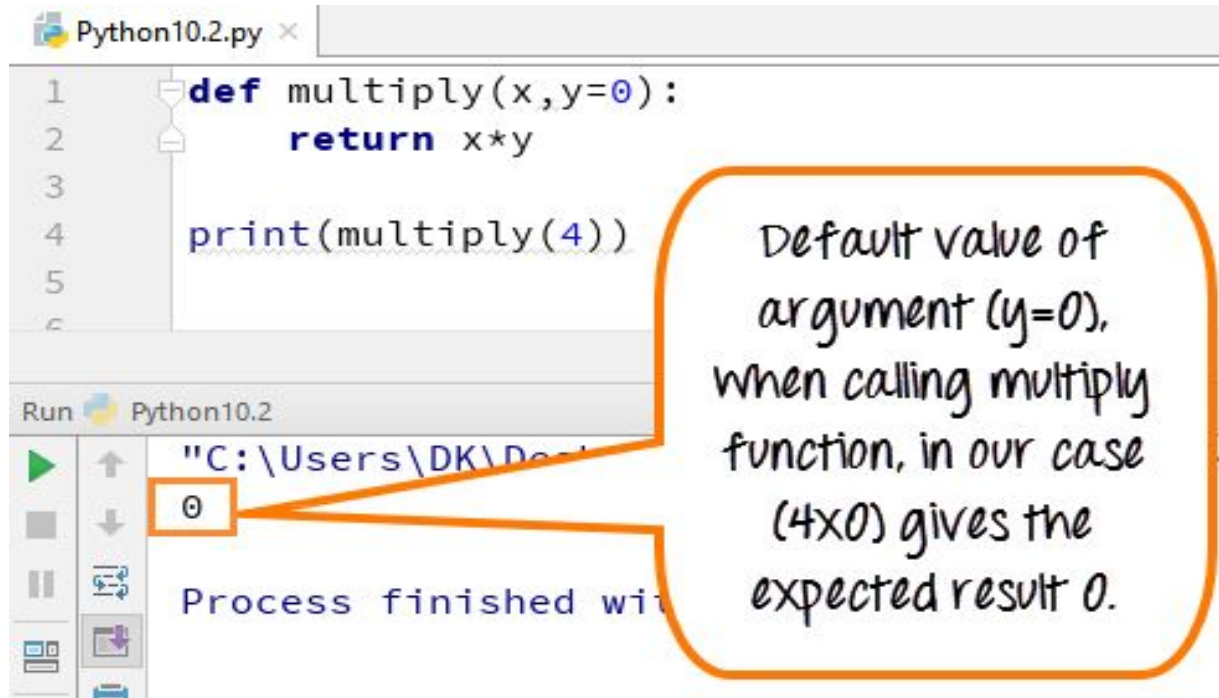
```
1 def square(x):  
2     return x*x  
3  
4 print(square)
```

A green dashed arrow originates from the `print(square)` statement on line 4 and points to the output window. The output window, titled "Run Python10.2", displays the following output:

```
"C:\Users\DK\Desktop\Python code"  
<function square at 0x036EE9C0>
```

The output `<function square at 0x036EE9C0>` is highlighted with an orange rectangular box.

# How does function work-7?



The screenshot shows a Python IDE window titled "Python10.2.py". The code editor contains the following Python code:

```
1 def multiply(x,y=0):  
2     return x*y  
3  
4 print(multiply(4))  
5  
6
```

Below the code editor, the "Run" button is clicked, and the output console shows the execution path: "C:\Users\DK\Desktop" and the result "0". The output "0" is highlighted with an orange box. A callout bubble points from this box to the right, containing the following text:

Default value of argument (y=0), when calling multiply function, in our case (4x0) gives the expected result 0.



# How does function work-8?

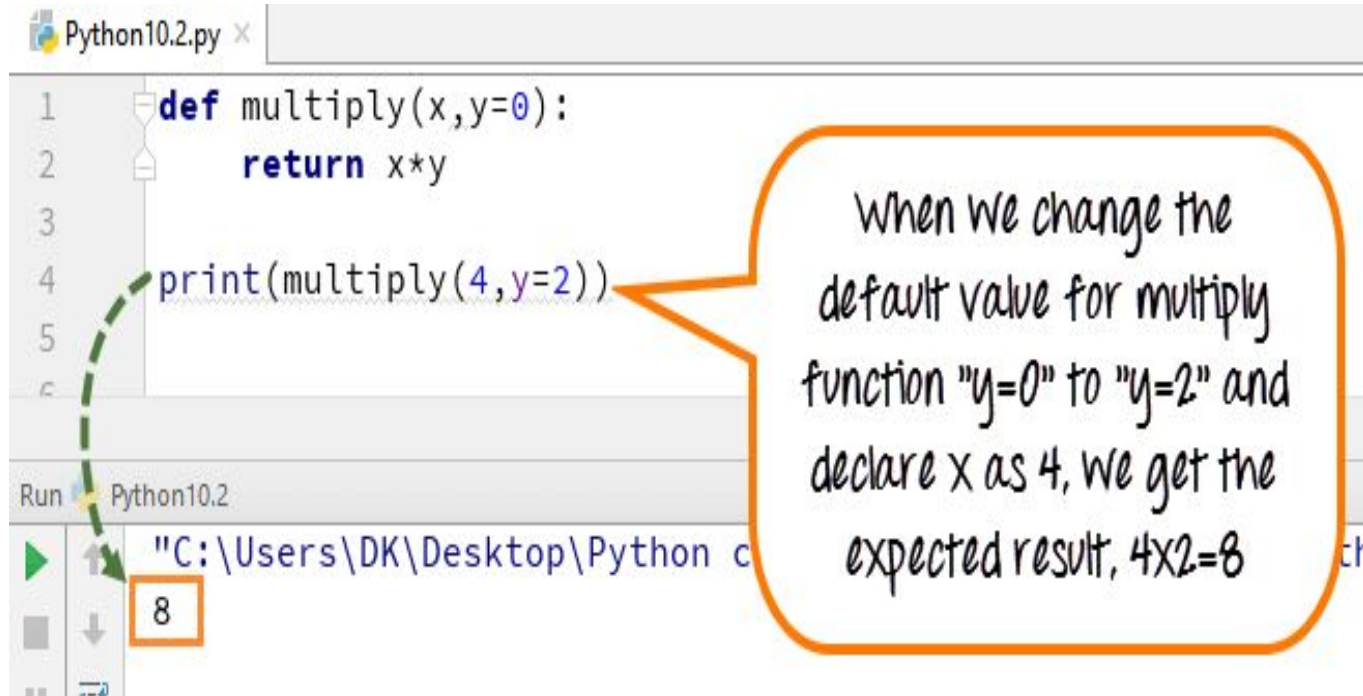
The screenshot shows a Python IDE window titled "Python10.2.py". The code is as follows:

```
1 def multiply(x,y=0):
2     return x*y
3
4 print(multiply(4))
5
6
```

Below the code editor, the "Run" button is clicked, and the output console shows the command prompt "C:\Users\DK\Desktop>" followed by the number "0". The number "0" is highlighted with an orange box. A callout box points to this "0" with the text:

Default value of argument (y=0), when calling multiply function, in our case (4x0) gives the expected result 0.

# How does function work-9?



```
Python10.2.py x
1 def multiply(x,y=0):
2     return x*y
3
4 print(multiply(4,y=2))
5
6
```

Run Python10.2

"C:\Users\DK\Desktop\Python c

8

When we change the default value for multiply function "y=0" to "y=2" and declare x as 4, we get the expected result,  $4 \times 2 = 8$

# How does function work-10?

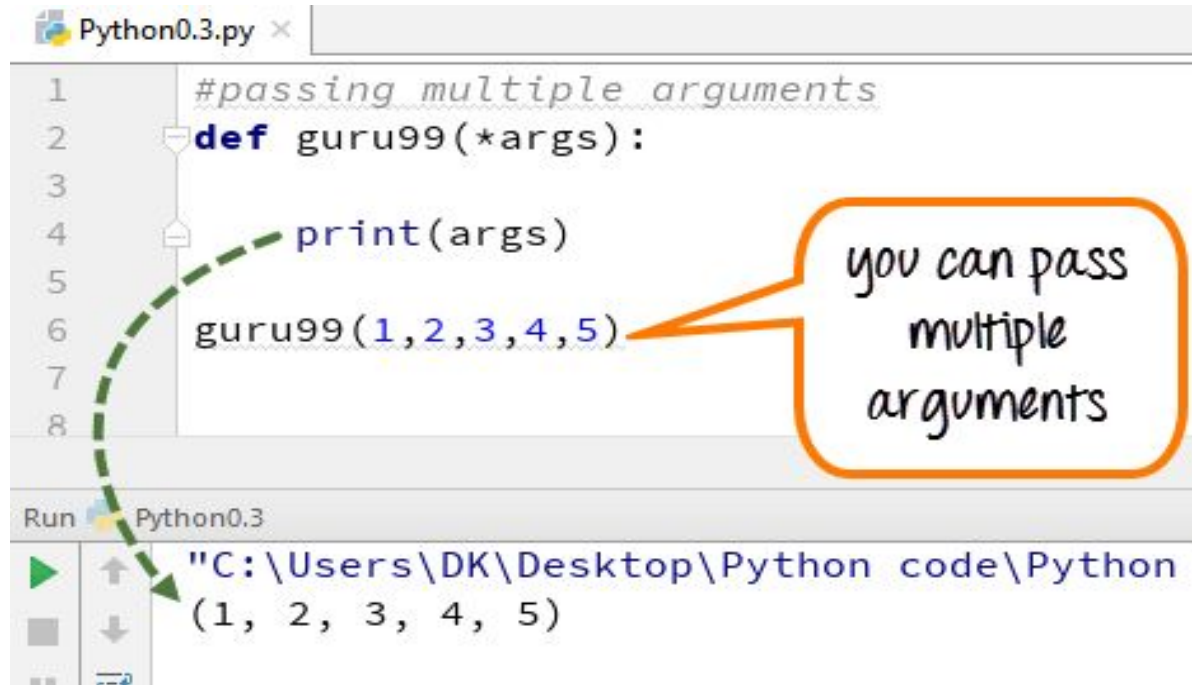
The screenshot shows a Python IDE with a file named 'Python10.2.py'. The code defines a function 'multiply' with two parameters, 'x' and 'y', where 'y' has a default value of 0. The function prints the values of 'x' and 'y' and returns their product. Below the function definition, the function is called with 'y=2' and 'x=4'. A callout bubble points to this call, stating: 'Here we have reversed the order of the value for x and y.' The output window at the bottom shows the execution results: the file path, 'value of x= 4', 'value of y= 2', and the result '8'. Green dashed arrows connect the function call in the code to the corresponding output lines.

```
1 def multiply(x,y=0):
2     print("value of x=",x)
3     print("value of y=",y)
4
5     return x*y
6
7 print(multiply(y=2,x=4))
8
9
```

Run Python10.2

"C:\Users\DK\Desktop\Python code\Python Test\Python 10\Python10.2.py"  
value of x= 4  
value of y= 2  
8

# How does function work-11?



```
Python0.3.py x
1  #passing multiple arguments
2  def guru99(*args):
3
4      print(args)
5
6      guru99(1,2,3,4,5)
7
8
Run Python0.3
"C:\Users\DK\Desktop\Python code\Python
(1, 2, 3, 4, 5)
```

you can pass multiple arguments

# Learning Resources:

1. Function-1: <https://www.guru99.com/functions-in-python.html> ( \*\*\*\*\* )
2. Function-2: <https://introcs.cs.princeton.edu/python/21function/>
3. Function-3: <https://python.swaroopch.com/functions.html>
4. Inner Function:  
<https://realpython.com/inner-functions-what-are-they-good-for/>

# Contract your instructor!

Find Me: <http://rafsanjani.pythonanywhere.com/contact>

Course Website: <https://mrzResearchArena.github.io/Big-Data-using-Python>

Thank you!