Python


Python
Functions

Python Functions

A function is a block of code that performs a specific task.

Types of function

There are two types of function in Python programming:

- **Standard library functions** - These are built-in functions in Python that are available to use.
- **User-defined functions** - We can create our own functions based on our requirements.

Python Function Declaration

The syntax to declare a function is:

```
def function_name(arguments):
    # function body

    return
```

Here,

- `def` - keyword used to declare a function
- `function_name` - any name given to the function
- `arguments` - any value passed to function
- `return` (optional) - returns value from a function

Calling a Function in Python

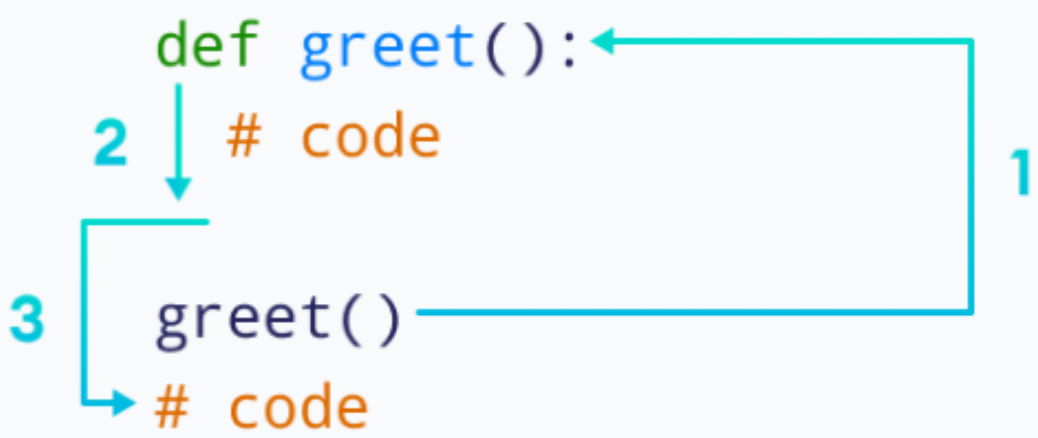
In the above example, we have declared a function named `greet()`.

```
def greet():
    print('Hello World!')
```

Now, to use this function, we need to call it.

Here's how we can call the `greet()` function in Python.

```
# call the function
greet()
```



Working of Python Function

Here,

- When the function is called, the control of the program goes to the function definition.
- All codes inside the function are executed.
- The control of the program jumps to the next statement after the function call.

Python Function Arguments

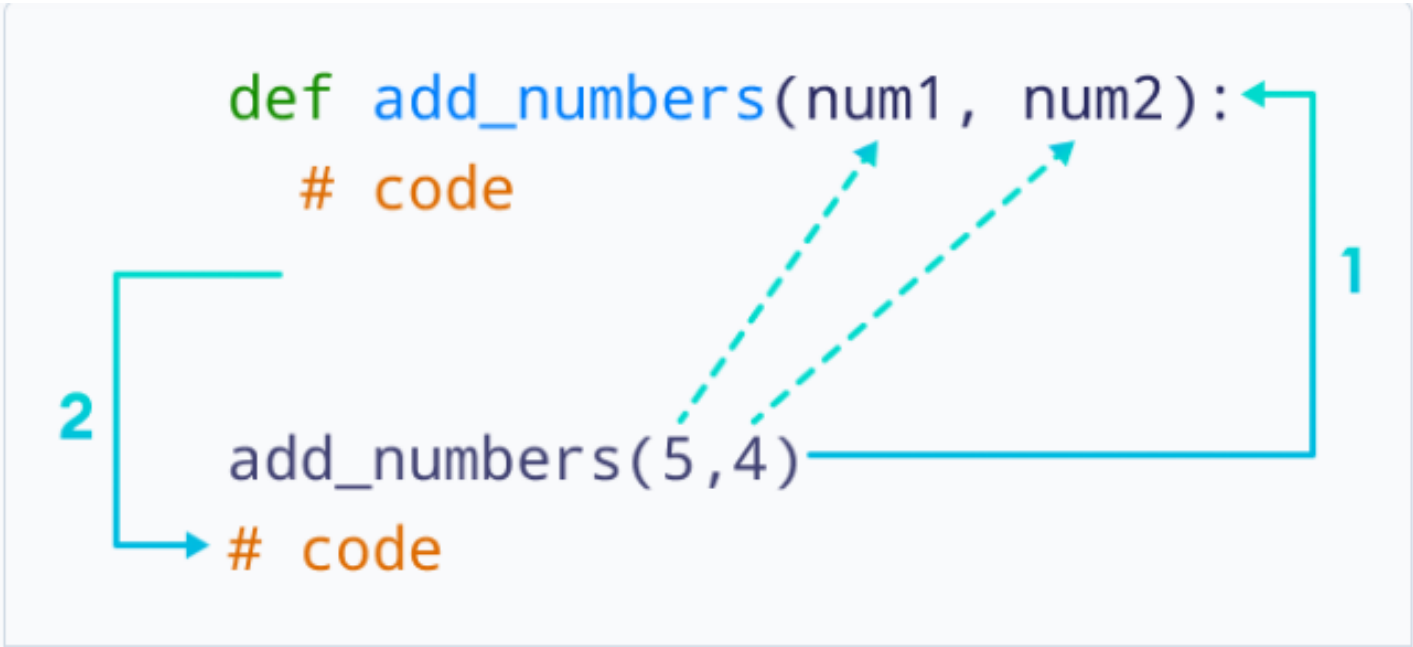
As mentioned earlier, a function can also have arguments. An argument is a value that is accepted by a function. For example,

```
# function with two arguments
def add_numbers(num1, num2):
    sum = num1 + num2
    print('Sum: ',sum)

# function with no argument
def add_numbers():
    # code
```

If we create a function with arguments, we need to pass the corresponding values while calling them. For example,

```
# function call with two values
add_numbers(5, 4)

# function call with no value
add_numbers()
```

Here, `add_numbers(5, 4)` specifies that arguments `num1` and `num2` will get values **5** and **4** respectively.



Python Function with Arguments

We can also call the function by mentioning the argument name as:

```
add_numbers(num1 = 5, num2 = 4)
```

In Python, we call it Keyword Argument (or named argument). The code above is equivalent to

```
add_numbers(5, 4)
```

The return Statement in Python

A Python function may or may not return a value. If we want our function to return some value to a function call, we use the `return` statement. For example,

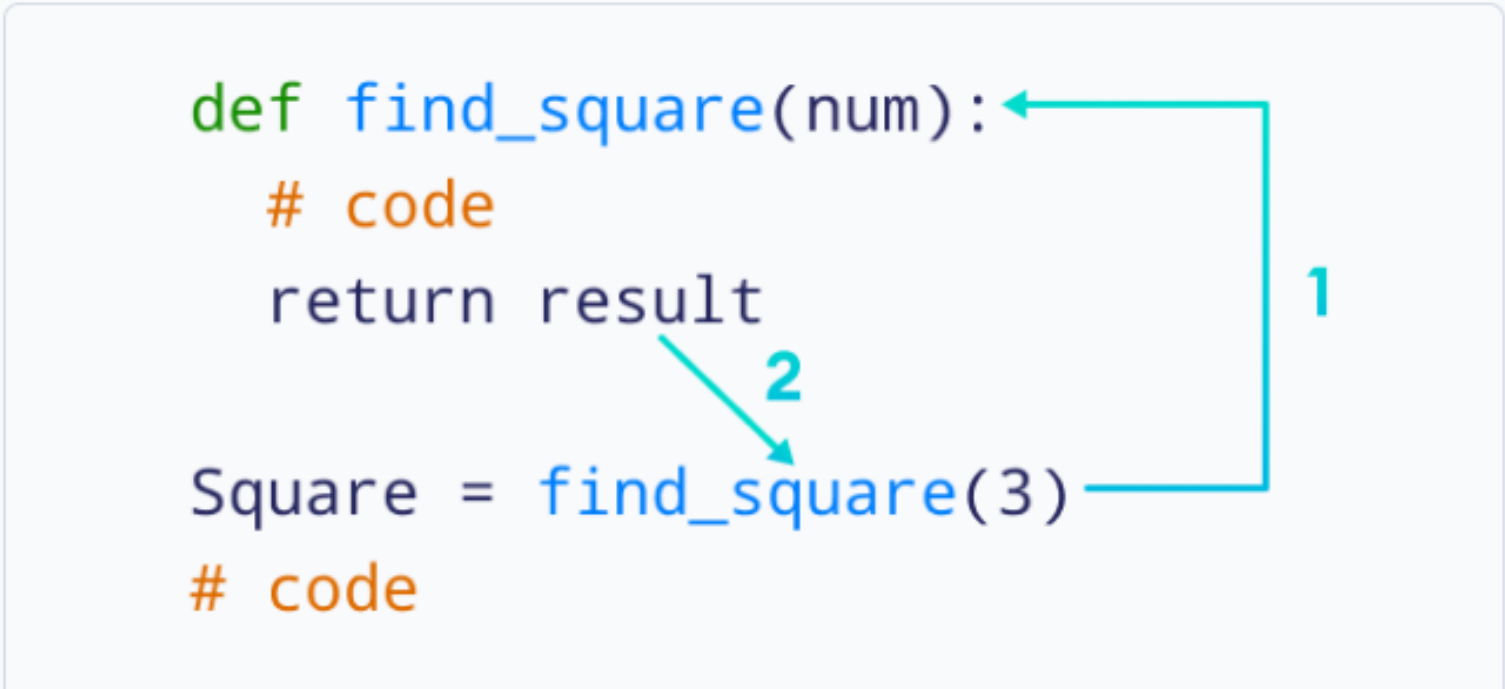
```
def add_numbers():
    ...
    return sum
```

Here, we are returning the variable `sum` to the function call.

> **Note:** The `return` statement also denotes that the function has ended. Any code after return is not executed.

```python
# function definition
def find_square(num):
    result = num * num
    return result

# function call
square = find_square(3)

print('Square:',square)

# Output: Square: 9
```

In the above example, we have created a function named `find_square()`. The function accepts a number and returns the square of the number.



Working of functions in Python

Example 3: Add Two Numbers

```
# function that adds two numbers
def add_numbers(num1, num2):
    sum = num1 + num2
    return sum

# calling function with two values
result = add_numbers(5, 4)

print('Sum: ', result)

# Output: Sum: 9
```

Python Library Functions

In Python, standard library functions are the built-in functions that can be used directly in our program. For example,

- `print()` - prints the string inside the quotation marks
- `sqrt()` - returns the square root of a number
- `pow()` - returns the power of a number

These library functions are defined inside the module. And, to use them we must include the module inside our program.

For example, `sqrt()` is defined inside the `math` module.

```
import math

# sqrt computes the square root
square_root = math.sqrt(4)

print("Square Root of 4 is",square_root)

# pow() comptes the power
power = pow(2, 3)

print("2 to the power 3 is",power)
```

Output

```
Square Root of 4 is 2.0
2 to the power 3 is 8
```

In the above example, we have used

- `math.sqrt(4)` - to compute the square root of **4**
- `pow(2, 3)` - computes the power of a number i.e. 2^3