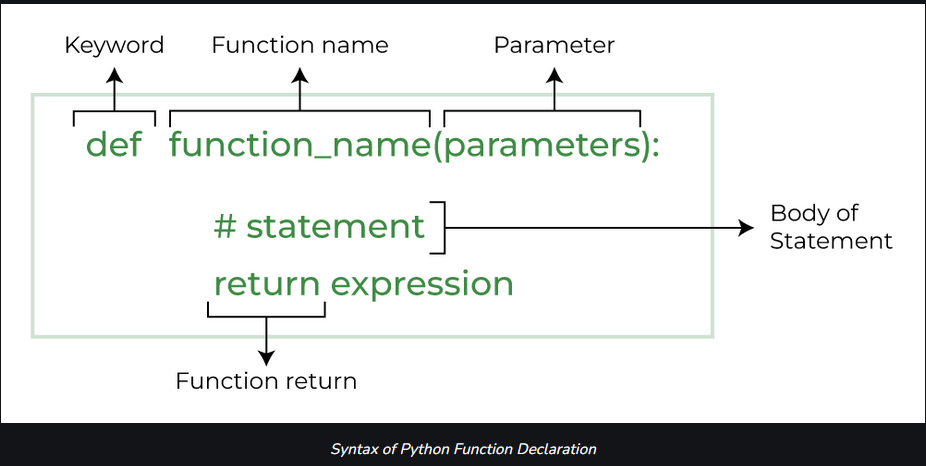
**Functions**

-- **Python Functions** is a block of statements that return the specific task.

-- The idea is to put some commonly or repeatedly done tasks together and make a function so that instead of writing the same code again and again for different inputs, we can do the function calls to reuse code contained in it over and over again.

-- Some **Benefits of Using Functions**

* Increase Code Readability
* Increase Code Reusability



**Types of Functions in Python**

There are mainly two types of functions in Python.

* **Built-in library function:** These are standard functions in Python that are available to use.
* **User-defined function:** We can create our own functions based on our requirements

**Creating a Python Function:**

-- We can create a user-defined function in Python, using the **def** keyword. We can add any type of functionalities and properties to it as we require.

# *A simple Python function*

def my\_*fun*():

  print("Hello World")

**Calling a Python Function:**

-- After creating a function in Python we can call it by using the name of the function followed by parenthesis containing parameters of that particular function.

# *A simple Python function*

def *my\_fun*():

  print("Hello World")

# *Driver code to call a function*

my\_fun()

**Python Function With Parameters:**

-- If you have experience in C/C++ or Java then you must be thinking about the *return type* of the function and *data type* of arguments.

-- That is possible in Python as well (specifically for Python 3.5 and above).

*def function\_name(parameter: data\_type) -> return\_type:*

*"""Docstring"""*

*# body of the function*

*return expression*

**Parameters:**

-- A parameter is the variable defined within the parentheses during function definition. Simply they are written when we declare a function.

# *Here a,b are the parameters*

def sum(a,b):

  print(a+b)

sum(1,2)

**Arguments:**

-- An argument is a value that is passed to a function when it is called. It might be a variable, value or object passed to a function or method as input. They are written when we are calling the function.

def sum(a,b):

  print(a+b)

# *Here the values 1,2 are arguments*

sum(1,2)

**Types of Arguments:**

Python functions can contain two types of arguments:

* Positional Arguments
* Keyword Arguments

***Positional Argument:***

-- Positional Arguments are needed to be included in proper order i.e the first argument is always listed first when the function is called, second argument needs to be called second and so on.

# *Positional argument*

def *person\_name*(first\_name,second\_name):

  print(first\_name+second\_name)

# *First name is Chandlar placed first*

# *Second name is Bing place second*

person\_name("Chandlar","Bing")

***Keyword Argument:***

-- Keyword Arguments is an argument passed to a function or method which is preceded by a keyword and an equal to sign.

-- The order of keyword argument with respect to another keyword argument does not matter because the values are being explicitly assigned.

#*### Keyword Argument*

def *person\_name*(first\_name, second\_name):

  print(first\_name + second\_name)

# *Here we are explicitly assigning the values*

person\_name(second\_name="Bing”, first\_name="Chandlar")

-- Following example of Python function with parameters uses arguments and parameters that we have just seen above.

# *Python Function With Parameters*

def *add*(num1: int, num2: int) -> int:

    """Add two numbers"""

    num3 = num1 + num2

*return* num3

# *Driver code*

num1, num2 = 5, 15

ans = add(num1, num2)

print(f"The addition of {num1} and {num2} results {ans}.")

**Python Function With Default Argument:**

-- A default argument is a parameter that assumes a default value if a value is not provided in the function call for that argument. The following example illustrates Default arguments.

#*### Python function with default argument*

def *myFun*(x, y=50):

    print("x: ", x)

    print("y: ", y)

# *Driver code (We call myFun() with only argument)*

myFun(10)

**Python Function with Arbitrary Keyword Argument:**

-- In Python Arbitrary Keyword Arguments, \*args, and \*\*kwargs can pass a variable number of arguments to a function using special symbols. There are two special symbols:

* \*args in Python (Non-Keyword Arguments)
* \*\*kwargs in Python (Keyword Arguments)

**Example 1:** Variable length non-keywords argument

# *Python program to illustrate \*args for variable number of arguments*

def *myFun*(\*argv):

*for* arg in argv:

        print(arg)

myFun('Chandlar', 'Rachel', 'Joey', 'Monica', 'Pheebe', 'Ross')

**Example 2:** Variable length keyword arguments

# *Python program to illustrate \*kwargs for variable number of keyword arguments*

def *myFun*(\*\*kwargs):

*for* key, value in kwargs.items():

        print("%s == %s" % (key, value))

# *Driver code*

myFun(first='Hello', mid='MF', last='World')

**Python Function Within Functions**

-- A function that is defined inside another function is known as the inner function or nested function.

-- Nested functions are able to access variables of the enclosing scope.

-- Inner functions are used so that they can be protected from everything happening outside the function.

#*### Python program to demonstrate accessing of variables of nested functions*

def *f1*():

    s = 'I love Python'

    def *f2*():

        print(s)

    f2()

# *Driver's code*

f1()

**Return Statement in Python Function:**

-- The function return statement is used to exit from a function and go back to the function caller and return the specified value or data item to the caller.

*return [expression\_list]*

-- The return statement can consist of a variable, an expression, or a constant which is returned at the end of the function execution.

-- If none of the above is present with the return statement a None object is returned.