UNIT V

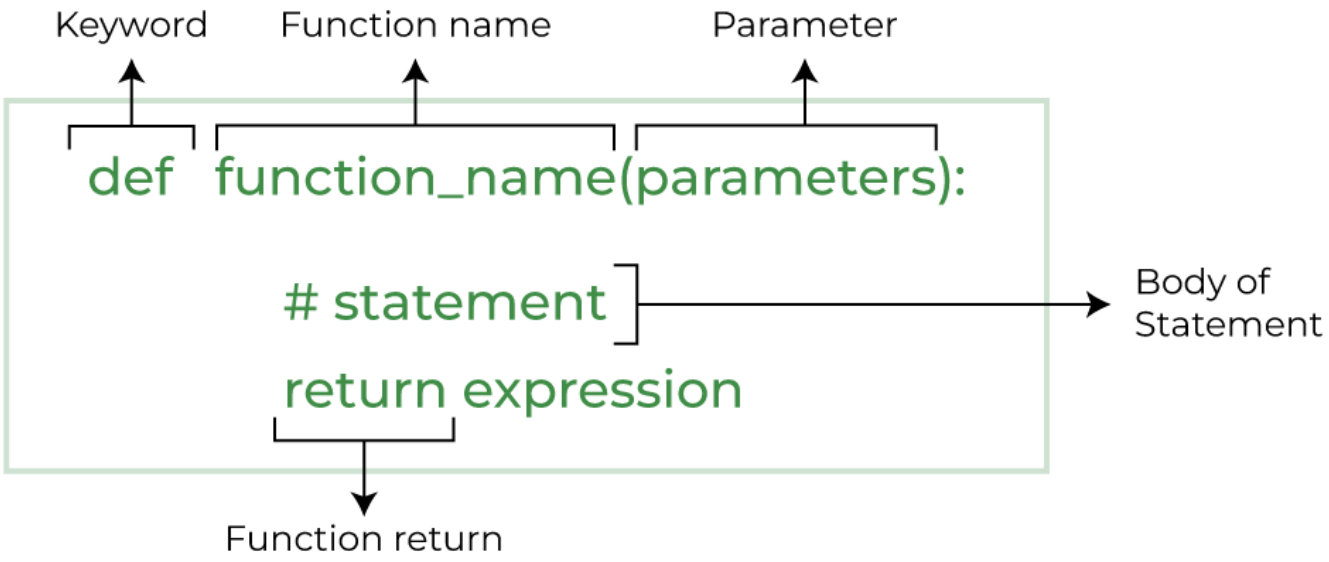
Python Functions is a block of statements that return the specific task. The idea is to put some commonly or repeatedly done tasks together and make a function so that instead of writing the same code again and again for different inputs, we can do the function calls to reuse code contained in it over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

Some **Benefits of Using Functions**

- Increase Code Readability
- Increase Code Reusability

Python Function Declaration

The syntax to declare a function is:



Syntax of Python Function Declaration

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 Function in Python

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 fun():
  print("Welcome to GFG")
```

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 fun():
    print("Welcome to GFG")


# Driver code to call a function
fun()
```

Output:

Welcome to GFG

Python Function Arguments

Arguments are the values passed inside the parenthesis of the function. A function can have any number of arguments separated by a comma.

In this example, we will create a simple function in Python to check whether the number passed as an argument to the function is even or odd.

```
# A simple Python function to check
# whether x is even or odd
def evenOdd(x):
    if (x % 2 == 0):
        print("even")
    else:
        print("odd")



# Driver code to call the function
evenOdd(2)
evenOdd(3)
```

Output:

```
even
odd
```

Types of Python Function Arguments

Python supports various types of arguments that can be passed at the time of the function call. In Python, we have the following 4 types of function arguments.

- **Default argument**

- **Keyword arguments (named arguments)**
- **Positional arguments**
- **Arbitrary arguments** (variable-length arguments *args and **kwargs)

Let's discuss each type in detail.

Default Arguments

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 program to demonstrate
# default arguments
def myFun(x, y=50):
    print("x: ", x)
    print("y: ", y)



# Driver code (We call myFun() with only
# argument)
myFun(10)
```

Output:

```
x:  10
y:  50
```

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 GeeksforGeeks'

    def f2():
        print(s)

    f2()

# Driver's code
f1()
```

Output:

I love GeeksforGeeks

Anonymous Functions in Python

In Python, an anonymous function means that a function is without a name. As we already know the def keyword is used to define the normal functions and the lambda keyword is used to create anonymous functions.

```
# Python code to illustrate the cube of a number
# using lambda function
def cube(x): return x*x*x

cube_v2 = lambda x : x*x*x

print(cube(7))
print(cube_v2(7))
```

Output:

343
343

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. **The syntax for the return statement is:**

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.

Example: Python Function Return Statement

```
def square_value(num):
    """This function returns the square
    value of the entered number"""
    return num**2


print(square_value(2))
print(square_value(-4))
```

Output:

4
16