Python FUNCTIONS Concept :

- A **function in Python** is a named block of organized and connected statements that performs a specific task.
- In other words, a function is a block of code that performs a specific and well-defined task. It organizes the code into a logical way to perform a certain task.
- Function is a syntax of structure in which we represents the reusable business logic.
- Every function is called by a certain name and the block of code inside the function gets executed when we call the function.
- If you have a block of code that gets invoked more than once, put it into a function. This is because function allows us to use a block of code repeatedly in different sections of a program.
- It makes the program more efficient by minimizing the repetition. It makes the code reusable.
- Functions help to break down long and complex programs into smaller and manageable segments and enhance program readability and comprehensibility.
- They provide better modularity for our application program and a high degree of code reusability. Sometimes, we call the function with another name called method. But, there is a slight difference between a function and method.
- Functions are defined outside the class, whereas methods are defined inside the class.

Syntax to define User defined Function in Python

The user creates or defines user-defined functions. The general syntax to declare a user defined function in Python is as follows:

```
def function_name(parameters): # Function header.
    """docstring"""
    . . . . . .
    . . . . . .

    body of the function
    . . . . . .
    . . . . . .
    return [expression or values]
```

In the above syntax, the first line of function definition is header, and the rest is the body of function. The function definition begins with def keyword followed by the function name, parentheses (()), and then colon (:). The parameter list placed within parentheses is optional.

Let's define a simple function that prints the message "Hello Python".

```
# Function definition.

def msg():

    """This statement prints a message."""

    print('Hello Python')
```

Components of Python Functions

A block of statements that defines a function in Python consists of the following parts:

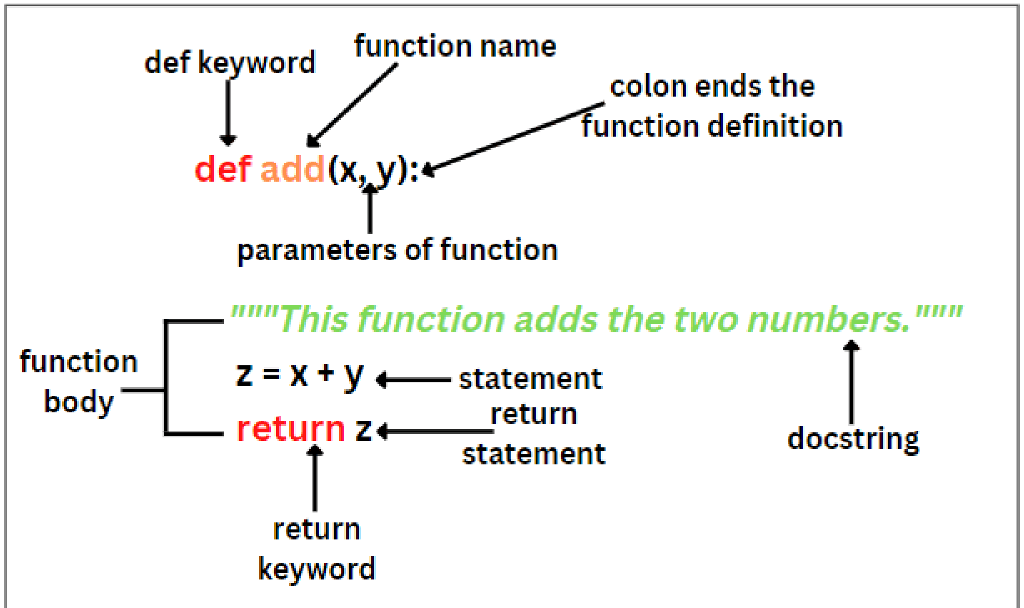

Fig: Function definition

1. **Keyword def:**

Function definition starts with the keyword 'def' that defines the function. It tells the beginning of the function header.

2. **Function name:**

The function_name represents the name of a function. Every function must be a unique user-defined name or an identifier and given to it in the function header statement.

It is an excellent practice to name your function according to the work it performs. In the above example, the function name is msg, and add.

3. Parameters:

The formal parameters (arguments) placed inside the parentheses are optional. A parameter is a piece of data or information that we use inside the function to perform a specific task.

Generally, parameters are a list of local variables that will be assigned with values when the function gets called. They are used to passing values to functions.

They can be zero, or more parameters, separated by commas inside the parentheses. In the above example, there are two parameters in the function definition.

If any function contains parameters, At the time of calling their function, we have to pass values to those parameters.

If not pass parameters values then we will get error like **TypeError**: functionName() missing 1 required positional argument: 'x'

4. Colon (:):

A colon indicates the end of function headers.

5. Docstring:

A **docstring** is a documentation string that is an optional component. We commonly use it to describe what the function does. We write it on the line next to the function header.

A **docstring** must enclose in triple quotes and can write up to in several lines. We can **access** it with: **function_name.__doc__.**

6. Statement(s):

The body of function consists of one or more valid statements. Each statement must be intended with the same indentation to form a block.

In general, 4 spaces of indentation are standard from the header line in Python. When the function gets called, the statements inside the function's body execute.

Once the function is invoked, the formal parameters inside the parentheses become arguments.

7. Return statement:

A return statement ends the function call and returns a value from a function back to the function calling code. It is optional. If we do not define return statement inside the function's body, the function returns the object 'None'.

After the return statement if we are creating any other function statements then those all statements are not executing. Because return keyword statement only last statement in every function body.

Rules for defining User defined Function in Python

There are the following important rules for defining the user-defined functions in Python that should you keep in your mind. They are as:

1. Functions must begin either with a letter or an underscore.
2. They should be lowercase.
3. They can comprise numbers but shouldn't begin with one.
4. Functions can be of any length.
5. The name of function should not be a keyword in Python.
6. We can use an underscore to separate more than one word. For example, function_name(), person_name(), etc. It improves the readability and maintains conventions.
7. There should not be any space between the two words.

===============   ===============   ==================

➢ **Note:**

- Function will not be executed automatically.
- Function will be executed when ever we make a function call.
- We can call one function for 'N' number of times.
- By using **def** keyword we can define the function in python.
- After creating **function_name** , we need to call this function with **function_name()** and required arguments for executing.

Syntax:-

```
def   function_name(parameters_lists):
      statements1
      statements2                # block / suite / scope
      -----------
      -----------
function_name(actual_arguments_list)
```

Write a function to add the given values and store result in a variable

```
def  add(a,b):
   return  a+b
```

```
x = add(10,20)
print(x)
```
Output: 30

Write a function to print a message?

```
def m1():
    print('hi')
m1()
```
Output : hi

Can we call one functionName from another functionName?

Yes, we can call one functionName inside another functionName in python.

For example:

```
def f1():
   print('I am from f1() function')

def f2():
   print('I am from f2() function - first line')
   f1()
   print('I am from f2() function - last line')

f2()
```
Output:

```
I am from f2() function - first line
I am from f1() function
I am from f2() function - last line
```

Can we call a functionName inside same functionName (It leads to Infinite function)

```
def f1():
   print('I am from f1() function')
   f1()
f1()
```
Output : RecursionError: maximum recursion depth exceeded

Can we assign a function result to a variable

Yes, we can assign

For example:

```
def addition(x,y):
    return x + y

result = addition(10,20)
print(result)
```

Output : 30

Can we use a function result in an expression

Yes , we can use function result in an expression.

For example:

```
def addition(x,y):
    return x + y

result = addition(10,20) * 2 + 10
print(result)
```

Output: 70