Python Programming

Functions

Agenda :

What are functions?

Why we use

Functions :

Function can be defined as a **named group of instructions** that accomplish a specific task when it is invoked.

Once defined, a function can be called repeatedly from different places of the program without writing all the codes of that function everytime, or it can be called from inside another function.

The use of function is one of the means to achieve **modularity** and **reusability**.

The Advantages of Function :

Increases readability, particularly for longer code as by using functions, the program is better organised and easy to understand.

Reduces code length as same code is not required to be written at multiple places in a program. This also makes debugging easier.

Increases reusability, as function can be called from another function or another program. Thus, we can reuse or build upon already defined functions and avoid repetitions of writing the same piece of code.

Work can be easily divided among team members and completed in parallel.

Built in vs user defined functions :

Built in Functions :

The Python built-in functions are defined as the functions whose functionality is pre-defined in Python. The python interpreter has several functions that are always present for use. These functions are known as Built-in Functions.

User Defined Functions :

In addition to the standard library functions, we can define our own functions while writing the program. Such functions are called user defined functions.

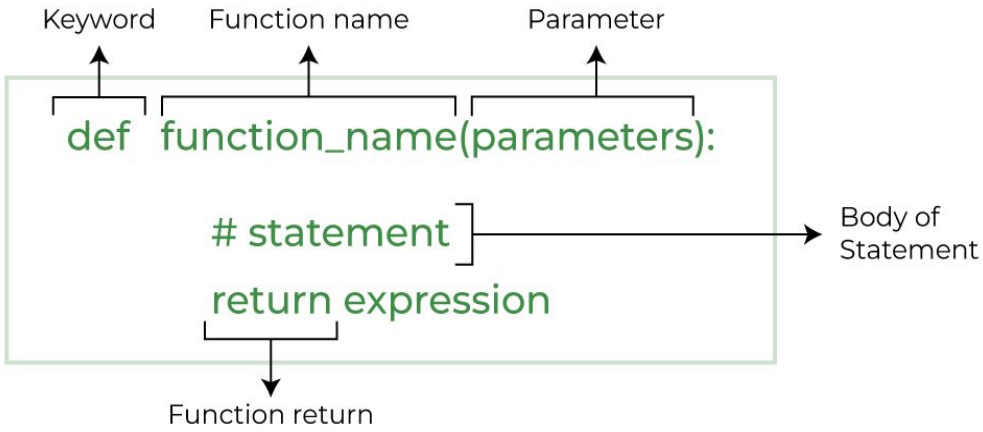
Function	Description
print()	Displayed function.
input ()	Get value from user.
int()	Convert integer data type.
float()	Convert float data type.
str()	Convert string data type.
type()	Displayed types of variables.
round()	Returns the nearest integer to its input.
max()	Returns the maximum value in a list.
min()	Returns the minimum value in a list.
sum()	Returns the sum of value in a list.
pow()	Return the power of values.

Creating User Defined Function :

A function definition begins with def (short for define).

Syntax -

```
def function_name(parameters):
        ## statements
        return
```



Source : geeksforgeeks

- Function header always ends with a colon (:).
- Function name should be unique.
- Rules for naming identifiers also applies for function naming.
- The statements outside the function indentation are not considered as part of the function.
- In order to execute the function, we need to call it.
- The function can be called in the program by writing function name followed by ()

Arguments and Parameters :

An argument is a value passed to the function during the function call which is received in corresponding parameter defined in function header.

```
#function header
def sumSquares(n): #n is the parameter
        sum = 0
        for i in range(1,n+1):
                sum = sum + i
                print("The sum of first",n,"natural numbers is: ",sum)

num = int(input("Enter the value for n: "))
#num is an argument referring to the value input by the user
sumSquares(num) #function call
```

Default Parameter :

Python allows assigning a default value to the parameter.

A default value is a value that is predecided and assigned to the parameter when the function call does not have its corresponding argument.

Eg -

```
def mixedFraction(num,deno = 1):
    print(num, deno)
```

- A function argument can also be an expression.

The parameters should be in the same order as that of the arguments.

The default parameters must be the **trailing** parameters in the function header that means if any parameter is having default value then all the other parameters to its right must also have default values

Functions Returning Value :

A function may or may not return a value when called.

The return statement returns the values from the function.

The function that does not return any value are called void function.

When we need to send value(s) from the function to its calling function. This is done using return statement.

The return statement does the following:

- returns the control to the calling function.
- return value(s) or None.

Flow of execution for functions:

When the interpreter encounters a function definition, the statements inside the function are not executed until the function is called.

when the interpreter encounters a function call, there is a little deviation in the flow of execution.

In that case, instead of going to the next statement, the control jumps to the called function and executes the statement of that function.

After that, the control comes back the point of function call so that the remaining statements in the program can be executed.

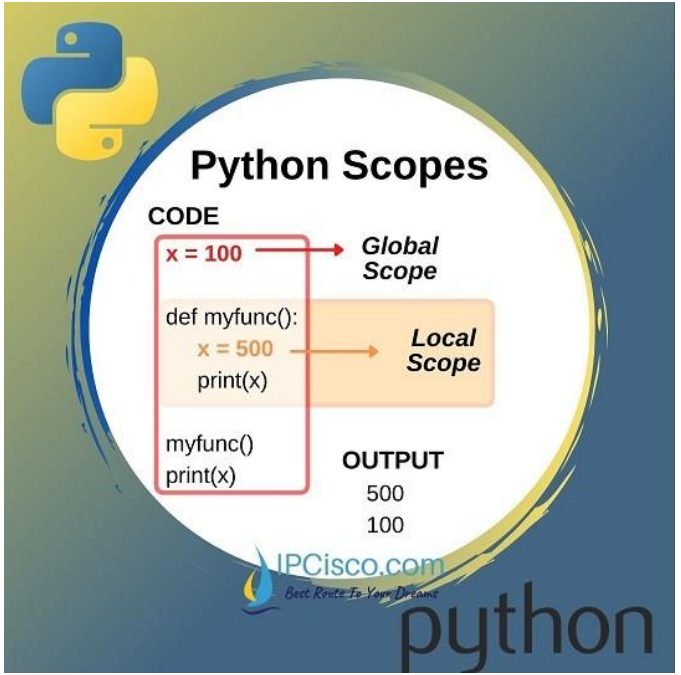
Scope of a Variable :

Global Variable :

In Python, a variable that is defined outside any function or any block is known as a global variable. It can be accessed in any functions defined onwards.

Local Variable :

A variable that is defined inside any function or a block is known as a local variable. It can be accessed only in the function. It exists only till the function executes.



Modules :

Other than the built-in functions, the Python standard library also consists of a number of modules.

While a function is a grouping of instructions, a module is a grouping of functions.

Suppose we have created some functions in a program and we want to reuse them in another program. In that case, we can save those functions under a module and reuse them.

Built in Modules : Python library has many built-in modules that are really handy to programmers. For eg - math, random, os

Using a module :

To use a module, we need to import the module. Once we import a module, we can directly use all the functions of that module.

Syntax - import modulename1 [,modulename2, ...]

To call a function of a module, the function name should be preceded with the name of the module with a dot(.) as a separator.

Syntax - modulename.functionname()

To import particular functions -

from modulename import functionname

Thank You