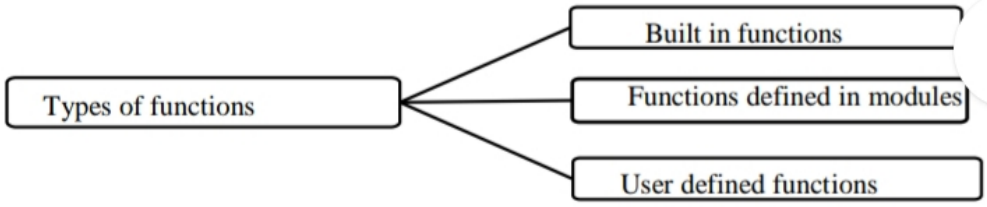
CHAPTER-2 FUNCTIONS IN PYTHON

Learning Outcomes: Understand the concept of functions in Python.

Definition: Functions are the subprograms that perform specific task. Functions are the small modules.

Types of Functions:

There are **three** types of functions in python:



Library Functions: These functions are already built in the python library.
Functions defined in modules: These functions defined in particular modules. When you want to use these functions in program, you have to import the corresponding module of that function.
User Defined Functions: The functions those are defined by the user are called user defined functions.

Library Functions in Python:

These functions are already built in the library of python.
For example: type( ), len( ), input( ) etc.

Functions defined in modules:

Functions of math module:

To work with the functions of math module, we must import math module in program.

import math

S. No.	Function	Description	Example
1	sqrt( )	Returns the square root of a number	>>>math.sqrt(49) 7.0
2	ceil( )	Returns the upper integer	>>>math.ceil(81.3) 82
3	floor( )	Returns the lower integer	>>>math.floor(81.3) 81
4	pow( )	Calculate the power of a number	>>>math.pow(2,3) 8.0
5	fabs( )	Returns the absolute value of a number	>>>math.fabs(-5.6) 5.6
6	exp( )	Returns the e raised to the power i.e. e^3	>>>math.exp(3) 20.085536923187668

Function in **random** module:

randint( )- function generates the random integer values including start and end values.
Syntax: randint(start, end)
It has two parameters. Both parameters must have integer values.
Example:
import random n=random.randint(3,7)

*The value of n will be 3 to 7.

User defined functions:

The syntax to define a function is: **def function-name (parameters) :**
#statement(s)

Where:

Keyword **def** marks the start of function header.
A function name to uniquely identify it. Function naming follows the same rules of writing identifiers in Python.
Parameters (arguments) through which we pass values to a function. They are optional.
A colon (**:**) to mark the end of function header.
One or more valid python statements that make up the function body. Statements must have same indentation level.
An optional return statement to return a value from the function.

Example:

```
def display(name):
        print("Hello  " +  name   + "  How are you?")
```

Function Parameters:

A functions has two types of parameters:
Formal Parameter: Formal parameters are written in the function prototype and function header of the definition. Formal parameters are local variables which are assigned values from the arguments when the function is called.
Actual Parameter: When a function is *called*, the values that are passed in the call are called *actual parameters*. At the time of the call each actual parameter is assigned to the corresponding formal parameter in the function definition.
Default Parameters: Python allows function arguments to have default values. If the function is called without the argument, the argument gets its default value.

Example :

```
def ADD(x, y):                  #Defining a function and x and y are formal parameters
        z=x+y
        print("Sum = ", z)
a=float(input("Enter first number: " ))
b=float(input("Enter second number: " ))
ADD(a,b)       #Calling the function by passing actual parameters
```

In the above example, **x** and **y** are formal parameters. **a** and **b** are actual parameters.

Calling the function:

Once we have defined a function, we can call it from another function, program or even the Python prompt. To call a function we simply type the function name with appropriate parameters.

Syntax:

function-name(parameter)

Example:

ADD(10,20)

OUTPUT:

Sum = 30.0

The return statement:

The **return** statement is used to exit a function and go back to the place from where it was called.
There are two types of functions according to return statement:

a. Function returning some value
b. Function not returning any value

a. Function returning some value :

Syntax:

return expression/value

Example-1: Function returning one value

```
def my_function(x):
        return 5 * x
```

Example-2 Function returning multiple values:

```
def sum(a,b,c):
        return a+5, b+4, c+7
S=sum(2,3,4)          # S will store the returned values as a tuple
print(S)
```

OUTPUT: (7, 7, 11)

Example-3: Storing the returned values separately:

```
def sum(a,b,c):
        return a+5, b+4, c+7
s1, s2, s3=sum(2, 3, 4)          # storing the values separately
print(s1, s2, s3)
```

OUTPUT:

7 7 11

b. Function not returning any value : The function that performs some operations but does not return any value, called void function.

```
def message():
    print("Hello")

m=message()
print(m)
```

OUTPUT: Hello

None

Scope and Lifetime of variables:

Scope of a variable is the portion of a program where the variable is recognized. Parameters and variables defined inside a function is not visible from outside. Hence, they have a local scope.

There are two types of scope for variables:

i) Local Scope
ii) Global Scope

Local Scope: Variable used inside the function. It cannot be accessed outside the function. In this scope, the lifetime of variables inside a function is as long as the function executes. They are destroyed once we return from the function. Hence, a function does not remember the value of a variable from its previous calls.

Global Scope: Variable can be accessed outside the function. In this scope, Lifetime of a variable is the period throughout which the variable exits in the memory.

Example:

```
def my_func():    x = 10
                  print("Value inside function:",x)
x = 20
 my_func()
print("Value outside function:",x)
```

OUTPUT:

Value inside function: 10
Value outside function: 20

Here, we can see that the value of **x** is **20** initially. Even though the function my_func()changed the value of **x** to **10**, it did not affect the value outside the function.

This is because the variable **x** inside the function is different (local to the function) from the one outside. Although they have same names, they are two different variables with different scope.

On the other hand, variables outside of the function are visible from inside. They have a **global** scope.

We can read these values from inside the function but cannot change (write) them. In order to modify the value of variables outside the function, they must be declared as global variables using the keyword **global**.

Very Short Answer Type Questions (1-Mark)

Q1. What is default parameter?

Ans: A parameter having default value in the function header is known as a default parameter.

Q2. Can a function return multiple values in python?

Ans: YES.

Short Answer Type Questions (2-Marks)

Q1. Rewrite the correct code after removing the errors: -

```
def SI(p,t=2,r):
    return (p*r*t)/100
```

Ans: -

```
def SI(p, r, t=2):
    return(p*r*t)/100
```

Q2. Consider the following function headers. Identify the correct statement: -
1) def correct(a=1,b=2,c): 2) def correct(a=1,b,c=3):
3) def correct(a=1,b=2,c=3): 4) def correct(a=1,b,c):
Ans: - 3) def correct(a=1,b=2,c=3)

Q3.What Will be the output of the following code?

```
a=1
def f():
    a=10
print(a)
```

Ans: The code will print 1 to the console.

Application Based Questions (3 Marks)

Q1. Write a python program to sum the sequence given below. Take the input ***n*** from the user.

$$1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{n!}$$

Solution:

```
def fact(x):
 j=1   ,  res=1
        while j<=x:
                res=res*j
                 j=j+1
    return res
n=int(input("enter the number : "))
 i=1, sum=1
while i<=n:
        f=fact(i)
        sum=sum+1/f
        i+=1
print(sum)
```

Q2. Write a program to compute GCD and LCM of two numbers

```
def gcd(x,y):
        while(y):
                x, y = y, x % y
        return x
def lcm(x, y):
        lcm = (x*y)//gcd(x,y)
        return lcm
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
print("The L.C.M. of", num1,"and", num2,"is", lcm(num1, num2))
print("The G.C.D. of", num1,"and", num2,"is", gcd(num1, num2))
```

Q3. How many values a python function can return? Explain how?
Ans: Python function can return **more than one** values.

```
        def square_and_cube(X):
                return X*X, X*X*X, X*X*X*X
        a=3
        x,y,z=square_and_cube(a)
        print(x,y,z)
```

Q4. Find the output of the following code: -
Ans: -

```
    def CALLME(n1=1,n2=2):
            n1=n1*n2
            n2+=2
            print(n1,n2)
    CALLME()
    CALLME(2,1)
```

Q2. Write a program to compute GCD and LCM of two numbers

```
def gcd(x,y):
        while(y):
                x, y = y, x % y
        return x
def lcm(x, y):
        lcm = (x*y)//gcd(x,y)
        return lcm
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
print("The L.C.M. of", num1,"and", num2,"is", lcm(num1, num2))
print("The G.C.D. of", num1,"and", num2,"is", gcd(num1, num2))
```

Q3. How many values a python function can return? Explain how?

Ans: Python function can return **more than one** values.

```
        def square_and_cube(X):
                return X*X, X*X*X, X*X*X*X
        a=3
        x,y,z=square_and_cube(a)
        print(x,y,z)
```

Q4. Find the output of the following code: -

Ans: -

```
def CALLME(n1=1,n2=2):
        n1=n1*n2
        n2+=2
        print(n1,n2)
CALLME()
CALLME(2,1)
```