Python Functions

Functions in Python can be defined as lines of codes that are built to create a specific task and can be used again and again in a program when called.

There are two types of functions in the Python language:

- Built-in functions
- User-defined functions - Functions that are created by the programmer to meet the requirements.

We have used a lot of built-in functions in our code till now, these functions include print(), or sum(), etc. So, we have a good idea about how to call a function. Built-in functions are already present in our python program, and we just have to call them whenever we need them to execute.

Defining a Function

A Function defined in Python should follow the following format:

1. Keyword **def** is used to start and declare a function. **def** specifies the starting of function block.
2. **def** is followed by function-name followed by parenthesis.
3. Parameters are passed inside the parenthesis. At the end a colon is marked.

Syntax

def <function_name>(parameters):

    BODY

4. Python code requires indentation (space) of code to keep it associate to the declared block.

How to run/invoke a function

To execute a function it needs to be called. This is called function calling.

Function Definition provides the information about function name, parameters and the definition what operation is to be performed. In order to execute the function definition, we need to call the function.

Forms of Function

There are 4 forms of Function:

1. Without Parameter and without Return
2. With Parameter and without Return
3. Without Parameter and with Return
4. With Parameter and with Return

Without Parameter and without Return

Simple program to print statement using Function

```
def myfunction():
    print("I love Python")

print("Hello World!!")
myfunction()

# -----OUTPUT-----
Hello World!!
I love Python
```

With Parameter and without Return

Simple program to calculate sum of two number and print the sum using Function

```
def myadd(a, b):
    c = a+b
    print("Addition = %d" % (c))


print("Hello World!!")
x = 43
y = 55
myadd(x, y)

# -----OUTPUT-----
Hello World!!
Addition = 98
```

Without Parameter and with Return

Simple program that will return value.

```
def myname():
    a = "My name is Rahul"
    return a


name = myname()
print(name)

# -----OUTPUT-----
My name is Rahul
```

With Parameter and with Return

Simple program to calculate sum of two number and return the sum using Function

```
def myadd(a, b):
    c = a+b
    return c


x = 43
y = 55
ans = myadd(x, y)
print("Answer =", ans)


# -----OUTPUT-----
Answer = 98
```

There can be two types of data passed in the function.

1. The First type of data is the data passed in the function call. This data is called “arguments”.
2. The second type of data is the data received in the function definition. This data is called ‘parameters’.

Arguments can be literals, variables and expressions. Parameters must be variable to hold incoming values.

Alternatively, arguments can be called as actual parameters or actual arguments and parameters can be called as formal parameters or formal arguments.

Passing Parameters

Apart from matching the parameters, there are other ways of matching the parameters.

Python supports following types of formal argument:

1. Positional argument (Required argument).
2. Default argument
3. Keyword argument (Named argument)

Positional/Required Arguments

When the function call statement must match the number and order of arguments as defined in the function definition. It is Positional Argument matching.

Example

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


x = 15
addition(x, 10)
addition(x, x)
y = 20
addition(x, y)

# This will give us error
addition(x,y,34)

# -----OUTPUT-----
25
30
35
Traceback (most recent call last):
  File "test.py", line 12, in <module>
    addition(x,y,34)
TypeError: addition() takes 2 positional arguments but 3 were given
```

Default Arguments

Default Argument is the argument which provides the default values to the parameters passed in the function definition, in case value is not provided in the function call default value is used.

```
def msg(Id, Name, Age=21):
    "Printing the passed value"
    print(Id)
    print(Name)
    print(Age)
    return


msg(Id=100, Name='Ravi', Age=20)
msg(Id=101, Name='Ratan')

# -----OUTPUT-----
100
Ravi
20
101
Ratan
21
```

Explanation:

1. In first case, when msg() function is called passing three different values i.e., 100 , Ravi and 20, these values will be assigned to respective parameters and thus respective values will be printed.
2. In second case, when msg() function is called passing two values i.e., 101 and Ratan, these values will be assigned to Id and Name respectively. No value is assigned for third argument via function call and hence it will retain its default value i.e, 21.

Keyword Arguments

Using the Keyword Argument, the argument passed in function call is matched with function definition on the basis of the name of the parameter.

```
def msg(id, name):
    "Printing passed value"
    print(id)
    print(name)
    return


msg(id=100, name='Raj')
msg(name='Rahul', id=101)

# -----OUTPUT-----
100
Raj
101
Rahul
```

Explanation:

1. In the first case, when msg() function is called passing two values i.e., id and name the position of parameter passed is same as that of function definition and hence values are initialized to respective parameters in function definition. This is done on the basis of the name of the parameter.
2. In second case, when msg() function is called passing two values i.e., name and id, although the position of two parameters is different it initialize the value of id in Function call to id in Function Definition. Same with name parameter. Hence, values are initialized on the basis of name of the parameter.

Anonymous Functions

Anonymous Functions are the functions that are not bond to name. It means anonymous function does not has a name.

Anonymous Functions are created by using a keyword "lambda".

Lambda takes any number of arguments and returns an evaluated expression.

Lambda is created without using the def keyword.

Syntax:-

ambda arg1,args2,args3,?,argsn :expression

Example to calculate square with both Function and Anonymous Function

```
def square(a):
    return a*a

print("Square of 10 using Square Function =", square(10))


square2=lambda x:x*x

print("Square of 10 using Lambda Function =", square2(10))


# -----OUTPUT-----
Square of 10 using Square Function = 100
Square of 10 using Lambda Function = 100
```

Scope of Variables

Scope of a variable can be determined by the part in which variable is defined. Each variable cannot be accessed in each part of a program. There are two types of variables based on Scope:

1. Local Variable
2. Global Variable

Local Variables

Variables declared inside a function body is known as Local Variable. These have a local access thus these variables cannot be accessed outside the function body in which they are declared.

```
def my_func():
    a = 10
    print("Value of a is %d" % (a))

my_func()
# This will give us error
print(a)



# -----OUTPUT-----
Value of a is 10

Traceback (most recent call last):
  File "test.py", line 7, in <module>
    print(a)
NameError: name 'a' is not defined
```

Global Variables

Variable defined outside the function is called Global Variable. Global variable is accessed all over program thus global variable have widest accessibility.

```
b=20

def my_func():
    a = 10
    print("Value of a is %d" % (a))
    print("Value of b is %d" % (b))

my_func()
print(b)



# -----OUTPUT-----
Value of a is 10
Value of b is 20
20
```

Questions

1. Create a function to add to 2 numbers using:-
    a. without return type - without parameter
    b. without return type - with parameter
    c. with return type - without parameter
    d. with return type - with parameter
2. Function to Calculate Circumference of Circle
3. Function to Calculate Area of Rectangle
4. Write a program that asks the user for an integer number and find the sum of all natural numbers upto that number in a function and check that sum is even or odd in another function.
5. Create two lists and find their Products and sum in separate functions.
6. Take a String and reverse it.
7. Create a dictionary and store Name, Marks in 5 subjects and Total in that Dictionary. Create a function which calculates Highest Marks scored by a Student by passing that dictionary into that function.
8. Create a function to Replace all the Spaces in sentence with “@”.
9. Create a function that accepts 2 Lists and returns the addition of both list.