FUNCTIONS IN PYTHON

A function is a set of statements that take inputs, do some specific computation and produces output. The idea is to put some commonly or repeatedly done task together and make a function, so that instead of writing the same code again and again for different inputs, we can call the function.
Python provides built-in functions like print(), etc. but we can also create your own functions. These functions are called user-defined functions.

```
# 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

evenOdd(2)

evenOdd(3)
```

Output:

```
even

odd
```

Pass by Reference or pass by value?

One important thing to note is, in Python every variable name is a reference. When we pass a variable to a function, a new reference to the object is created. Parameter passing in Python is same as reference passing in Java.

```
# Here x is a new reference to same list lst
def myFun(x):
x[0] = 20


# Driver Code (Note that lst is modified
# after function call.
lst = [10, 11, 12, 13, 14, 15]
myFun(lst);
print(lst)
```

Output:

```
[20, 11, 12, 13, 14, 15]
```

When we pass a reference and change the received reference to something else, the connection between passed and received parameter is broken. For example, consider below program.

```
def myFun(x):


# After below line link of x with previous
# object gets broken. A new object is assigned
# to x.
x = [20, 30, 40]
```

```
# Driver Code (Note that lst is not modified
# after function call.
lst = [10, 11, 12, 13, 14, 15]
myFun(lst);
print(lst)
```

Output:

```
[10, 11, 12, 13, 14, 15]
```

Another example to demonstrate that reference link is broken if we assign a new value (inside the function).

```
def myFun(x):

# After below line link of x with previous
# object gets broken. A new object is assigned
# to x.
x = 20

# Driver Code (Note that lst is not modified
# after function call.
x = 10
myFun(x);
print(x)
```

Output:

```
10
```

Exercise: Try to guess the output of following code.

```
def swap(x, y):
        temp = x;
        x = y;
        y = temp;


# Driver code
x = 2
y = 3
swap(x, y)
print(x)
print(y)
```

Output:

```
2
3
```

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)
```

Like C++ default arguments, any number of arguments in a function can have a default value. But once we have a default argument, all the arguments to its right must also have default values.

Keyword arguments:

The idea is to allow caller to specify argument name with values so that caller does not need to remember order of parameters.

```
# Python program to demonstrate Keyword Arguments
def student(firstname, lastname):
    print(firstname, lastname)


# Keyword arguments
student(firstname ='Geeks', lastname ='Practice')
student(lastname ='Practice', firstname ='Geeks')
```

Output:

```
('Geeks', 'Practice')
('Geeks', 'Practice')
```

Variable Length arguments:

We can have both normal and keyword variable number of arguments. Please see this for details.

```
# Python program to illustrate
# *args for variable number of arguments
def myFun(*argv):
    for arg in argv:
        print (arg)

myFun('Hello', 'Welcome', 'to', 'GeeksforGeeks')
```

Output:

```
Hello
Welcome
to
GeeksforGeeks
```

Anonymous Functions:

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

```
# Python code to illustrate cube of a number
# using labmda function
```

```
cube = lambda x: x*x*x

print(cube(7))
```

Output:

```
343
```