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.

```
def evenOdd( x ):
    if (x % 2 == 0):
        print("even")
    else:
        print("odd")


# Calling function
evenOdd(2)
evenOdd(3)
```

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

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

```
def myFun(x):
  x = [20, 30, 40]

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

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)

# We call myFun() with only argument
myFun(10)
```

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')
```

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')


# Python program to illustrate *kargs for variable number of keyword arguments

def myFun(**kwargs):
    for key, value in kwargs.items():
        print ("%s == %s" %(key, value))

# calling function
myFun(first ='Geeks', mid ='for', last='Geeks')
```

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))
```

Example use with filter()

As the name suggests, it is used to filter the iterables (Lists, tuples, dictionaries, and sets are all iterable objects.) as per the conditions. Filter filters the original iterable and passes the items that returns True for the function provided to filter. Therefore only the items in the iterables can be expected to be seen in the output.

The filter() function in Python takes in a function and a list as arguments.
The function is called with all the items in the list and a new list is returned which contains items for which the function evaluates to True.
Here is an example use of filter() function to filter out only even numbers from a list.

```
# Program to filter out only the even items from a list
my_list = [1, 5, 4, 6, 8, 11, 3, 12]

new_list = list(filter(lambda x: (x%2 == 0) , my_list))

print(new_list)
```

Example use with map()

The basic function of map() is to manipulate iterables. Map executes all the conditions of a function on the items in the iterable. Map function takes all elements and allows you to apply a function on it and then passes it to the output which can have same as well as different values .

The map() function in Python takes in a function and a list.
The function is called with all the items in the list and a new list is returned which contains items returned by that function for each item.

Here is an example use of map() function to double all the items in a list.

```
# Program to double each item in a list using map()

my_list = [1, 5, 4, 6, 8, 11, 3, 12]

new_list = list(map(lambda x: x * 2 , my_list))

print(new_list)
```

Passing Multiple arguments

```
>>> def add(x, y):
...     """Return x plus y"""
...     return x + y
...
>>>
>>> add(4,5)
9
>>> add(8,3)
11
>>>
```

It is possible to return multiple values from a function in the form of tuple, list, dictionary.

Return as tuple

```
>>> def function():
    a=10; b=10
```

```
    return a,b

>>> x=function()
>>> type(x)
<class 'tuple'>
>>> x
(10, 10)
>>> x,y=function()
>>> x,y
(10, 10)
```

Return as list

```
>>> def function():
    a=10; b=10
    return [a,b]

>>> x=function()
>>> x
[10, 10]
>>> type(x)
<class 'list'>
```

Return as dictionary

```
>>> def function():
    d=dict()
    a=10; b=10
    d['a']=a; d['b']=b
    return d

>>> x=function()
>>> x
{'a': 10, 'b': 10}
>>> type(x)
<class 'dict'>
```

https://www.w3resource.com/python-exercises/python-functions-exercises.php