+

**Tutorial No. 5**

**Title: Program on Functional Programming: lambda, map, filter and reduce**

**Batch: B1 Roll No: 1914078 Tutorial No.:5**

**Aim:** **Program on Functional Programming: lambda, map, filter and reduce**

**Resources needed:** Python IDE

### Theory:

# Anonymous Function

# In Python, anonymous function means that a function is without a name. As we know that *def* keyword is used to define the normal functions and the *lambda* keyword is used to create anonymous functions.

**Syntax:**

Lamda arguments: expressions

**Lambda functions can be used along with built-in functions like filter(), map() and reduce().**

**Use of lambda() with map()**

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

Example:

li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]

final\_list = list(map(lambda x: x\*2 , li))

print(final\_list)

**Use of lambda() with reduce()**

The reduce() function in Python takes in a function and a list as argument. The function is called with a lambda function and a list and a new reduced result is returned. This performs a repetitive operation over the pairs of the list. This is a part of functools module.

Example:

|  |
| --- |
| from functools import reduce  li = [5, 8, 10, 20, 50, 100]  sum = reduce((lambda x, y: x + y), li)  print (sum) |

**Use of lambda() with reduce()**

The reduce() function in Python takes in a function and a list as argument. The function is called with a lambda function and a list and a new reduced result is returned. This performs a repetitive operation over the pairs of the list. This is a part of functools module.

Example:

|  |
| --- |
| from functools import reduce  li = [5, 8, 10, 20, 50, 100]  sum = reduce((lambda x, y: x + y), li)  print (sum) |

**range()function** accepts an integer and returns a range object, which is nothing but a sequence of integers, range() takes three arguments Out of the three 2 arguments are optional. I.e., Start and Step are the optional arguments.

**Syntax:** range (start, stop[, step]).

**Parameters**

* A start argument is a starting number of the sequence. i.e., lower limit. By default, it starts with 0 if not specified.
* A stop argument is an upper limit. i.e.generate numbers up to this number, The range()  function doesn’t include this number in the result.
* The step is a difference between each number in the result. The default value of the step is 1 if not specified.

**Factory function**

A factory function is basically used to return (new) objects. Factory function provides encapsulation, that means that the code is not tied to specific classes or objects, and thus the class hierarchy or prototypes can be changed or [refactored](https://en.wikipedia.org/wiki/Refactored) without needing to change code that uses them – they abstract from the class hierarchy or prototypes.

**Factory method can be used in following scenarios**:

* A user may click on a certain button that creates an object.
* A user may create several new documents of different types.
* If a user starts a web browser, the browser does not know in advance how many tabs (where every tab is an object) will be opened

**Syntax :**

**def** fun(N):

**def** action(X):

**return** X \* N

**return** action

### Activity

a) Write a program which can map() and filter() to make a list whose

elements are square of even number in [1,2,3,4,5,6,7,8,9,10].  
b) WAP to find sum of squares of even numbers using reduce().

**Program:**

# A

import functools as f

mlist = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

def oddEven(num):

    if(num % 2 == 0):

        return True

    else:

        return False

def sumOfSquares(num1, num2):

    return num1 + num2\*\*2

nlist = list(filter(oddEven, mlist))

flist = list(map(lambda x: x\*\*2, nlist))

print(flist)

# B

nlist = f.reduce(sumOfSquares,

                 [num for num in mlist if oddEven(num)], 0)

print(nlist)

### Output:

### 

### Post Tutorial Questions:

### **What is the difference between range & xrange?**

range() – This returns a range object (a type of iterable).  
xrange() – This function returns the generator object that can be used to display numbers only by looping.

The variable storing the range created by range() takes more memory as compared to variable storing the range using xrange().

### Outcomes: Use of Different Decision Making Statements and Functions in Python

**Conclusion:** We learnt and implemented python map, reduce and filter functions

**References:**

1. Reema Thareja , *Python Programing: Using Problem Solving Approach,* Oxford University Press, First Edition 2017, India
2. Dr. R. Nageswara Rao, *Core Python Programming,* Wiley Publication, Second Edition 2018,India
3. Sheetal Taneja and Naveen Kumar, *Python Programing: A Modular Approach,* Pearson India
4. Swarroop C.H, *Byte of python,* e-book, Kindle edition
5. Martin C Brown, *The Complete Reference Python,* Brandon A Nordin, First Edition 2001