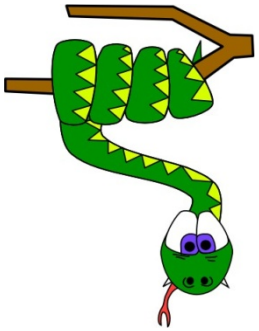


Higher Order Functions in Python
Part-II
KNC-402

MAP FILTER and REDUCE

- **map(), filter() and reduce()** are inbuilt functions of Python.

- The arguments passed to these HOFs are the only factors that decide upon the output. These functions can take any other function as a parameter and can be supplied to other functions as parameters as well.

The MAP Function

- Map function takes another function as a parameter along with a sequence of iterables and returns an output after applying the function to each iterable present in the sequence.

SYNTAX:

map(function, iterables)

Example:

```
def newfunc(a):
   return a*a
#x is the map object
x = map(newfunc, (1,2,3,4))
print(x)
print(set(x))
```

Output:

<map object at 0x0285CFE8>
{16, 1, 4, 9}

The map with lambda Function

Example:

```
tup= (5, 7, 22, 97, 54, 62, 77, 23, 73, 61)
newtuple = tuple(map(lambda x: x+3 , tup))
print(newtuple)
```

OUTPUT:

(8, 10, 25, 100, 57, 65, 80, 26, 76, 64)

New ..The map with lambda Function

```
# Add two lists using map and lambda

numbers1 = [1, 2, 3]
numbers2 = [4, 5, 6]

result = map(lambda x, y: x + y, numbers1, numbers2)
print(list(result))
```

OUTPUT:

[5, 7, 9]

The filter() function

- The filter() function is used to create an output list consisting of values for which the function returns true. The syntax of it is as follows:

SYNTAX:

filter(function, iterables)

#Example Program

```
def func(x):
    if x>=3:
        return x
y = filter(func, (1,2,3,4))
print(y)
print(list(y))
```

Output: <filter object at 0x00FA3E20>
[3, 4]

filter() with lambda function

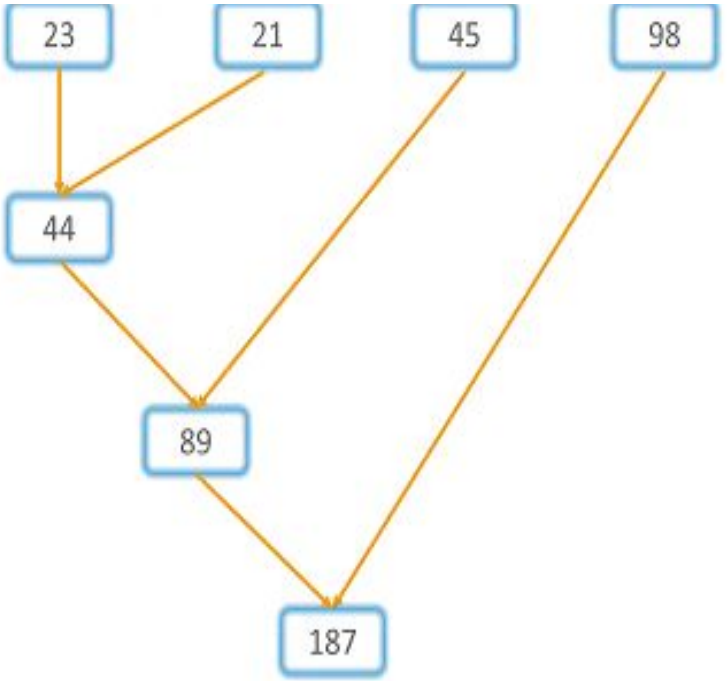
Example:

```
y = filter(lambda x: (x>=3), (1,2,3,4))
print(list(y))
```

OUTPUT: [3, 4]

The reduce() with lambda function

- The reduce() function, as the name describes, applies a given function to the iterables and returns a single value.



SYNTAX:
reduce(function, iterables)

Ex.

```
from functools import reduce
result=reduce(lambda a,b: a+b,[23,21,45,98])
print(result)
```

OUTPUT: 187

The reduce() with lambda function

Example:

```
from functools import reduce
result = reduce((lambda x, y: x * y), [1, 2, 3, 4,5])
print(result)
```

OUPUT:

120

Thumb rule HOF:

- *If you already have a list of values and you want to do the exact same operation on each of the elements in the array and return the same amount of items in the list, in these type of situations it is better to use the map method.*

- *If you already have list of values but you only want to have items in the array that match certain criteria, in these type of situations it is better to use the filter method.*

- *If you already have list of values, but you want to use the values in that list to create something completely new, in these type of situations it is better to use the reduce method.*