whenever we are working with data we need to modify data, filter data so we have in-build functions like map() filter() reduce()

map() function

The map function is used when you need to modify all elements with an iterables data

Syntax:

map(function, iterables)

Parameters:

function: The function to be called for each element of the specified iterable.

iterables: One or more iterables

Return Value:

When using map, it returns a map object, which is an iterator

```
Multiply without lambda
def d1(n1,n2):
    return n1*n2
print(d1(5,10)) # 50

Multiply lambda
d1 = lambda n1,n2:n1*n2
print(d1(5,10)) # 50
```

```
Multiply iterables
lst = [10,20,30,40,50,60,70]
print(lst*2) # [10, 20, 30, 40, 50, 60, 70, 10, 20, 30, 40, 50, 60, 70]

Multiply iterables using for loop
lst = [10,20,30,40,50,60,70]
l = []
r = lambda n: n*2
for i in lst:
    l.append(r(i))
print(l) # [20, 40, 60, 80, 100, 120, 140]

Multiply iterables with map function
lst = [10,20,30,40,50,60,70]
def d1(n):
    return n*2
m = list(map(d1, lst)) # map(func, iterables...)
print(m) # [20, 40, 60, 80, 100, 120, 140]

Multiply iterables with map function and lambda
lst = [10,20,30,40,50,60,70]
l = list(map(lambda n : n*2, lst)) # map(func, iterables...)
print(l) # [20, 40, 60, 80, 100, 120, 140]
```

```
# multiply sequence
l1 = [1,2,3,4]
l2 = [1,2,3,4]
print(l1*l2)  #TypeError: can't multiply sequence by non-int of type 'list'

# multiply sequence using map function
def d1(a,b):
   return a*b
x = map(d1, [1,2,3,4], [1,2,3,4]) # map(func, iterables...)
print(list(x))  # [1, 4, 9, 16]

# multiply sequence using map function and lambda
l1 = [1,2,3,4]
l2 = [1,2,3,4]
x = map(lambda a,b: a*b, l1,l2) # lambda args : expression, iterables
print(list(x))  # [1, 4, 9, 16]
```

```
# find the length of elements using forloop
lst = ["NameOne", "NameTwo", "NameThree", "NameFour"]
ls = []
result = lambda l:len(l)
for i in lst:
   ls.append(result(i))
print(ls) # [7, 7, 9, 8]

# find the length of elements using map
lst = ["NameOne", "NameTwo", "NameThree", "NameFour"]

def d1(n):
   return len(n)
result = map(d1, lst)  # map(func, iterables...)
print(list(result)) # [7, 7, 9, 8]

# find the length of elements using map and lambda
lst = ["NameOne", "NameTwo", "NameThree", "NameFour"]
result = map(lambda l: len(l), lst) # lambda args : expression, iterables
print(list(result)) # [7, 7, 9, 8]
```

```
# reverse list elemnets
lst = ["NameOne", "NameTwo", "NameThree", "NameFour"]
lst.reverse()
print(lst) # ['NameFour', 'NameThree', 'NameTwo', 'NameOne']

# reverse list elemnets
lst = ["NameOne", "NameTwo", "NameThree", "NameFour"]
print(lst[::-1]) # ['NameFour', 'NameThree', 'NameTwo', 'NameOne']

# reverse list elements
def d1(lst):
  return lst[::-1]
d = d1(["NameOne", "NameTwo", "NameThree", "NameFour"])
print(d) # ['NameFour', 'NameThree', 'NameTwo', 'NameOne']
```

```
#reverse list of strings using for loop
lst = ["NameOne", "NameTwo", "NameThree", "NameFour"]
ls = []
r = lambda l:l[::-1]
for i in lst:
   ls.append(r(i))
print(ls) # ['enOemaN', 'owTemaN', 'eerhTemaN', 'ruoFemaN']

#reverse list of strings using map function
def d1(lst):
  return lst[::-1]
d = (map(d1, ["NameOne", "NameTwo", "NameThree", "NameFour"]))
print(list(d)) # ['enOemaN', 'owTemaN', 'eerhTemaN', 'ruoFemaN']

#reverse list of strings using map function and lambda
lst = ["NameOne", "NameTwo", "NameThree", "NameFour"]
result = map(lambda l:l[::-1], lst)
print(list(result)) # ['enOemaN', 'owTemaN', 'eerhTemaN', 'ruoFemaN']
```