In [8]:
```
# map fuction takes 2 argument
# A function and any iterable data structure

data = [3,4,8]

#get a series of square of all elements

print(list(map(lambda x : x**2, data)))
# map function only maps the data set with function defined
# convert it as a readable list
```

```
[9, 16, 64]
```

In [10]:
```
# We can do it using conventional function as well

def squareit(x):
    return x**2

data = [4,6,12]

print(list(map(squareit, data)))
```

```
[16, 36, 144]
```

In [12]:
```
# sum of two lists

a = [2,3,4,4]
b = [3,4,5,6]

print(list(map(lambda x, y : x + y, a,b)))
# number of arguments of the function will be equal to number of iterable data
 lists
```

```
[5, 7, 9, 10]
```

In [14]:
```
first_name = ["nikola", "albert", "milan"]
last_name = [ "tesla", "einstain", "stephen"]

proper = lambda x, y : x[0].upper() + x [1:] + " " + y[0].upper() + y[1:]

print (list(map(proper, first_name, last_name)))
```

```
['Nikola Tesla', 'Albert Einstain', 'Milan Stephen']
```

In [18]:
```
## Filter create a subset of the list basis logical rule
# So function used will provide a logical value True / False

my_list = [3,42,42,4,9,9,36,4,3]

divby3 = lambda x : x % 3 == 0 #try != as well

div = filter(divby3, my_list)

print(list(div))
```

```
[3, 42, 42, 9, 9, 36, 3]
```

In [25]:
```
# Reduce
# Reduce is not an inbuilt function like map and filter
# It does aggregate operation in any data structure

from functools import reduce

q = reduce(lambda x,y: x+y, range(1,4))
# range gives 0,1,2,3
# lambda takes 0,1 and returns 1
# now reduce func works
# it takes first output as second input and so on till last number comes

print(q)
```

```
6
```

In [23]:
```
for i in range(1,4):
    print(i)
```

```
1
2
3
```

In [28]:
```
# Find the biggest number in a series

num =[3,47,23,1,44,3,220,1]

print(reduce(lambda x, y: x if x>y else y, num))
```

```
220
```

In [ ]:
```

```