Lab Manual Lecture 8

```
#Defining an Inner Function
'''def function1(): # outer function
    print ("Hello from outer function")
    def function2(): # inner function
        print ("Hello from inner function")
    function2()

function1()'''

##calling Inner Function without calling outer function
'''def function1(): # outer function
    print ("Hello from outer function")
    def function2(): # inner function
        print ("Hello from inner function")
    function2()'''
#function1()



#What if we attempt to change the variables of the outer function
from inside the inner function?
'''def function1(): # outer function
    x = 2 # A variable defined within the outer function
    def function2(a): # inner function
       # Let's define a new variable within the inner function
       # rather than changing the value of x of the outer function
        x = 6
        print (a+x)
    print (x) # to display the value of x of the outer function
    function2(3)

function1()'''
# another example
'''def num1(x):
   def num2(y):
      return x * y
   return num2
res = num1(10)

print(res(5))'''
#Nonlocal Variable in Function

'''def outer_func():
    x = 777

    def inner_func():
        # local variable now acts as global variable
        nonlocal x
        x = 700
        print("value of x inside inner function is :", x)

    inner_func()
    print("value of x inside outer function is :", x)

outer_func()'''
```

```
#Positional Arguments

'''def add(a, b, c):
    print(a - b + c)

add(50, 10, 50)
# Output 40
add(10, 50, 60)
# Output -40'''

#Keyword Arguments

'''def message(name, surname):
    print("Hello", name, surname)

message(name="John", surname="Wilson")
message(surname="Wilson", name="John")
message("Wilson", "John")
message("John", "Wilson")'''
#keyword and positional argument simultaneously
'''def message(first_nm, last_nm):
    print("Hello..!", first_nm, last_nm)

# correct use
message("John", "Wilson")
message("John", last_nm="Wilson")

# Error
# SyntaxError: positional argument follows keyword argument
message(first_nm="John", last_nm="Wilson")'''


#Default Arguments

# function with default argument

'''def message(name="Respected", surname="Guest"):
    print("Hello", name, surname)

# calling function with argument
message("John", "Wilson")

# calling function without argument
message()'''

#Variable length arguments


'''def addition(*numbers):
    total = 0
    for no in numbers:
        total = total + no
    print("Sum is:", total)


# 0 arguments
addition()

# 5 arguments
addition(10, 5, 2, 5, 4)
```

```
# 3 arguments
addition(78, 7, 2.5)'''

#Recursive Function

'''def factorial(no):
    if no == 0:
        return 1
    else:
        return no * factorial(no - 1)

print("factorial of a number is:", factorial(5))'''

#Example 1: Program for even numbers without lambda function
'''def even_numbers(nums):
    even_list = []
    for n in nums:
        if n % 2 == 0:
            even_list.append(n)
    return even_list

num_list = [10, 5, 12, 78, 6, 1, 7, 9]
ans = even_numbers(num_list)
print("Even numbers are:", ans)'''

#Example 2: Program for even number with a lambda function

l = [10, 5, 12, 78, 6, 1, 7, 9]
even_nos = list(filter(lambda x: x % 2 == 0, l))
print("Even numbers are: ", even_nos)

#Example: lambda function with  filter()

l = [-10, 5, 12, -78, 6, -1, -7, 9]
positive_nos = list(filter(lambda x: x > 0, l))
print("Positive numbers are: ", positive_nos)

#Example: lambda function with map() function

list1 = [2, 3, 4, 8, 9]
list2 = list(map(lambda x: x*x*x, list1))
print("Cube values are:", list2)

#Example: lambda function with reduce()

from functools import reduce
list1 = [20, 13, 4, 8, 9]
add = reduce(lambda x, y: x+y, list1)
print("Addition of all list elements is : ", add)
```