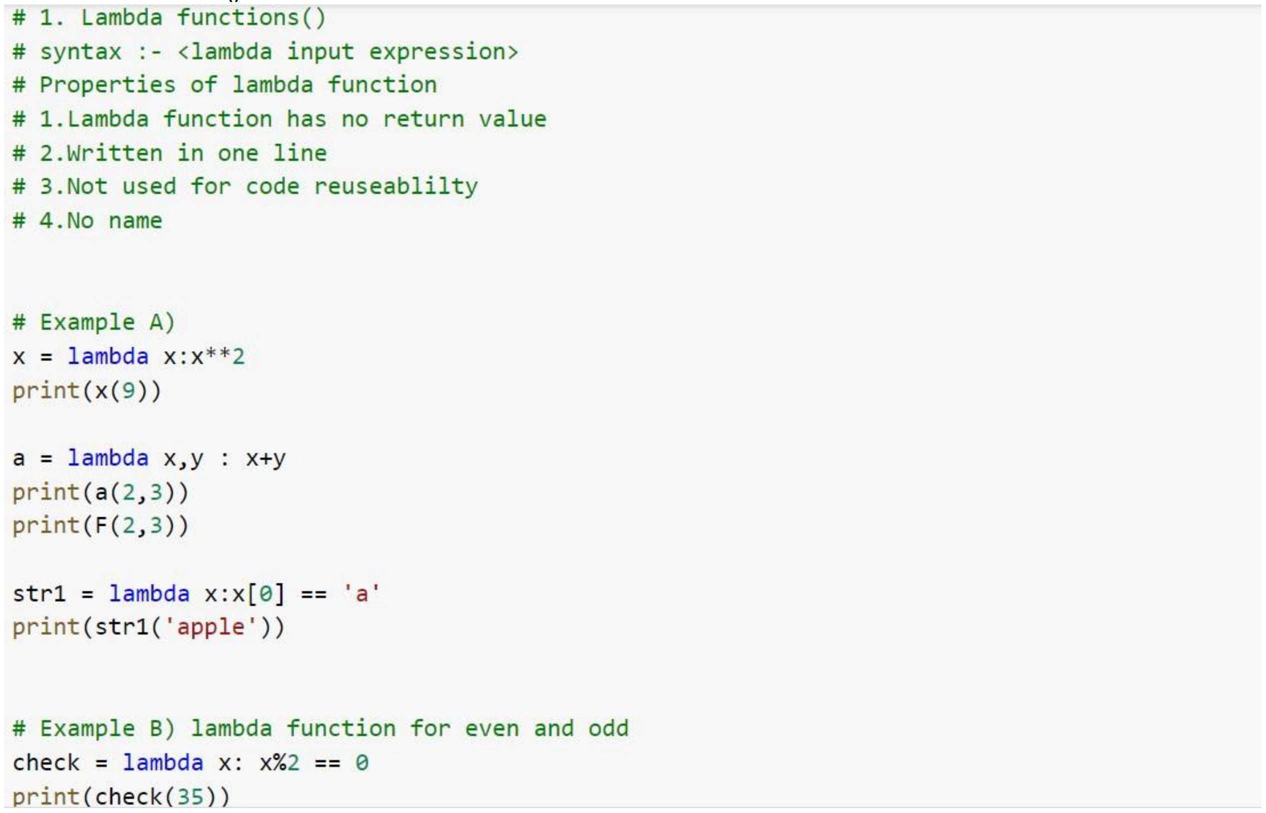
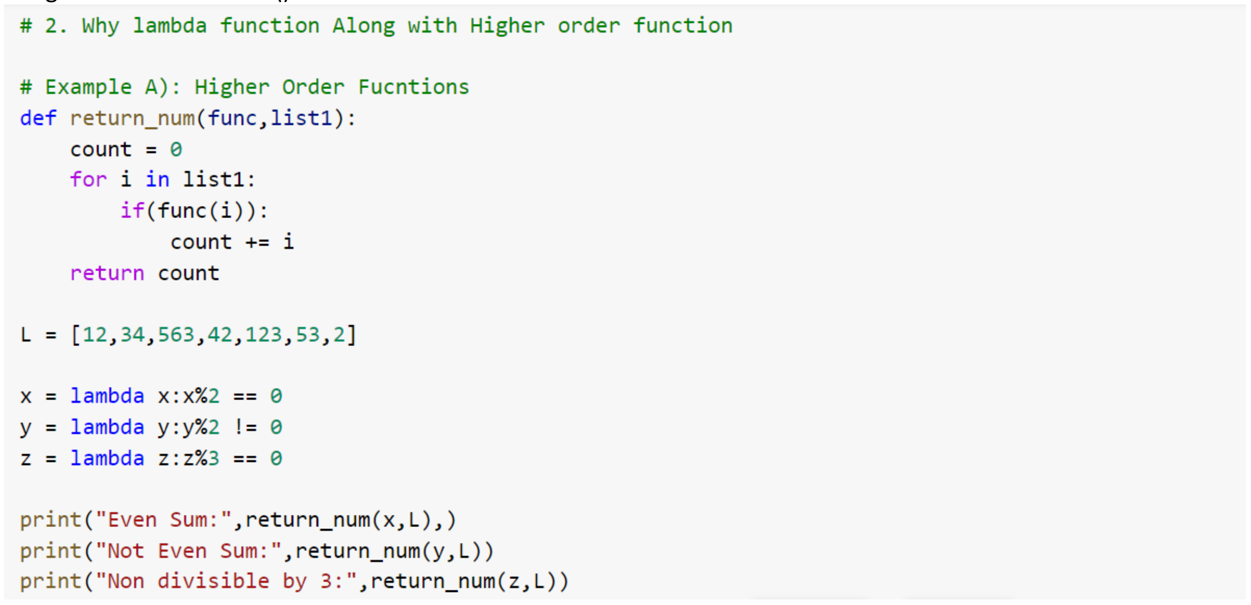
# Higher Order Functions

# Lambda Functions ()



```
# 1. Lambda functions()
# syntax :- <lambda input expression>
# Properties of lambda function
# 1.Lambda function has no return value
# 2.Written in one line
# 3.Not used for code reuseablilty
# 4.No name


# Example A)
x = lambda x:x**2
print(x(9))

a = lambda x,y : x+y
print(a(2,3))
print(F(2,3))

str1 = lambda x:x[0] == 'a'
print(str1('apple'))


# Example B) lambda function for even and odd
check = lambda x: x%2 == 0
print(check(35))
```

# Higher Order Functions()

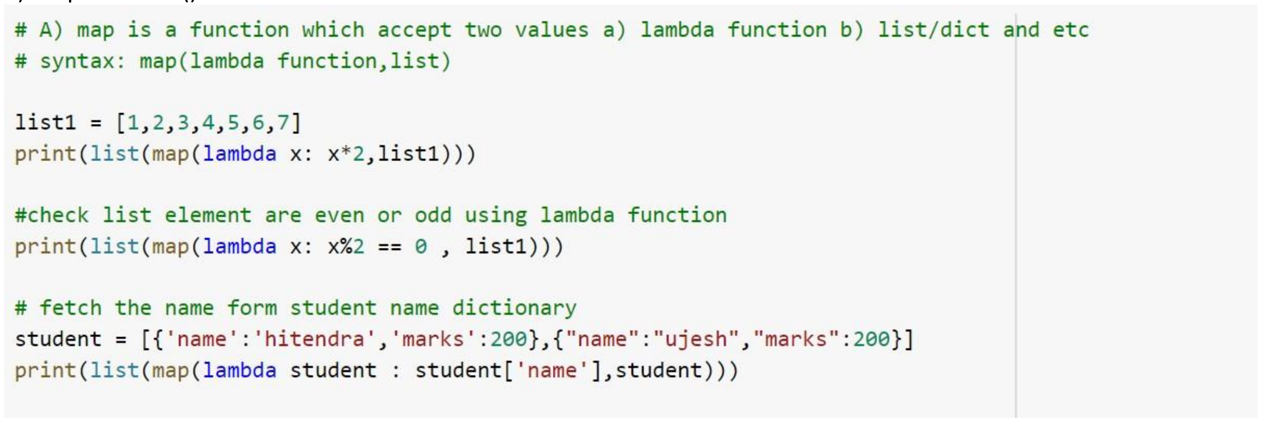


```
# 2. Why lambda function Along with Higher order function

# Example A): Higher Order Fucntions
def return_num(func,list1):
    count = 0
    for i in list1:
        if(func(i)):
            count += i
    return count

L = [12,34,563,42,123,53,2]

x = lambda x:x%2 == 0
y = lambda y:y%2 != 0
z = lambda z:z%3 == 0

print("Even Sum:",return_num(x,L),)
print("Not Even Sum:",return_num(y,L))
print("Non divisible by 3:",return_num(z,L))
```

Higher Order Functions

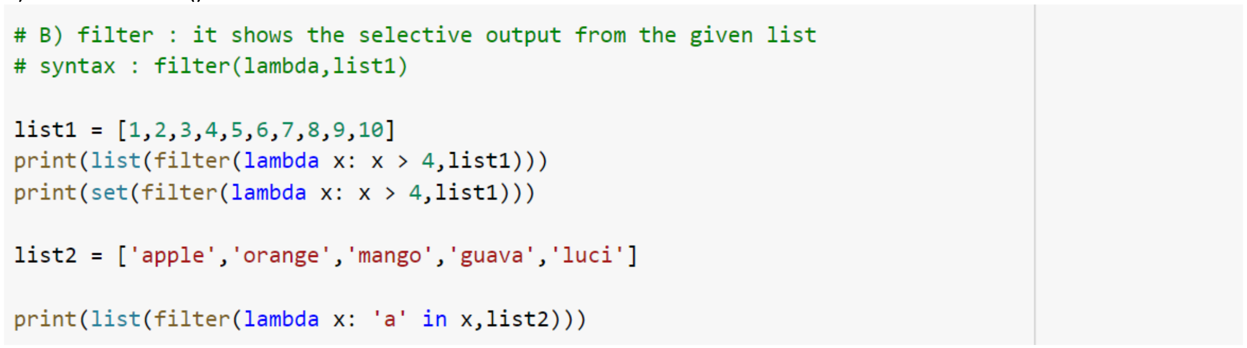
Types of Higher Order Function

1) Map Function()



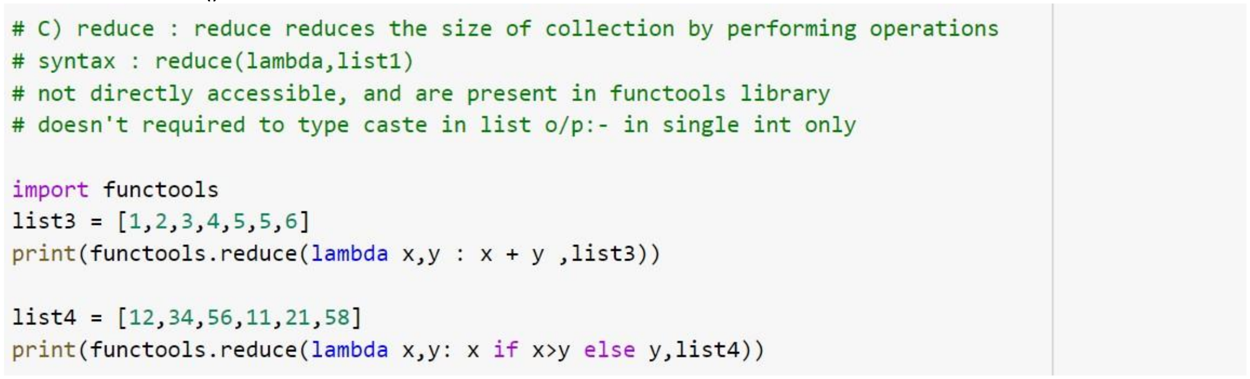
```python
# A) map is a function which accept two values a) lambda function b) list/dict and etc
# syntax: map(lambda function,list)

list1 = [1,2,3,4,5,6,7]
print(list(map(lambda x: x*2,list1)))

#check list element are even or odd using lambda function
print(list(map(lambda x: x%2 == 0 , list1)))

# fetch the name form student name dictionary
student = [{'name':'hitendra','marks':200},{"name":"ujesh","marks":200}]
print(list(map(lambda student : student['name'],student)))
```

2) Filter Function()



```python
# B) filter : it shows the selective output from the given list
# syntax : filter(lambda,list1)

list1 = [1,2,3,4,5,6,7,8,9,10]
print(list(filter(lambda x: x > 4,list1)))
print(set(filter(lambda x: x > 4,list1)))

list2 = ['apple','orange','mango','guava','luci']

print(list(filter(lambda x: 'a' in x,list2)))
```

# Reduce Function()



```python
# C) reduce : reduce reduces the size of collection by performing operations
# syntax : reduce(lambda,list1)
# not directly accessible, and are present in functools library
# doesn't required to type caste in list o/p:- in single int only

import functools
list3 = [1,2,3,4,5,5,6]
print(functools.reduce(lambda x,y : x + y ,list3))

list4 = [12,34,56,11,21,58]
print(functools.reduce(lambda x,y: x if x>y else y,list4))
```