**-: Practical Set – 6 :-**

1. Write a program to find the prime numbers in a specific range using filter.

import sympy

n = 10

l1 = list(range(1,n))

ans = list(filter(lambda x : sympy.isprime(x), l1))

print(ans)

**OUTPUT:**



1. Write a python program to make sum of particular range using reduce.

import functools

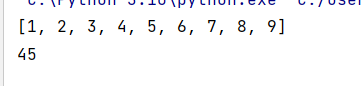
n = 10

l1 = list(range(1, n))

print(l1)

print(functools.reduce(lambda x, y: x+y, l1))

**OUTPUT:**



1. Write a python program to find maximum from a list using reduce.

import functools

l1 = [1,2,5,7,6,11,65,3,30,24]

print(functools.reduce(lambda x,y : x if x>y else y, l1))

**OUTPUT:**



1. Write a python program to find Armstrong number in a specific range using map.

def is\_armstrong(number):

num\_digits = len(str(number))

digit\_sum = sum(map(lambda x: int(x)\*\*num\_digits, str(number)))

return digit\_sum == number

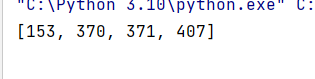
start = 100

end = 1000

armstrong\_numbers = list(filter(is\_armstrong, range(start, end+1)))

print(armstrong\_numbers)

**OUTPUT:**



1. Write a python program to apply two functions (square and cube) simultaneously on a specific range using map.

def square(x):

return x\*\*2

def cube(x):

return x\*\*3

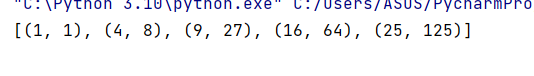
start = 1

end = 5

squared\_and\_cubed\_values = list(map(lambda x: (square(x), cube(x)), range(start, end+1)))

print(squared\_and\_cubed\_values)

**OUTPUT:**



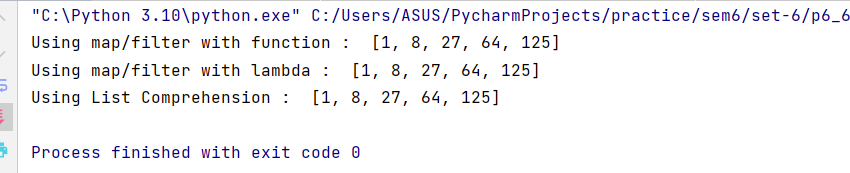
1. Write python programs using (i) map/filter and function (ii) map/filter and lambda (iii) list comprehension.

* Create a list to store the cube of all the elements in a given list.

def cube(num):  
 return num\*\*3

given\_list = [1, 2, 3, 4, 5]  
cubed\_list = list(map(cube, given\_list))  
print("Using map/filter with function : ", cubed\_list)  
cubed\_list = list(map(lambda x: x\*\*3, given\_list))  
print("Using map/filter with lambda : ", cubed\_list)  
cubed\_list = [num\*\*3 for num in given\_list]  
print("Using List Comprehension : ", cubed\_list)

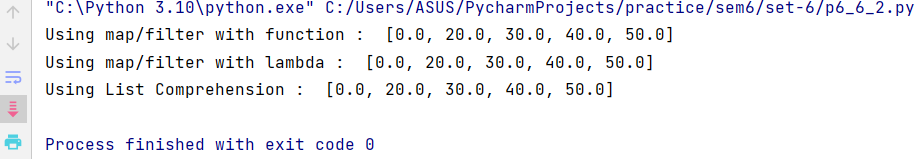
**OUTPUT :**



* Create a list of equivalent Celsius degree from Fahrenheit.

def f\_to\_c(deg\_f):  
 return (deg\_f - 32) \* 5/9  
  
deg\_f\_list = [32, 68, 86, 104, 122]  
deg\_c\_list = list(map(f\_to\_c, deg\_f\_list))  
print("Using map/filter with function : ", deg\_c\_list)  
deg\_c\_list = list(map(lambda f: (f - 32) \* 5/9, deg\_f\_list))  
print("Using map/filter with lambda : ", deg\_c\_list)  
deg\_c\_list = [(deg\_f - 32) \* 5/9 for deg\_f in deg\_f\_list]  
print("Using List Comprehension : ", deg\_c\_list)

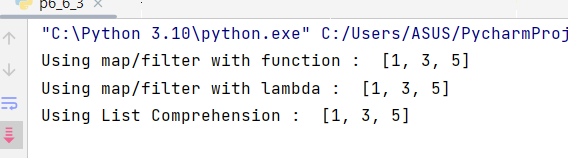
**OUTPUT :**



* Create a list that stores only positive numbers from given list.

def is\_positive(num):  
 return num > 0  
  
given\_list = [1, -2, 3, -4, 5]  
positive\_list = list(filter(is\_positive, given\_list))  
print("Using map/filter with function : ", positive\_list)  
  
positive\_list = list(filter(lambda x: x > 0, given\_list))  
print("Using map/filter with lambda : ", positive\_list)  
  
positive\_list = [num for num in given\_list if num > 0]  
print("Using List Comprehension : ", positive\_list)

**OUTPUT :**



* Create a list that stores only alphabets from given list

def is\_alpha(char):  
 return char.isalpha()  
  
given\_list = ['a', '1', 'B', '2', 'c', '3']  
alpha\_list = list(filter(is\_alpha, given\_list))  
print("Using map/filter with function : ", alpha\_list)  
  
alpha\_list = list(filter(lambda x: x.isalpha(), given\_list))  
print("Using map/filter with lambda : ", alpha\_list)  
  
alpha\_list = [char for char in given\_list if char.isalpha()]  
print("Using List Comprehension : ", alpha\_list)

**OUTPUT :**

