**Practical Set:- 06**

**Practical 01:-** Write a program to find the prime numbers in a specific range using filter.

Input:-

def is\_prime(n): if n < 2:

return False

for i in range(2, int(n \*\* 0.5) + 1):

if n % i == 0:

return False

return True

start = 10

end = 50

prime\_numbers = list(filter(is\_prime, range(start, end + 1)))

print(f"Prime numbers between {start} and {end}: {prime\_numbers}")

**Output:**

Prime numbers between 10 and 50: [11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]

**Practical 02:-** Write a python program to make sum of particular range using reduce.

Input:-

from functools import reduce

start = 1

end = 10

sum\_of\_range = reduce(lambda x, y: x + y, range(start, end + 1))

print(f"Sum of numbers between {start} and {end}: {sum\_of\_range}")

**Output:**

Sum of numbers between 1 and 10: 55

**Practical 03:-** Write a python program to find maximum from a list using reduce.

Input:-

from functools import reduce

numbers = [3, 41, 12, 9, 74, 15]

max\_value = reduce(lambda x, y: x if x > y else y, numbers)

print(f"The maximum value in the list is: {max\_value}")

**Output:**

The maximum value in the list is: 74

**Practical 04:-** Write a python program to find Armstrong number in a specific range using map.

Input:-

def is\_armstrong(num):

num\_str = str(num)

num\_len = len(num\_str)

return armstrong\_

sum == num

start = 100

end = 1000

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

print(f"Armstrong numbers between {start} and {end}: {armstrong\_numbers}")

**Output:**

Armstrong numbers between 100 and 1000: {153, 370, 371, 401}

**Practical 05:-** Write a python program to apply two functions (square and cube) simultaneously on a specific range using map.

Input:-

def square(n):

return n \*\* 2 def cube(n):

return n \*\* 3

start = 1

end = 5

result = list(map(lambda x: (square(x), cube(x)), range(start, end + 1)))

for i, (sq, cu) in enumerate(result, start=start):

print(f"Number: {i}, Square: {sq}, Cube: {cu}")

**Output:**

Number: 1, Square: 1, Cube: 1

Number: 2, Square: 4, Cube: 8

Number: 3, Square: 9, Cube: 27

Number: 4, Square: 16, Cube: 64

Number: 5, Square: 25, Cube: 125

**Practical 06:-** 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.
* Create a list of equivalent Celsius degree from Fahrenheit.
* Create a list that stores only positive numbers from given list.
* Create a list that stores only alphabets from given list.

numbers = [1, -2, 3, -4, 5, 6, -7, 8, 9, -10]

fahrenheit\_temps = [32, 50, 77, 100, 212]

mixed\_list = ['a', '1', 'b', 'c', '9', '@', 'd', '3']

def cube(n):

return n \*\* 3

cubes\_function = list(map(cube, numbers))

cubes\_lambda = list(map(lambda n: n \*\* 3, numbers))

cubes\_list\_comp = [n \*\* 3 for n in numbers]

def to\_celsius(f):

return (f - 32) \* 5 / 9

celsius\_function = list(map(to\_celsius, fahrenheit\_temps))

celsius\_lambda = list(map(lambda f: (f - 32) \* 5 / 9, fahrenheit\_temps))

celsius\_list\_comp = [(f - 32) \* 5 / 9 for f in fahrenheit\_temps]

def is\_positive(n):

return n > 0

positives\_function = list(filter(is\_positive, numbers))

positives\_lambda = list(filter(lambda n: n > 0, numbers))

positives\_list\_comp = [n for n in numbers if n > 0]

def is\_alpha(ch):

return ch.isalpha()

alphabets\_function = list(filter(is\_alpha, mixed\_list))

alphabets\_lambda = list(filter(lambda ch: ch.isalpha(), mixed\_list))

alphabets\_list\_comp = [ch for ch in mixed\_list if ch.isalpha()]

print("Cubes (Function):", cubes\_function)

print("Cubes (Lambda):", cubes\_lambda)

print("Cubes (List Comp):", cubes\_list\_comp)

print("\nCelsius (Function):", celsius\_function)

print("Celsius (Lambda):", celsius\_lambda)

print("Celsius (List Comp):", celsius\_list\_comp)

print("\nPositive Numbers (Function):", positives\_function)

print("Positive Numbers (Lambda):", positives\_lambda)

print("Positive Numbers (List Comp):", positives\_list\_comp)

print("\nAlphabets (Function):", alphabets\_function)

print("Alphabets (Lambda):", alphabets\_lambda)

print("Alphabets (List Comp):", alphabets\_list\_comp)

**Output:**

Cubes (Function): [1, -8, 27, -64, 125, 216, -343, 512, 729, -1000]

Cubes (Lambda): [1, -8, 27, -64, 125, 216, -343, 512, 729, -1000]

Cubes (List Comp): [1, -8, 27, -64, 125, 216, -343, 512, 729, -1000]

Celsius (Function): [0.0, 10.0, 25.0, 37.77777777777778, 100.0]

Celsius (Lambda): [0.0, 10.0, 25.0, 37.77777777777778, 100.0]

Celsius (List Comp): [0.0, 10.0, 25.0, 37.77777777777778, 100.0]

Positive Numbers (Function): [1, 3, 5, 6, 8, 9]

Positive Numbers (Lambda): [1, 3, 5, 6, 8, 9]

Positive Numbers (List Comp): [1, 3, 5, 6, 8, 9]

Alphabets (Function): ['a', 'b', 'c', 'd']

Alphabets (Lambda): ['a', 'b', 'c', 'd']

Alphabets (List Comp): ['a', 'b', 'c', 'd']