Practical Set – 4

Program 4.1: Write a Program to find the prime number in a specific range using filter.

Code:

```
def is_prime_number(x):
    if x >= 2:
        for y in range(2,x):
            if not ( x % y ):
                return False
    else:
        return False
    return True
nums = range(2, 100)
f= filter(is_prime_number,nums)
for j in f:
    print(j)
```

Output:

```
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97
```

Program 4.2: Write a Program to make sum of particular range using reduce.

Code :

```
from functools import reduce
def do_sum(x1, x2):
    return x1 + x2
nums=range(1,30)
print(reduce(do_sum, nums))
```

Output :

435

Program 4.3: Write a Program to find Armstrong number in a specific range using map.

Code:

```
def find_armstrong(x,y):
    val=[]
    for num in range(x,y+1):
        a=list(map(int,str(num)))
        b=list(map(lambda x:x**3,a))
        if sum(b) == num:
            val.append(num)
    return val
a=find_armstrong(100,600)
print(a)
```

Output:

```
[153, 370, 371, 407]
```

Program 4.4: Use map() to create a function which finds the length of each word in the phrase (broken by spaces) and returns the values in a list.

Code:

```
s = "hello world this is the practical set 4 "
list(map(len, s.split()))
```

Output:-

```
[5, 5, 4, 2, 3, 9, 3, 1]
```

Practical 4.5: Write a Program to find Prime Factors of given number using recursion, filter() & Lambda function.

Code:

```
import math

def list_of_prime(x):
    filtered_list = list(range(2, x))
    for i in range (2, int(round(math.sqrt(x), 0))):
        filtered_list = list(filter(lambda x: x == i or x % i, filtered_list))
    return filtered_list

def prime_factor(x,y):
    if (x-1) != 0:
        for i in list_of_prime(x):
            if x % i == 0:
                y.append(i)
                x = x // i
                prime_factor(x, y)
        return y

prime_factor(75,[])
```

Output:

```
[3, 5, 5]
```

Practical 4.6: Use reduce() to take a list of digits and return the number that they correspond to

Code:

```
from functools import reduce
def appender_fun(x1,x2):
    return x1*10+x2
nums=[1,2,3]
print(reduce(appender_fun, nums))
```

Output:

123