[Day 5]

Map function – Applies a given function to all items in an input list (or any other iterable) and returns a map object (an iterator). This is particularly useful for transforming data in a list comprehensively.

```
def square(n):
    return(n * n)
square(10)
```

o/p: 100

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8]
list(map(square, numbers))
```

o/p : [1, 4, 9, 16, 25, 36, 49, 64]

Lambda function with map

```
numbers = [1, 2, 3, 4, 5]
list(map(lambda x: x * x, numbers))
```

o/p: [1, 4, 9, 16, 25]

Map multiple iterables.

```
numbers1 = [1, 2, 3]
numbers2 = [4, 5, 6]

added_numbers = map(lambda x, y: x+y, numbers1, numbers2)
ans = list(added_numbers)
print(ans)
```

op : [5, 7, 9]

map() to convert a list of strings to integers

```
str_num = ['1', '2', '3', '4', '5']
int_num = list(map(int, str_num))
print(int_num)
```

o/p: [1, 2, 3, 4, 5]

[Note]: Inbuilt fn can also be called with map fn.

Ex: upper_world = list(map(str.upper, words))

Filter() function — constructs an iterator from elements of an iterable for which a function returns true. It is used to filter out items from a list (or any other iterable) based on a condition.

Ex:
```
list_1 = [1, 2, 3, 4, 5, 6, 7, 8]
list(filter(even, list_1))
```

o/p: [2, 4, 6, 8]

Filter with lambda function

```
num = [1, 2, 3, 4, 5, 6, 7, 8]
greater = list(filter(lambda x: x>5, num))
print(greater)
```

o/p: [6, 7, 8]

filter with lambda fn & multiple conditions

```
num = [1, 2, 3, 4, 5, 6, 7, 8, 9]
even_and_greater = list(filter(lambda x: x>5 and x%2==0, num))
print(even_and_greater)
```

o/p: [6, 8]

Thus, filter() function is a powerful tool for creating iterators that filter items out of an iterable based on a function. It is commonly used for data cleaning, filtering objects, and removing unwanted elements from lists. By mastering filter(), we can write more concise & efficient code for processing & manipulating collections in Python.