→ finding whether a number is prime or not using labda ① expression & list comprehension.

```
is_prime = lambda n: n>1 and all(n%i != 0 for
        i in [x for x in range(2, int(n**0.5) +1)])
```

Eg:→ Checking whether n=10 is prime or not.

Step1:→ Generate possible divisors
range is [2, 3).

Step2:→ check divisibility.
10%2 ==0 → False

So, all(---) will return false.

∴ 10 is not prime

→ function to find out the sum of odd nos, even nos and nos. divisible by 3. Using lambda.

```
L = [11, 14, 21, 23, 56, 78, 45, 29, 28]

def return_sum(L):
    even_sum = 0
    Odd_sum = 0
    sum_div3 = 0
    for i in L:
        if i%2 ==0:
            even_sum = even_sum+i
```

P.T.O

```
elif i%2!=0:
    odd_sum = odd_sum+i

elif i%3==0:
    sum_div3 = sumdiv3+i

return(even_sum, odd_sum, sum_div3)
print(return_sum(L))
```

This is the normal function.

Now let's try to do the same thing using lambda expression.

```
def return_sum(func,L):
    result = 0
    for i in L:

        if func(i):
            result = result+i

    return result.

L=[11,14,21,23,56,78,45,29,28]

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

print(return_sum(x,L))
print (return_sum(y,L))
print (return_sum(z,L))
```

→ "Map() function in Python" :⇒

The map() function executes a specified function for each item in an iterable. The item is sent to the function as a parameter.

Syntax: map(func(), iterable)
↳ Can be a lambda function.

Eg:⇒ Let's say we need to double each element of a list.

```
L=[1,2,3,4,5,6,7]

map(lambda x: x*2 , L)
```

↳ It will return an object of type map.
So, let's convert it into a list

```
list(map(lambda x: x*2, L))
```

O/p: [2,4,6,8,10, 12,14]

Let's say we need to find out which of the list elements are even, It can be done like

```
list(map(lambda x: x%2 ==0, L))
```

O/p: [False, True, False, True, False, True, False]

P.T.O

let's say we have a list of directories as a database.

```
students = [
    {
        "name" : "Vipul Sharma",
        "father. name" : "ML Sharma",
        "Address" : 'Jammu"
    },
    {
        'name" : " Pankaj Dhiman",
        "father name" : " KD Dhiman",
        " Address" : "Hamirpur",
    },
    {
        "name" : " Nishant Singh",
        "father name" : " LK Singh",
        " Address" : " Shimla"
    }
]
```

Now, we need to iterate over this dataset & we need to fetch the name of each & every student.

```
list(map(lambda student : student['name'], students))
```

O/p ['Vipul Sharma', 'Pankaj Dhiman', 'Nishant Singh'].

→ Python filter() function:→

The filter() function returns an iterator where the items are filtered through a function to test if the item is accepted or not.

Syntax :- filter(function, iterable)

Here function is a function to be run for each item in the iterable.

& iterable is an iterable to be filtered.

Eg:→ L = [1, 2, 3, 4, 5, 6, 7]

we have to filter out the items which are greater than 4.

```
list(filter(lambda x: x>4, L))
```

o/p [5, 6, 7].

Eg:→ fruits = ['Apple', 'Orange', 'Mango', 'Guava']

we need to filter out the items which have "e" in it as a letter.

```
list(filter(lambda x: "e" in x, fruits))
```

o/p ['Apple', 'orange']

PTO

→ "Reduce function in Python" :→

The reduce function is used to apply a particular function passed in its argument to all of the list elements mentioned in the sequence passed along.

This function is defined in functools module.

Let's say we have a list

$L = [5, 7, 8, 13]$

Let's say we created a lambda function as :

```
lambda x,y : x+y
```

Now when we pass these two into reduce function, we will get the result as shown.

```
reduce(lambda x,y : x+y , L)
```

$$\begin{tikzcd}[5, 7, 8, 13] \arrow[d] \\ [12, 8, 13] \arrow[d] \\ [20, 13] \arrow[d] \\ 33\end{tikzcd}$$

Reduce basically reduces our list.

```
import functools
functools.reduce (lambda x,y : x+y, L)
```

Eg:- L1 = [12, 34, 56, 11, 21, 58]

```
functools.reduce(lambda x,y: x if x>y else y, L1)
```

O/p, 58

```
functools.reduce(lambda x,y: x if x<y else y, L1)
```

O/p, 11.