lambda :

- A lambda function is a small anonymous function.
- A lambda function can take any number of arguments, but can only have one expression.

```
lambda arguments : expression
```

In [ ]:

```
```

In [1]:

```python
lambda x:x+2
```

Out[1]:

```
<function __main__.<lambda>(x)>
```

In [2]:

```python
x = lambda x:x+2
x(1)
```

Out[2]:

```
3
```

In [3]:

```python
# addn using lambda fuctn

addx = lambda x,y:x+y
addx(10,20)
```

Out[3]:

```
30
```

In [ ]:

```
```

Lambda v/s Normal fuctn :

- Lambda fuctn has no return value
    - (it returns fuctn)
- 1 line
- Not used for code-readability as normal fuctn's
- no name
    - ex. def name()

Why ?

- Along with higher order fuctn

In [ ]:

```
```

check if the no. is even or odd

In [4]:

```python
check = lambda x:'Even' if x%2==0 else 'Odd'
check(12)
```

Out[4]:

```
'Even'
```

In [5]:

```python
check(7)
```

Out[5]:

```
'Odd'
```

```
In [ ]:
```

check if the string consist 3 char [scratch card]

- www : winning combination

```
In [6]: card = lambda x:'Won lottery' if x[0]=='w' and x[1]=='w' and x[2]=='w' el
```

```
In [7]: card('www')

Out[7]: 'Won lottery'
```

```
In [8]: card('xne')

Out[8]: 'Lose'
```

```
In [9]: card('wnw')

Out[9]: 'Lose'
```

```
In [ ]:
```

Inside list if

- all even no. : find sum
- all odd no : find sum
- all no/3 : find sum

```
In [21]: def listsol(l):
             even,odd,divby3 = 0,0,0

             for i in l:
                 if i%2==0:
                     even+=i
                 if i%3==0:
                     divby3+=i
                 if i%2!=0:
                     odd+=i

             print(f'Sum of even no in {l} is {even}')
             print(f'\nSum of odd no in {l} is {odd}')
             print(f'\nSum of Divby3 no in {l} is {divby3}')
```

```
In [22]: listsol([1,3,6,5])

         Sum of even no in [1, 3, 6, 5] is 6

         Sum of odd no in [1, 3, 6, 5] is 9

         Sum of Divby3 no in [1, 3, 6, 5] is 9
```

```
In [ ]:
```

Creating the same prog. using Lambda fuctn

In [17]:
```
def listsoln2(fuctn,l):
    res = 0
    for i in l:
        if fuctn(i):
            res+=i
    return res

even = lambda x:x%2==0
odd = lambda x:x%2!=0
divby3 = lambda x:x%3==0
```

In [18]:
```
listsoln2(even,[1,3,6,5])
```

Out[18]:
```
6
```

In [19]:
```
listsoln2(odd,[1,3,6,5])
```

Out[19]:
```
9
```

In [20]:
```
listsoln2(divby3,[1,3,6,5])
```

Out[20]:
```
9
```

In [ ]:

Higher Order Function :

- A function is called Higher Order Function if it contains other functions as a parameter or returns a function as an output
  - i.e, the functions that operate with another function are known as Higher order Functions.

In [ ]:

Different Higher order fuctn's are :

- Map()
- Filter()
- Reduce()

Map()

- Map in Python is a function that works as an iterator to return a result after applying a function to every item of an iterable (tuple, lists, etc.).

syntex : map(fun, iter)

NOTE : You can pass one or more iterable to the map() function.

Returns :

- Returns a list of the results after applying the given function to each item of a given iterable (list, tuple etc.)

```
In [ ]:
```

Create a program to multiply each item in a iterable

```
In [24]: l = [1,2,3,4,5,6]

         map(lambda x:x*2,l) # map fuctn returns a map obj

Out[24]: <map at 0x7f93443381c0>
```

```
In [25]: list(map(lambda x:x*2,l))

Out[25]: [2, 4, 6, 8, 10, 12]
```

```
In [ ]:
```

Creting a prog. to check whether each item of the list is divisible by 2 or not

```
In [27]: l = [1,2,3,4,5,6]
         list(map(lambda x:x%2==0,l))

Out[27]: [False, True, False, True, False, True]
```

```
In [ ]:
```

Extract name from the var student using map

```
In [30]: student = [{'name':'harsh',
                     'age':19,
                     'country':'USA'},
                    {'name':'methew',
                     'age':18,
                     'country':'Russia'},
                    {'name':'Rin',
                     'age':22,
                     'country':'Canada'}]
```

```
In [31]: student

Out[31]: [{'name': 'harsh', 'age': 19, 'country': 'USA'},
          {'name': 'methew', 'age': 18, 'country': 'Russia'},
          {'name': 'Rin', 'age': 22, 'country': 'Canada'}]
```

```
In [36]: list(map(lambda i:i['name'],student))

Out[36]: ['harsh', 'methew', 'Rin']
```

or.

```
In [35]: for i in student:
             print(i['name'])

         harsh
         methew
         Rin
```

In [ ]:

Filter

Print the no. which are greater than 4

In [40]:
```
l = [1,13,4,5,6,7,2,0]

list(map(lambda i:i>4,l))
```

Out[40]:
```
[False, True, False, True, True, True, False, False]
```

In [ ]:

In [45]:
```
list(filter(lambda i:i>4,l))
```

Out[45]:
```
[13, 5, 6, 7]
```

In [ ]:

Fetch the names of all the fruits that consist 'e' in their name

In [55]:
```
txt = '''Apple
Watermelon
Mandarin
Jackfruit
Papaya
Kiwi
Nectarine
Grape
Mango
Blueberry
Pomegranate'''
x = txt.split()
print(x)
```

```
['Apple', 'Watermelon', 'Mandarin', 'Jackfruit', 'Papaya', 'Kiwi', 'Nec
tarine', 'Grape', 'Mango', 'Blueberry', 'Pomegranate']
```

In [ ]:

In [56]:
```
fruits = ['Apple', 'Watermelon', 'Mandarin', 'Jackfruit', 'Papaya', 'Kiwi
list(filter(lambda i:'e' in i,fruits))
```

Out[56]:
```
['Apple', 'Watermelon', 'Nectarine', 'Grape', 'Blueberry', 'Pomegranat
e']
```

In [ ]:

Map v/s Filter :

In map:

- Function will be applied to all objects of iterable.

In filter:

- Function will be applied to only those objects of iterable who goes True on the condition specified in expression.

```
In [ ]:
```

Reduce :

- The reduce() function in python performs functional computation by taking a function and an iterable (eg: list, tuple, dictionary etc.) as arguments and result is returned after computation (the process of applying the function on the iterable)

Syntax

- functools.reduce(function, iterable)

```
In [59]: from functools import reduce

         nums = [1, 2, 3, 4]
         ans = reduce(lambda x, y: x + y, nums)
         print(ans)

         10
```

or.

```
In [61]: import functools
         nums = [1, 2, 3, 4]
         ans = functools.reduce(lambda x, y: x + y, nums)
         print(ans)

         10
```

```
In [ ]:
```

Find the greatest number inside the list

```
In [62]: l1 = [1,2,3,4,5,62,33,11,55,21]
         max(l1)

Out[62]: 62
```

or.

```
In [66]: l1 = [1,2,3,4,5,62,33,11,55,21]
         functools.reduce(lambda x,y: x if x>y else y,l1)

Out[66]: 62
```

```
In [ ]:
```

Find the least number inside the list

In [67]:
```
l1 = [1,2,3,4,5,62,33,11,55,21]
min(l1)
```

Out[67]:
```
1
```

In [69]:
```
functools.reduce(lambda x,y: x if x<y else y,l1)
```

Out[69]:
```
1
```

In [ ]:

In [ ]:

In [ ]: