Copy

copy.copy(x) :

- It is known as shallow copy . Let us know why it is said as shallow copy

lets take an example

L = [10 , 20 , 30 , 40 , 50]

- If you want to copy L in L1 . Then L1 will be holding the same object .
- L1 will create the new list is created but it will not create objects of the L

```
>>> import copy
>>> L = [10,20,30,40,50]
>>> L1 = copy.copy(L)
>>> L1
[10, 20, 30, 40, 50]
>>>
>>> id(L)
140302108233664
>>> id(L1)
140302108131392
>>>
>>> id(L[0])
140302061017680
>>> id(L1[0])
140302061017680
```

- If we check the id for L and L1 (list) it is different cause both are different List . But if we check the id of a particular object using indexing the id will be the same for L1 and L
- So it has shallowed copied it

copy.deepcopy(x)

- This is also said as recursive copy (copy of list then sublist than sublist than sublist …. So on)
- In this deep copy it will create the L1 (list) and it will create an object(numbers) too . This is like real duplicate
- By this deep copy will not work for int , float , string , complex etc …. It will works with classes , objects , functions ..

```
>>> import copy
>>> class Person:
        def __init__(self,name):
            self.name=name


>>> L = [Person('John'), Person('Tim'), Person('Jim')]
>>> L1 = copy.deepcopy(L)
>>> id(L[0])
140479058600576
>>> id(L1[0])
140479053248448
```