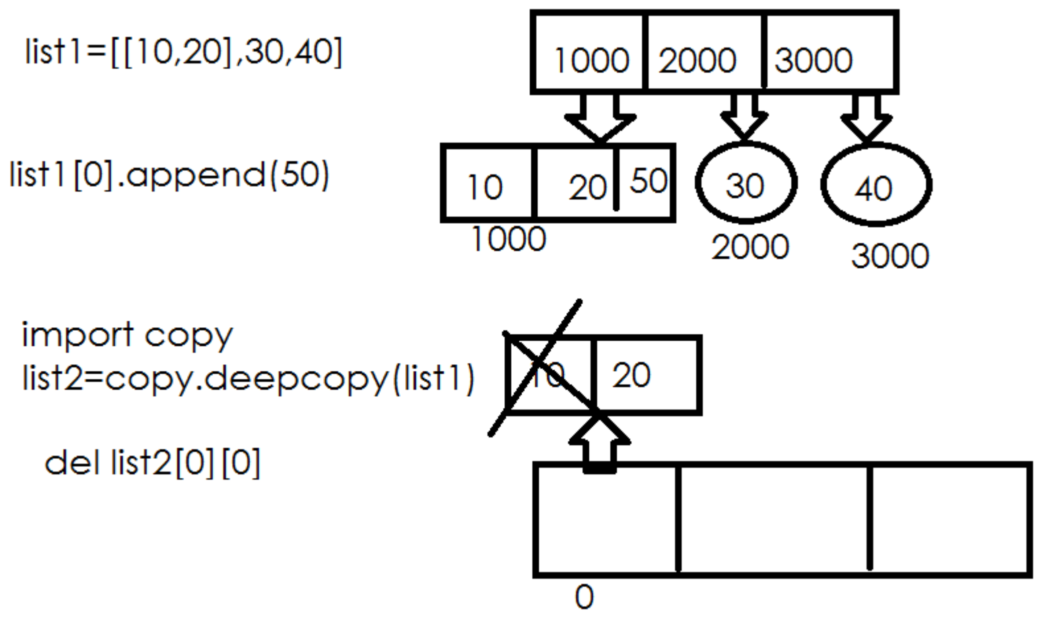
Deep copy
Deep copy is object copy
It will create list by adding copy of the objects
In order perform deep copy we use “copy” module
This module provide function to perform deep copy



```
list1=[[10,20],30,40]

list1[0].append(50)

import copy
list2=copy.deepcopy(list1)

  del list2[0][0]
```

Example:
```
import copy
list1=[[10,20],40,50]
list2=copy.deepcopy(list1)
print(list1)
print(list2)
list1[0].append(99)
print(list1)
print(list2)
del list2[0][0]
print(list2)
print(list1)
```
Output:
```
[[10, 20], 40, 50]
[[10, 20], 40, 50]
[[10, 20, 99], 40, 50]
[[10, 20], 40, 50]
[[20], 40, 50]
[[10, 20, 99], 40, 50]
```

s.insert(i, x)	inserts *x* into *s* at the index given by *i* (same as s[i:i] = [x])

```
list1=list(range(10,110,10))
print(list1)
list1.insert(0,99)
print(list1)
list1.insert(-1,88)
print(list1)
list1.insert(5,77)
print(list1)
list1[0:0]=[11,22]
print(list1)
```

Output:

```
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
[99, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
[99, 10, 20, 30, 40, 50, 60, 70, 80, 90, 88, 100]
[99, 10, 20, 30, 40, 77, 50, 60, 70, 80, 90, 88, 100]
[11, 22, 99, 10, 20, 30, 40, 77, 50, 60, 70, 80, 90, 88, 100]
```

s.extend(t) or s += t	extends *s* with the contents of *t* (for the most part the same as s[len(s):len(s)] = t)

```
list1=[10,20,30,40,50]
list2=[60,70,80,90,100]
print(list1)
print(list2)
list1.extend(list2)
print(list1)
l1=[10,20,30]
l2=[40,50]
l1.append(l2)
print(l1)
```

Output:

```
[10, 20, 30, 40, 50]
[60, 70, 80, 90, 100]
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
[10, 20, 30, [40, 50]]
```

s.reverse()	reverses the items of *s* in place

```
>>> list1=[10,20,40,90,100,5,6]
>>> print(list1)
[10, 20, 40, 90, 100, 5, 6]
>>> list1.reverse()
>>> print(list1)
[6, 5, 100, 90, 40, 20, 10]
>>>
```

sort(*, key=None, reverse=False)
This method sorts the list in place, using only < comparisons between items. Exceptions are not suppressed - if any comparison operations fail, the entire sort operation will fail (and the list will likely be left in a partially modified state).

```
>>> list1=[10,20,40,90,100,5,6]
>>> print(list1)
[10, 20, 40, 90, 100, 5, 6]
>>> list1.reverse()
>>> print(list1)
[6, 5, 100, 90, 40, 20, 10]
>>> list1=[10,5,6,1,9,12,8,3]
>>> list1.sort()
>>> print(list1)
[1, 3, 5, 6, 8, 9, 10, 12]
>>> list1.sort(reverse=False)
>>> print(list1)
[1, 3, 5, 6, 8, 9, 10, 12]
>>> list1.sort(reverse=True)
>>> print(list1)
[12, 10, 9, 8, 6, 5, 3, 1]
>>>
>>> list2=['A','B','C','a','b','c']
>>> list2=['a','B','c','A','C','b']
>>> list2.sort()
>>> list2
['A', 'B', 'C', 'a', 'b', 'c']
>>> list2.sort(key=str.upper)
>>> print(list2)
['A', 'a', 'B', 'b', 'C', 'c']
```

```
>>>
>>> list1=[100,'NIT',1.5,1+2j]
>>> list1.sort()
Traceback (most recent call last):
  File "<pyshell#18>", line 1, in <module>
    list1.sort()
TypeError: '<' not supported between instances of 'str' and 'int'
>>>
```

Nested List
A list within list is called nested list
Representing a list as an element inside list is called nested list
Nested list can be represented as matrix

```
[[1,”naresh”,”python”],[2,”suresh”,”java”],..]
```