In [1]:
```
lst1 = [1,2,3,4]
lst2 = lst1
```

In [2]:
```
print(lst1,id(lst1))
print(lst2,id(lst2))
```

```
[1, 2, 3, 4] 139896617740288
[1, 2, 3, 4] 139896617740288
```

In [3]:
```
lst1[1] = 1000
```

In [4]:
```
print(lst1,id(lst1))
print(lst2,id(lst2))
```

```
[1, 1000, 3, 4] 139896617740288
[1, 1000, 3, 4] 139896617740288
```

In [ ]:
```

```

Shallow copy :

- A shallow copy creates a new object which stores the reference of the original elements.
- So, a shallow copy doesn't create a copy of nested objects, instead it just copies the reference of nested objects.
- This means, a copy process does not create copies of nested objects itself.

Using .copy : Shallow copy

In [5]:
```
# using .copy() : different memory locn

lst1 = [1,2,3,4] # here the list consist of different items
lst2 = lst1.copy()
```

In [6]:
```
print(lst1,id(lst1))
print(lst2,id(lst2))
```

```
[1, 2, 3, 4] 139896617563072
[1, 2, 3, 4] 139896617740992
```

In [7]:
```
lst1[1] = 1000
```

In [8]:
```
print(lst1,id(lst1))
print(lst2,id(lst2))
```

```
[1, 1000, 3, 4] 139896617563072
[1, 2, 3, 4] 139896617740992
```

In [ ]:
```

```

In [13]:
```
# Shallow copy : with respect to nested list

lst1 = [[1,2,3,4],[5,6,7,8]]
'''
Here the list consist of 2 sub list as items
and these sub list consist of different objects
'''
lst2 = lst1.copy()
```

In [14]:
```
print(lst1,id(lst1))
print(lst2,id(lst2))
```

```
[[1, 2, 3, 4], [5, 6, 7, 8]] 139896488209984
[[1, 2, 3, 4], [5, 6, 7, 8]] 139896480348864
```

In [15]:
```
lst1[1][0] = 1000
```

In [16]:
```
print(lst1,id(lst1))
print(lst2,id(lst2))

'''
Here we can see the value 1000 is been updated in both
the nested list,
because it is referring to the same obj present
inside the nested list
'''
```

```
[[1, 2, 3, 4], [1000, 6, 7, 8]] 139896488209984
[[1, 2, 3, 4], [1000, 6, 7, 8]] 139896480348864
```

In [ ]:
```

```

In [17]:
```
# Shallow copy : with respect to nested list

lst1 = [[1,2,3,4],[5,6,7,8]]
'''
Here the list consist of 2 sub list as items
and these sub list consist of different objects
'''
lst2 = lst1.copy()
```

In [18]:
```
print(lst1,id(lst1))
print(lst2,id(lst2))
```

```
[[1, 2, 3, 4], [5, 6, 7, 8]] 139896480339904
[[1, 2, 3, 4], [5, 6, 7, 8]] 139896127516160
```

In [19]:
```
lst1.append([11,12,13,1])
```

In [20]:
```
print(lst1,id(lst1))
print(lst2,id(lst2))
```

```
[[1, 2, 3, 4], [5, 6, 7, 8], [11, 12, 13, 1]] 139896480339904
[[1, 2, 3, 4], [5, 6, 7, 8]] 139896127516160
```

In [ ]:
```

```

Deep Copy :

- A deep copy is a process where we create a new object and add copy elements recursively.
- In case of deep copy, a copy of object is copied in other object.
- We will use the deecopy() method which present in copy module.

In [22]:
```
'''
When u have a 1D list, It works same as shallow copy
'''
import copy

lst1 = [1,2,3,4] # here the list consist of different items
lst2 = copy.deepcopy(lst1)
```

In [23]:
```
print(lst1,id(lst1))
print(lst2,id(lst2))
```

```
[1, 2, 3, 4] 139896127644480
[1, 2, 3, 4] 139896127229120
```

In [24]:
```
lst2[0]=1556
```

In [25]:
```
print(lst1,id(lst1))
print(lst2,id(lst2))
```

```
[1, 2, 3, 4] 139896127644480
[1556, 2, 3, 4] 139896127229120
```

Note :

- In a normal list shallow copy == deep copy

In [26]:
```
import copy

lst1 = [[1, 2, 3, 4], [5, 6, 7, 8], [11, 12, 13, 1]]
# here the list consist of different items

lst2 = copy.deepcopy(lst1)
```

In [27]:
```
print(lst1,id(lst1))
print(lst2,id(lst2))
```

```
[[1, 2, 3, 4], [5, 6, 7, 8], [11, 12, 13, 1]] 139896127219264
[[1, 2, 3, 4], [5, 6, 7, 8], [11, 12, 13, 1]] 139896127223680
```

In [28]:
```
lst2[0][3]=4444
```

In [29]:
```
print(lst1,id(lst1))
print(lst2,id(lst2))
```

```
[[1, 2, 3, 4], [5, 6, 7, 8], [11, 12, 13, 1]] 139896127219264
[[1, 2, 3, 4444], [5, 6, 7, 8], [11, 12, 13, 1]] 139896127223680
```

In [ ]: