python™

DURGA SOFTWARE SOLUTIONS®

Learn Complete Python In Simple Way

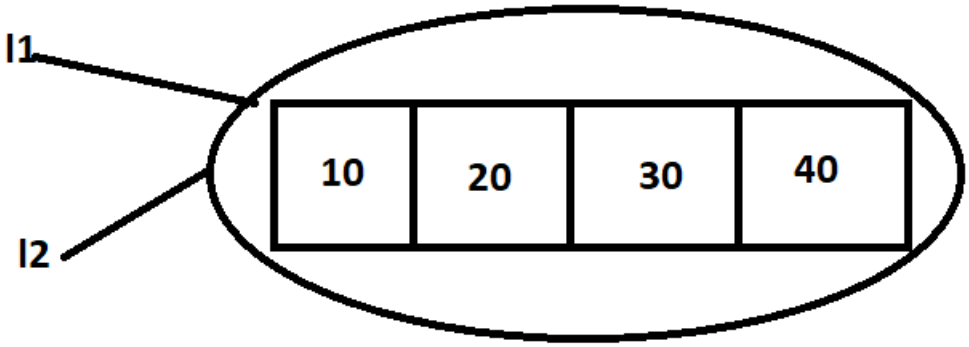
python™

DURGA SOFTWARE SOLUTIONS©

Aliasing and Cloning,Deep Copy and Shallow Copy

STUDY MATERIAL

Aliasing: Aliasing is the process of creating duplicate reference variable.



l1=[10,20,30,40]
l2=l1 assingning l1 reference to l2(Aliasing)

e.g.

```
l1=[10,20,30,40]
l2=l1 #creating duplicate reference variable (Aliasing)
l1[1]=60 #If we try to make any change in l1 elements it will be automatically reflecting to l2 as both references pointing to same object.
print(f'l1:{l1}')
print(f'l2:{l2}')
```

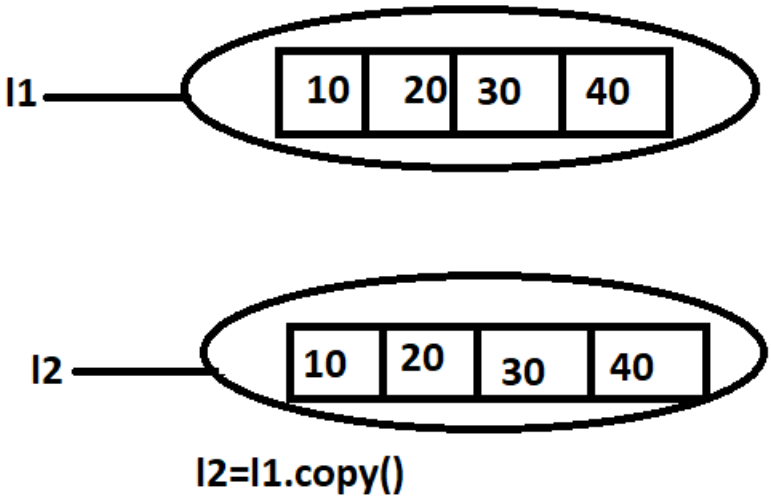
Output:

```
l1:[10, 60, 30, 40]
l2:[10, 60, 30, 40]
```

Note: By using Aliasing if we perform any changes in one reference it will reflect automatically those changes to remaining references also.

Cloning or Shallow Cloning: The process of creating duplicate object instead of creating duplicate reference variable is known as Cloning.

If we want duplicate object instead of duplicate reference variable then we should go for cloning.



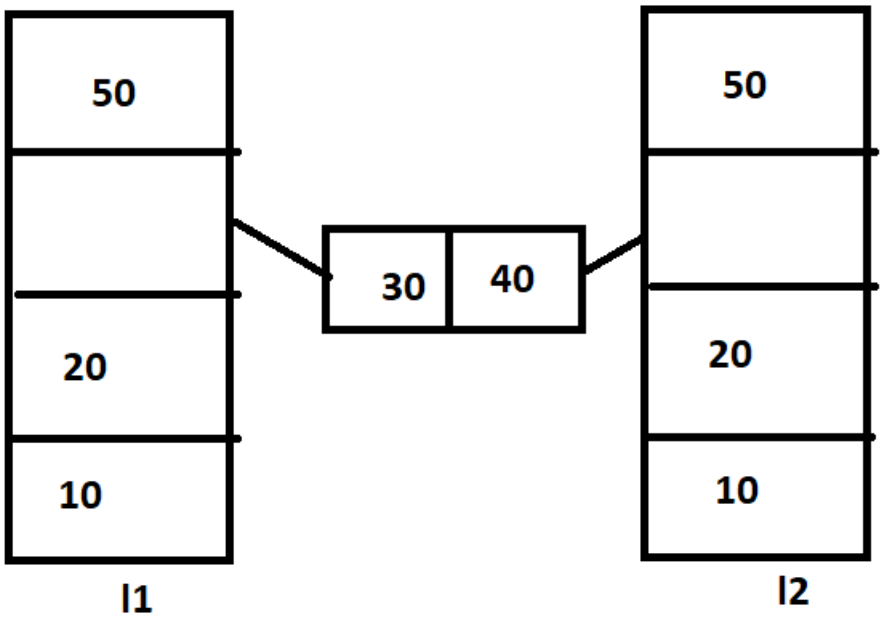
e.g.

```
l1=[10,20,30,40]
l2=l1.copy() #creating duplicate object
l1[1]=60 #If we try to make any change in l1 elements it will not be automatically reflecting to l2 as both references pointing to different objects by using .copy().
print(f'l1:{l1}')
print(f'l2:{l2}')
```

Output:

```
l1:[10, 60, 30, 40]
l2:[10, 20, 30, 40]
```

Note : For nested object in a list duplicate object not created by using shallow cloning only duplicate reference is created and it points to old object even if you create new object.



l2=l1.copy()

Shallow Cloning

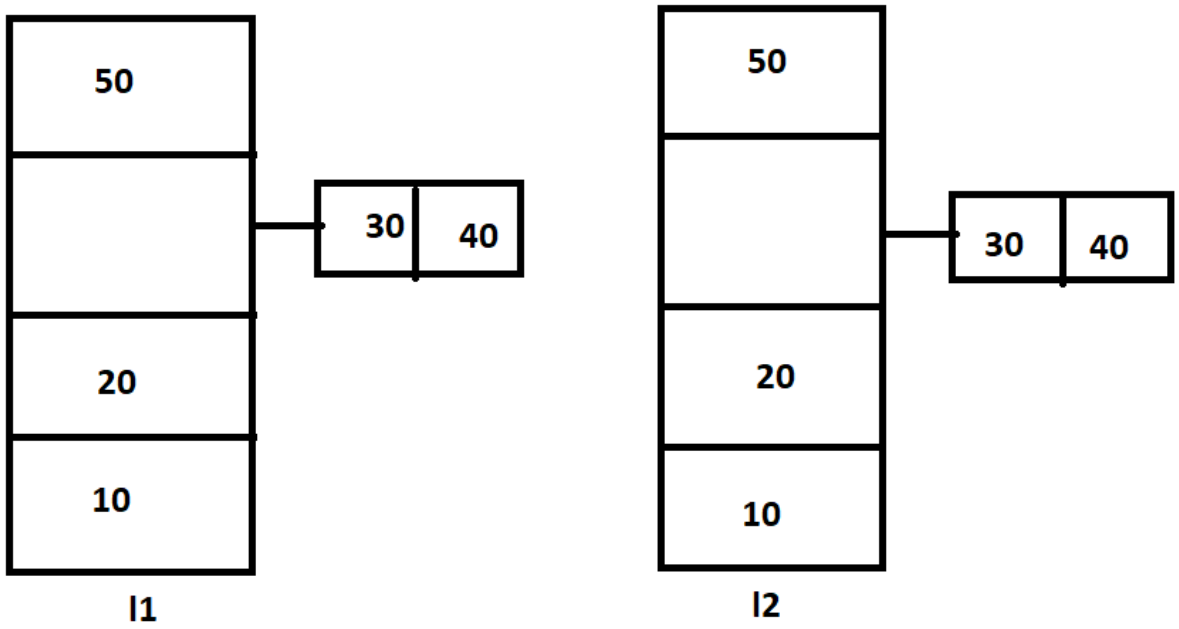
e.g.

```
l1=[10,20,[30,40],50]
l2=l1.copy() #Shallow Cloning
l1[2][0]=80 # If we try to change nested element it will reflect changes in l2 as well even if we creating duplicate object in  case of shallow cloning.
print(f'l1:{l1}')
print(f'l2:{l2}')
print(f'Id of l1[2][0]:{id(l1[2][0])} and Id of l2[2][0]:{id(l2[2][0])}')# printing same id or address for nested elements in shallow cloning.
```

Output:

```
l1:[10, 20, [80, 40], 50]
l2:[10, 20, [80, 40], 50]
Id of l1[2][0]:140713858226752 and Id of l2[2][0]:140713858226752
```

Deep Cloning: The process of creating duplicate object for object as well as nested object is nothing but Deep Cloning. For complete complete duplication of objects like nested objects one can use Deep Cloning.



l2=copy.copy(l1)

Deep Cloning

e.g.

```
import copy
l1=[10,20,[30,40],50]
l2=copy.copy(l1) #Deep Cloning
l1[2][0]=80 # If we try to change nested element it will not reflect changes in l2 as we have creating duplicate object as well as duplicate nested object in case of deep cloning.
print(f'l1:{l1}')
print(f'l2:{l2}')
print(f'Id of l1[2][0]:{id(l1[2][0])} and Id of l2[2][0]:{id(l2[2][0])}')# printing different id or address for nested elements in deep cloning.
```

Output:

```
l1:[10, 20, [80, 40], 50]
l2:[10, 20, [30, 40], 50]
Id of l1[2][0]:140713858226752 and Id of l2[2][0]:140713858226542
```