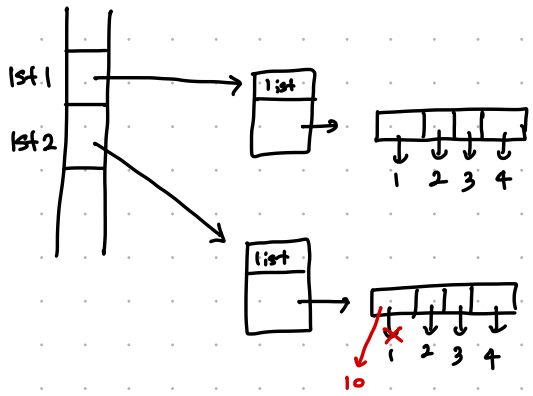
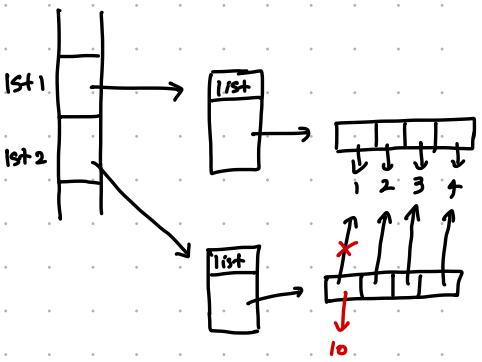
```
> import copy
> lst1 = [1,2,3,4]
> lst2 = copy.deepcopy(lst1)
> lst2[0] = 10
```



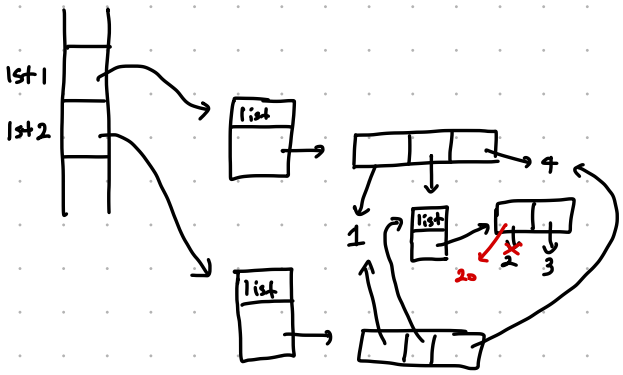
- slow
- more space

```
> import copy
> lst1 = [1,2,3,4]
> lst2 = copy.copy(lst1)
> lst2[0] = 10
```

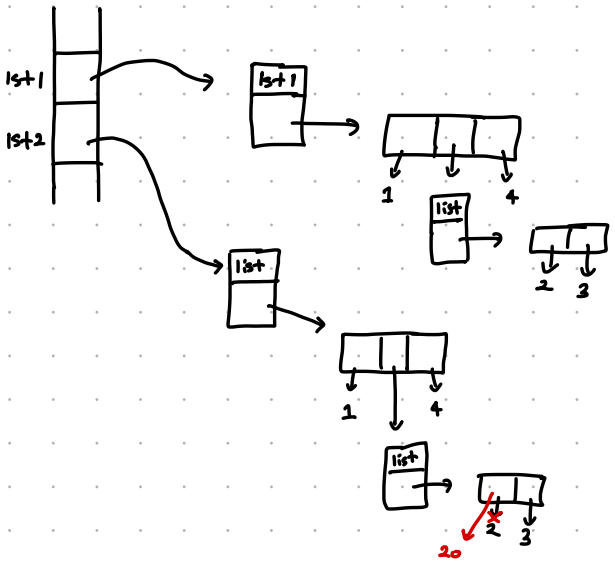


- faster than deepcopy
- more efficient
- less space than deep copy

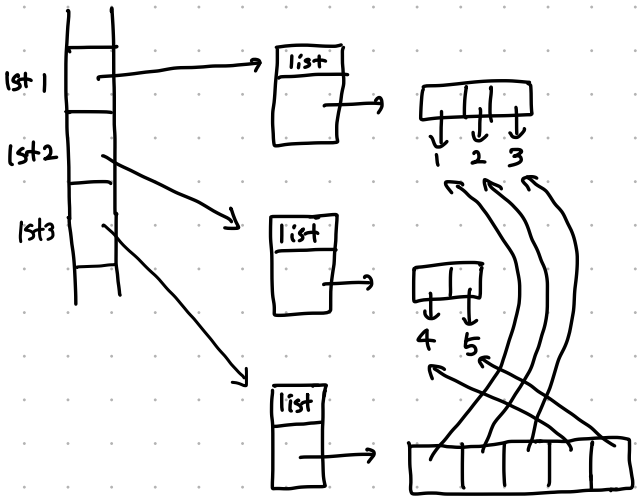
```
> lst1 = [1, [2,3], 4]
> lst2 = copy.copy(lst1)
> lst2[1][0] = 20
```



```
> lst1 = [1, [2,3], 4]
> lst2 = copy.deepcopy(lst1)
> lst2[1][0] = 20
```




```
> lst1 = [1,2,3]
> lst2 = [4, 5]
> lst3 = lst1 + lst2
```

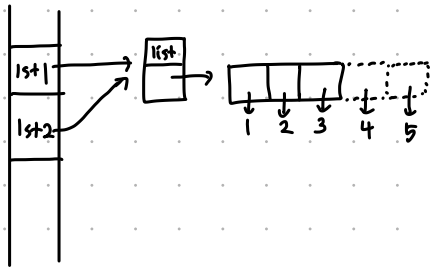



```
> lst1 = [1,2,3]
> lst2 = [ x ** x  for  x  in  lst1 ]
```

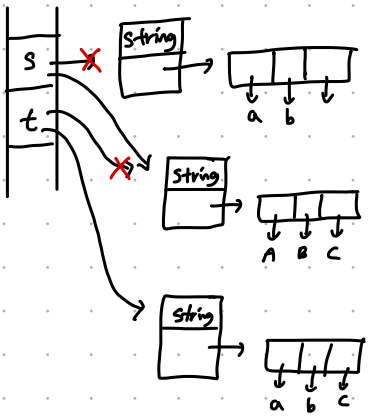
[f(x) for x in collection]

- Python default is Shallow copy

```
> lst1 = [1, 2, 3]
> lst2 = lst1
> lst1.append(4)
> lst2.append(5)
```

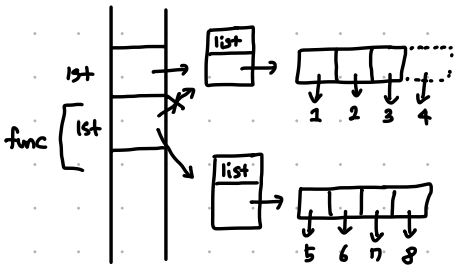



```
> s = "abc"
> s = s.upper()
> t = s
> t = t.lower()
```




```
> lst = [1, 2, 3]
> def func(lst):
      lst.append(4)
      lst = [5, 6, 7, 8]
      print("Inside func lst =", lst)

> print(lst)
> func(lst)
> print(lst)
```

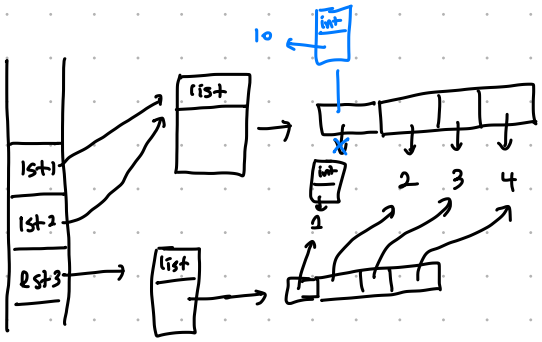



```
[1, 2, 3]
Inside func lst = [5, 6, 7, 8]
[1, 2, 3, 4]
```

ex)

```
lst1 = [1,2,3,4]
lst2 = lst1
lst3 = copy(lst)
lst1[0] = 10
```

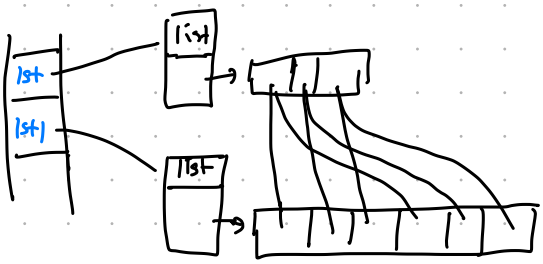
↳ 이건 not shallow copy 에러리얼트이다



Shallow/deep 든 copy 하면 새로운 object 만듦

(whatever shallow or deep copy, creates a new object)

```
lst = [1,2,3]
lst1 = lst *2
```




```
lst1 = [1, 2, [n3, 19]]
lst2 = lst1 *2
lst3 = lst2[::-2]
lst4 = [lst1 for i in range(2)]
lst1[2][1] = "cu"
lst1.pop()
print(lst1)
print(lst2)
print(lst3)
print(lst4)
```

- Box is removed, but the object remains there.
- 박스는 사라져, object는 남아있다