
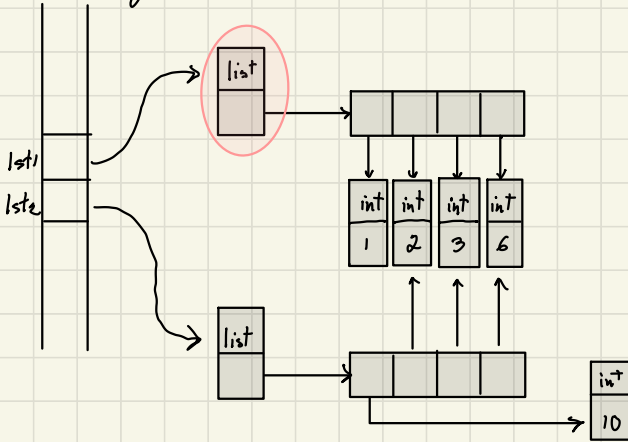


Memory Maps 2



Shallow Copy



from copy import copy

lst1 = [1, 2, 3, 6]

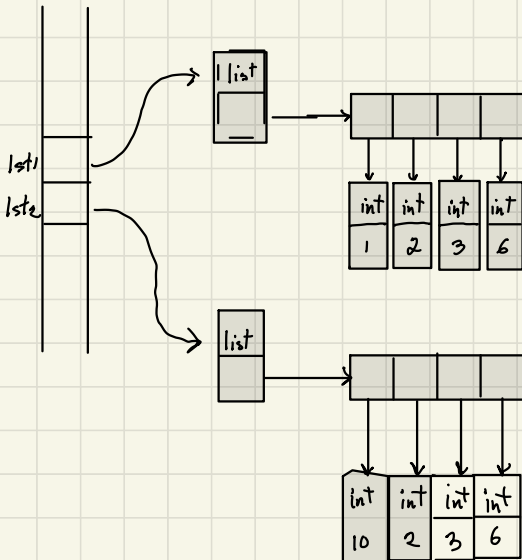
lst2 = copy(lst1)

lst2[0] = 10

print(lst1)

print(lst2)

Deep Copy



from copy import deepcopy

lst1 = [1, 2, 3, 6]

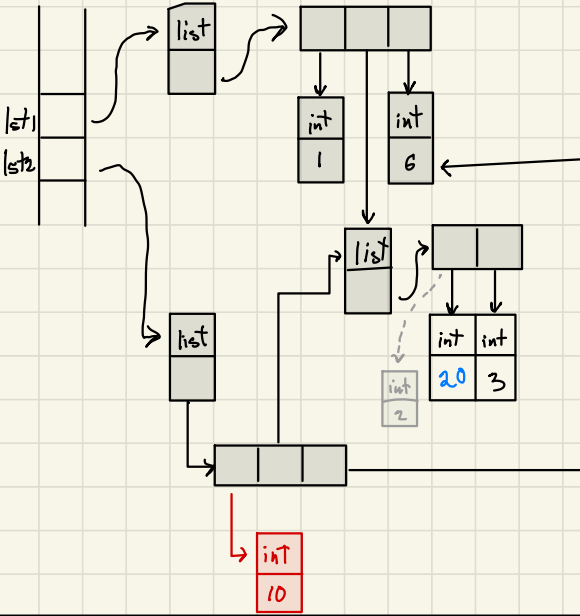
lst2 = deepcopy(lst1)

lst2[0] = 10

print(lst1)

print(lst2)

Shallow Copy (Nested Lists)



from copy import copy

lst1 = [1, [2, 3], 6]

lst2 = copy(lst1)

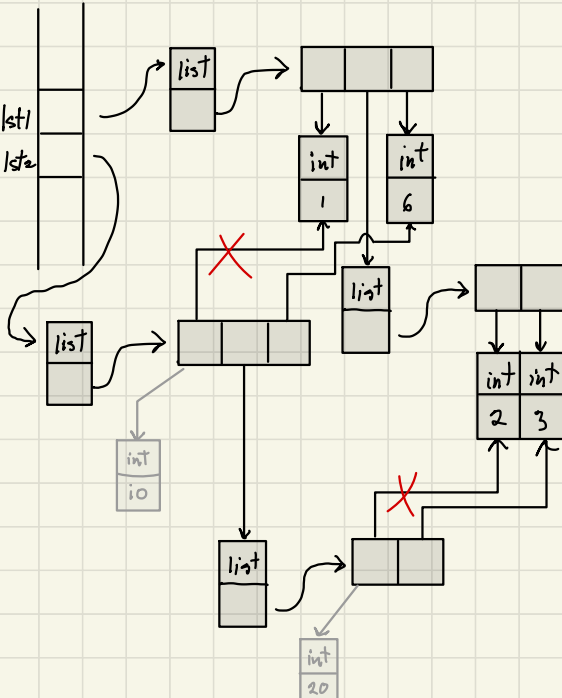
lst2[0] = 10 #1

lst2[1][0] = 20 #2

print(lst1) # [1, [20, 3], 6]

print(lst2) # [10, [20, 3], 6]

Deep Copy (Nested Lists)



from copy import deepcopy

lst1 = [1, [2, 3], 6]

lst2 = deepcopy(lst1)

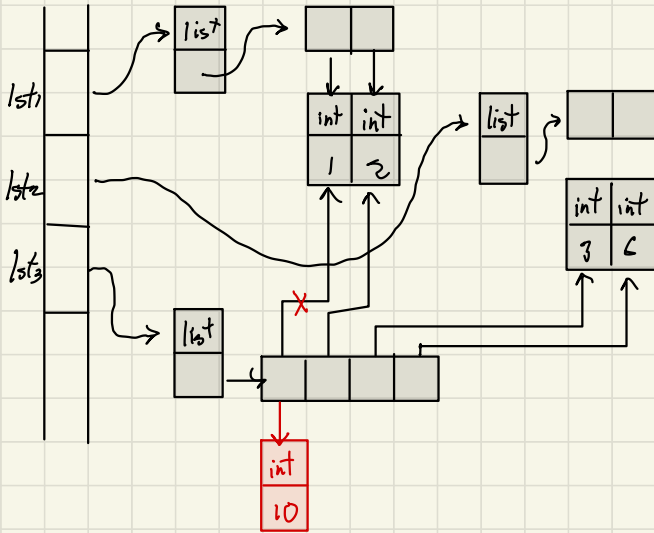
lst2[0] = 10

lst2[1][0] = 20

print(lst1) # [1, [2, 3], 6]

print(lst2) # [10, [20, 3], 6]

List Concatenation



`lst1 = [1, 2]`

`lst2 = [3, 6]`

`lst3 = lst1 + lst2`

`lst3[0] = 10`

*List slicing
works the same!*

List Comprehension

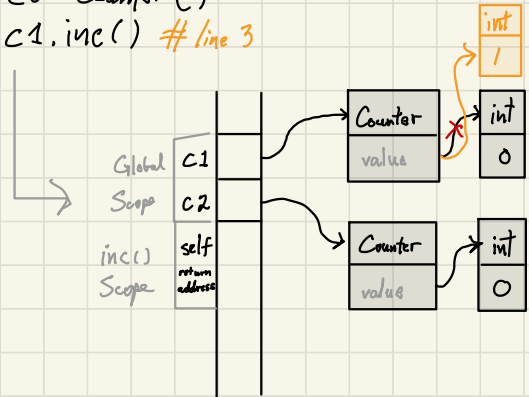
lst1 = [1, 2, 3, 4]

lst2 = [x * x for x in lst1] # [1, 4, 9, 16]

Counter Object

```
class Counter:
    def __init__(self):
        self.value = 0
    def inc(self):
        self.value += 1
    def __repr__(self):
        return str(self.value)
```

c1 = Counter()
c2 = Counter()
c1.inc() # line 3



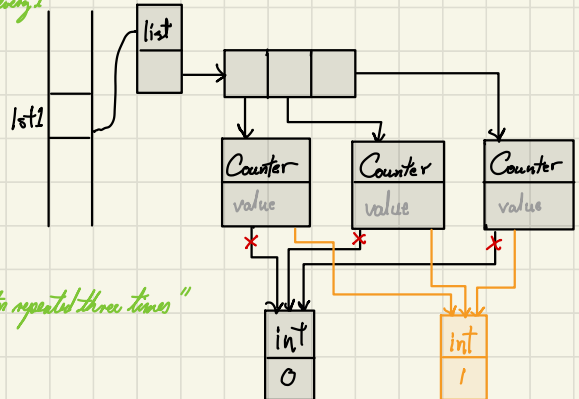
"A separate instance of Counter for every i"

lst1 = [Counter() for i in range(3)]

```
for c in lst1:
    c.inc() # a
    print(c)
```

Output

```
> 1
> 1
> 1
```



"One single instance of Counter repeated three times"

lst2 = [Counter()] * 3

```
for c in lst2:
    c.inc()
    print(id(c), c)
```

Output

```
> 433025560 1
> 433025560 2
> 433025560 3
```

