Chapter 4, Miller 3rd ed, Introducing the Python Collections

Section 4.3.2, Miller 3rd ed, Lists

- We've seen one type of Python collection — strings

1. Lists
   a. A list is a sequential collection of objects.
   [4, 8, 12]
   ['Iowa', 19, -37.286]

   b. Like strings, we have operations for lists

   ```
   myList = [2, 4, 6, 8, 10]
   yourList = ['newt', 'stone', 'witch']
   ```

   Indexing
   `yourList[1]` → 'stone'

   Concatenation
   `myList + yourList` → [2, 4, 6, 8, 10, 'newt', 'stone', 'witch']

   Repetition
   `yourList * 3`
   → ['newt', 'stone', 'witch', 'newt', 'stone', 'witch', 'newt', 'stone', 'witch']

   Membership
   `'newt' in yourList` → True
   `'lumberjack' in yourList` → False
   `13 not in myList` → True

   Length
   `len(yourList)` → 3

   Slicing
   `myList[2:]` → [6, 8, 10]
   `myList[1::3]` → [4, 10]
   `myList[::2]` → [2, 6, 10]

c. Lists are mutable (unlike strings)

```
yourList = ['newt', 'stone', 'witch']
yourList[2] = 'shrubbery'
```
→ ['newt', 'stone', 'shrubbery']

Can't do this with a string:

```
name = 'lancelot'
name[0] = 'L'
```
→ Error

```
del yourList[1]
```
→ ['newt', 'shrubbery']

d. Iterating over a list

```
L = [1, 2, 3, 'a', 'b', 'c']
for item in L:
    print(item, end=' ')
```
→ 1 2 3 a b c

```
for i in range(len(L)):
    L[i] = L[i] + L[i]
```
→ L: [2, 4, 6, 'aa', 'bb', 'cc']

2. 2D lists, list of lists

Print a Square 2D List

```
L = [[1,2,3], [4,5,6], [7,8,9]]
for i in range(len(L)):
    for j in range(len(L[i])):
        print(L[i][j], end = ' ')
    print()
```

```
1 2 3
4 5 6
7 8 9
```

Print a Rectangular 2D List

```
L = [[1,2,3,4],[5,6,7,8]]
```
Print the same way

```
1 2 3 4
5 6 7 8
```

```
L = [[1,2],[3,4],[5,6],[7,8]]
```
Print the same way

```
1 2
3 4
5 6
7 8
```

Print a Jagged 2D List

```
L = [[0,1,2],[],[3,4,5,6,7],[8,9]]
```
Print the same way

```
0 1 2

3 4 5 6 7
8 9
```

Better way to print 2D List

```
for subL in L:
    for item in subL:
        print(item, end=' ')
    print()
```

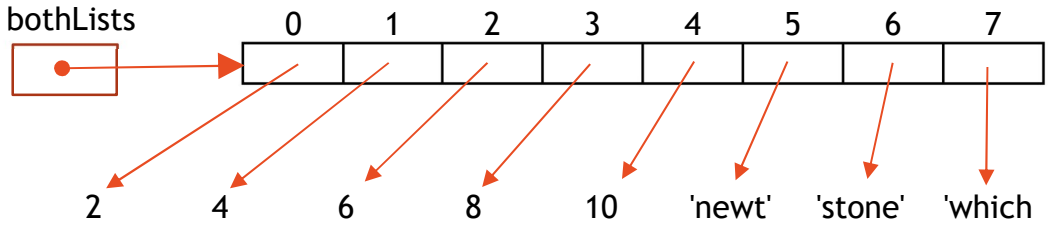
```
0 1 2

3 4 5 6 7
8 9
```

3. Multiple references to mutable objects

A List is really a collection of references

```
myList = [2, 4, 6, 8, 10]
yourList = ['newt', 'stone', 'witch']
bothLists = myList + yourList
```

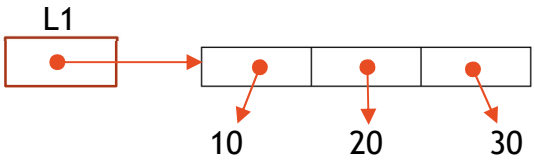
→ [2, 4, 6, 8, 10, 'newt', 'stone', 'witch']



Changing a list element **Only Changes the Reference**

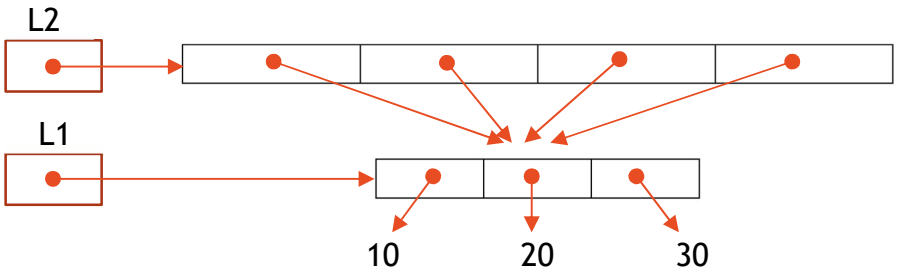
```
L1 = [10, 20, 30]
```

→ [10, 20, 30]




```
L2 = [L1, L1, L1, L1]
```

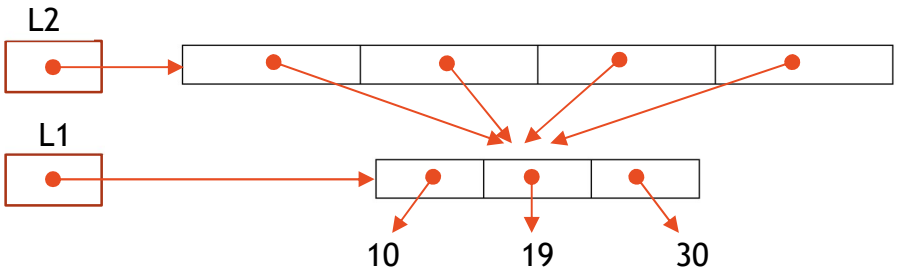
→ [[10, 20, 30], [10, 20, 30], [10, 20, 30]]



L1[1] = 19 → [10, 19, 30]

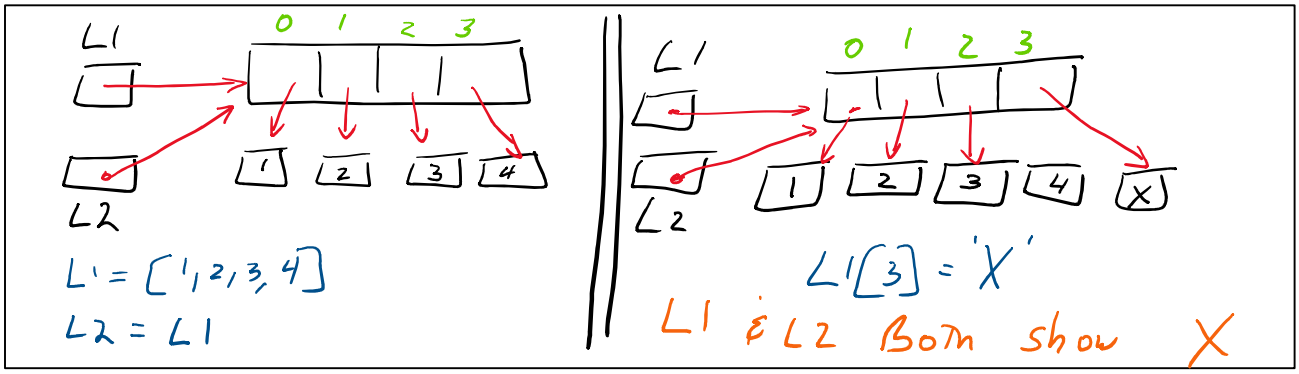
Question: What will L2 look like?
Answer: Every 20 replaced with 19

L2 → [[10, 19, 30], [10, 19, 30], [10, 19, 30]]



4. Copying Lists – Still related to "**Multiple References to Mutable Objects**"
   a. Assignment doesn't make a copy

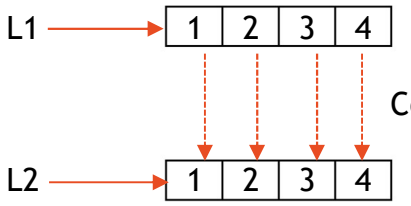
L1 = [1, 2, 3, 4] → Make a new list
L2 = L1 → Two references to the same list
L1[3] = 'X' → [1, 2, 3, X] - Changes that one list
L2 → [1, 2, 3, X]



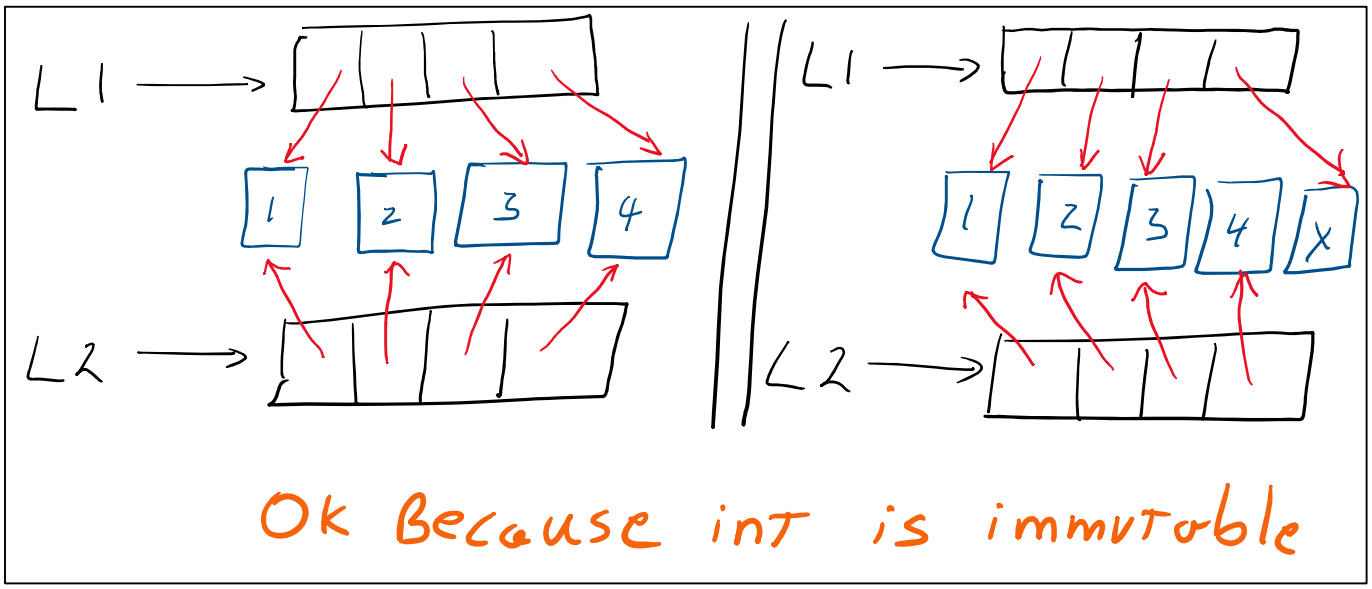
i. Copy slice

```
L1 = [1, 2, 3, 4]
L2 = L1[:]
L1[3] = 'X'
L2
```

→ Make a new list
→ Make an actual copy
→ [1, 2, 3, 'X'] - Changes only the copy
→ [1, 2, 3, 4] - L2 is not changed

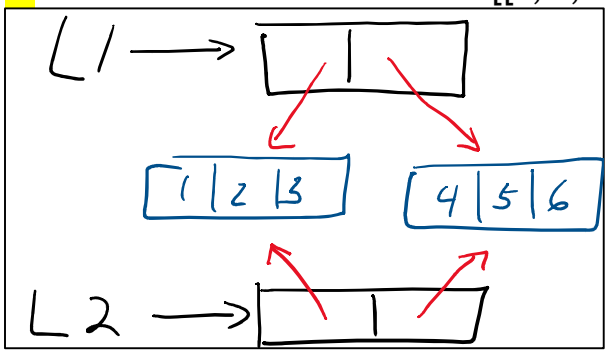


Copy Refs to **Immutable** int objects

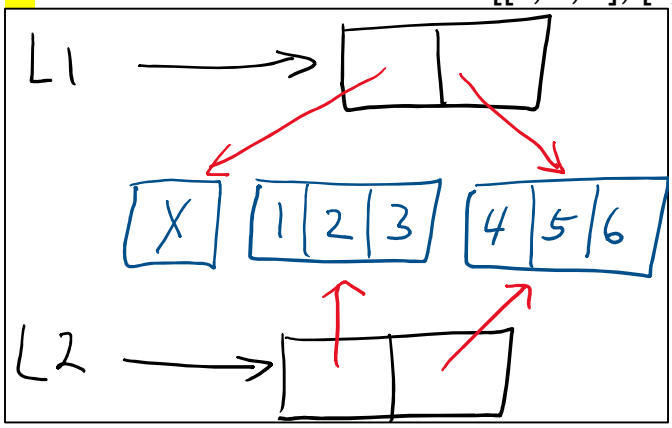


b. Copy slice doesn't work for list of **mutable** objects (like lists)

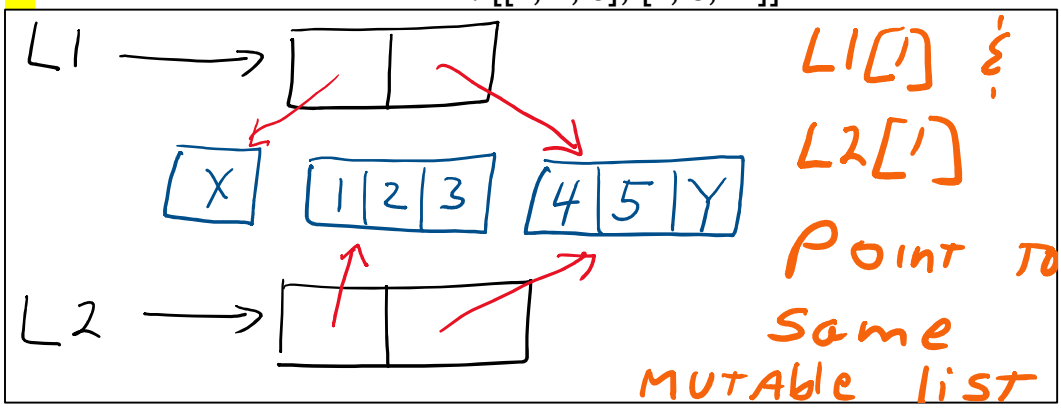
```
L1 = [[1, 2, 3], [4, 5, 6]]
L2 = L1[:]
L1                    → L1: [[1, 2, 3], [4, 5, 6]]
L2                    → L2: [[1, 2, 3], [4, 5, 6]]
```




```
L1[0] = 'X'
L1                    → L1: ['X', [4, 5, 6]]
L2                    → L2: [[1, 2, 3], [4, 5, 6]]
```




```
L1[1][2] = 'Y'
L1                    → L1: ['X', [4, 5, 'Y']]
L2                    → L2: [[1, 2, 3], [4, 5, 'Y']]
```



c. Deep Copy

```
import copy
L1 = [[1, 2, 3], [4, 5, 6]]
L2 = copy.deepcopy(L1)
L1                          → L1: [[1, 2, 3], [4, 5, 6]]
L2                          → L2: [[1, 2, 3], [4, 5, 6]]

L1[0] = 'X'
L1                          → ['X', [4, 5, 6]]
L2                          → [[1, 2, 3], [4, 5, 6]]

L1[1][2] = 'Y'
L1                          → ['X', [4, 5, 'Y']]
L2                          → [[1, 2, 3], [4, 5, 6]]
```