LECTURE 9

DOHYUNG KIM

WHAT IS DISCUSSED IN THE LAST CLASS

- List

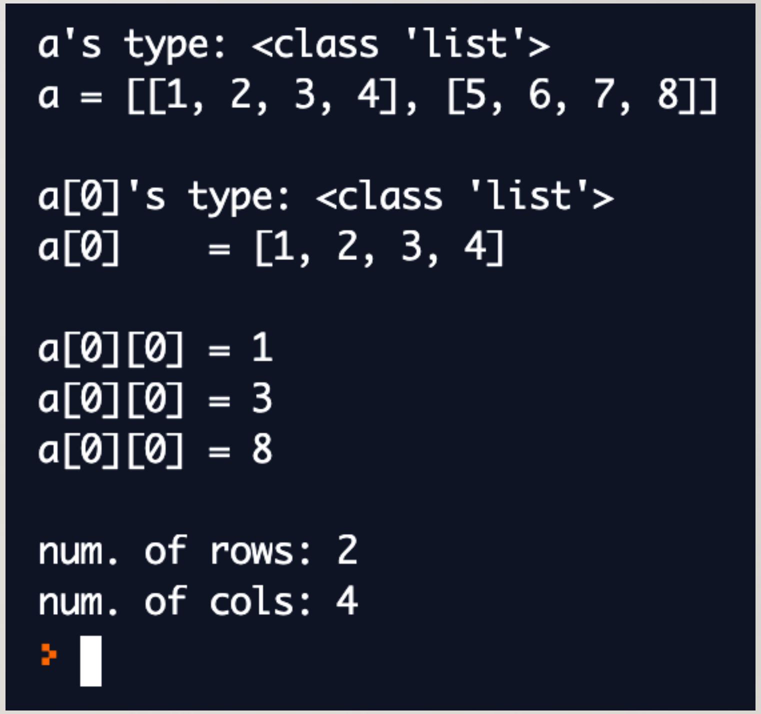
TODAY, WE WILL LEARN ABOUT

- More about list

LISTS INSIDE A LIST

- List is a object, and it can be an element of a list

```python
a = [[ 1, 2, 3, 4], [5, 6, 7, 8]]
print("a's type:", type(a))
print("a =", a, "\n")

print("a[0]'s type:", type(a[0]))
print("a[0]     =", a[0], "\n")

print("a[0][0] =", a[0][0])
print("a[0][2] =", a[0][2])
print(“a[1][3] =", a[1][3], "\n")

#Dimension info
print("num. of rows:", len(a))
print("num. of cols:", len(a[0]))
```



```
a's type: <class 'list'>
a = [[1, 2, 3, 4], [5, 6, 7, 8]]

a[0]'s type: <class 'list'>
a[0]     = [1, 2, 3, 4]

a[0][0] = 1
a[0][0] = 3
a[0][0] = 8

num. of rows: 2
num. of cols: 4
```

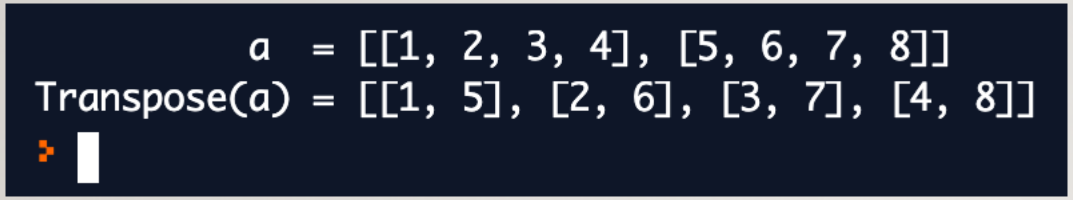
PRACTICE

- What would be the result?

```
a = [[ 1, 2, 3, 4], [5, 6, 7, 8]]
row = 1
rowList = a[row]
print(rowList)

a = [[ 1, 2, 3, 4], [5, 6, 7, 8]]
col = 1
colList = [ ]
for i in range(len(a)):
  colList += [ a[i][col] ]
print(colList)

a = [[ 1, 2, 3, 4], [5, 6, 7, 8]]
col = 1
colList = [ a[i][col] for i in range(len(a)) ]
print(colList)
```

PRACTICE

- Transpose a given matrix (change row and column of the matrix)

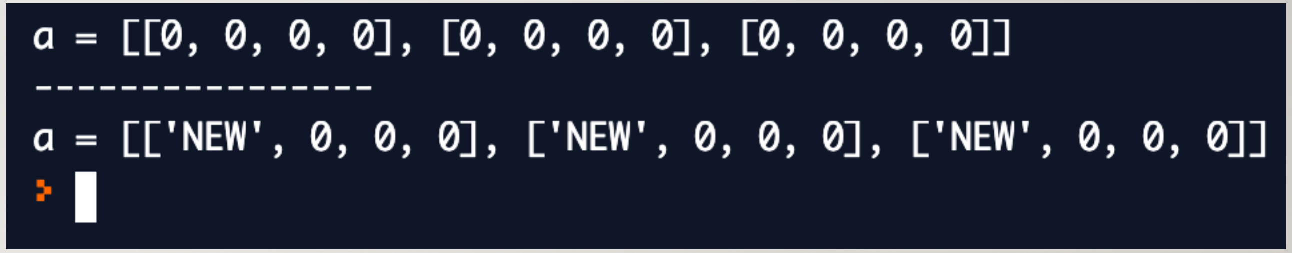


```
           a = [[1, 2, 3, 4], [5, 6, 7, 8]]
Transpose(a) = [[1, 5], [2, 6], [3, 7], [4, 8]]
>
```

CREATING 2D LIST

- Let's create a list of 3x4 size

```python
row = 3
col = 4

a = [[0]*col]*row
print("a =", a)

print("----------------")

a[0][0] = "NEW"
print("a =", a)
```



```
a = [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
----------------
a = [['NEW', 0, 0, 0], ['NEW', 0, 0, 0], ['NEW', 0, 0, 0]]
❯
```

* This is surely something that we DO NOT WANT. By the way, why do we have this result?

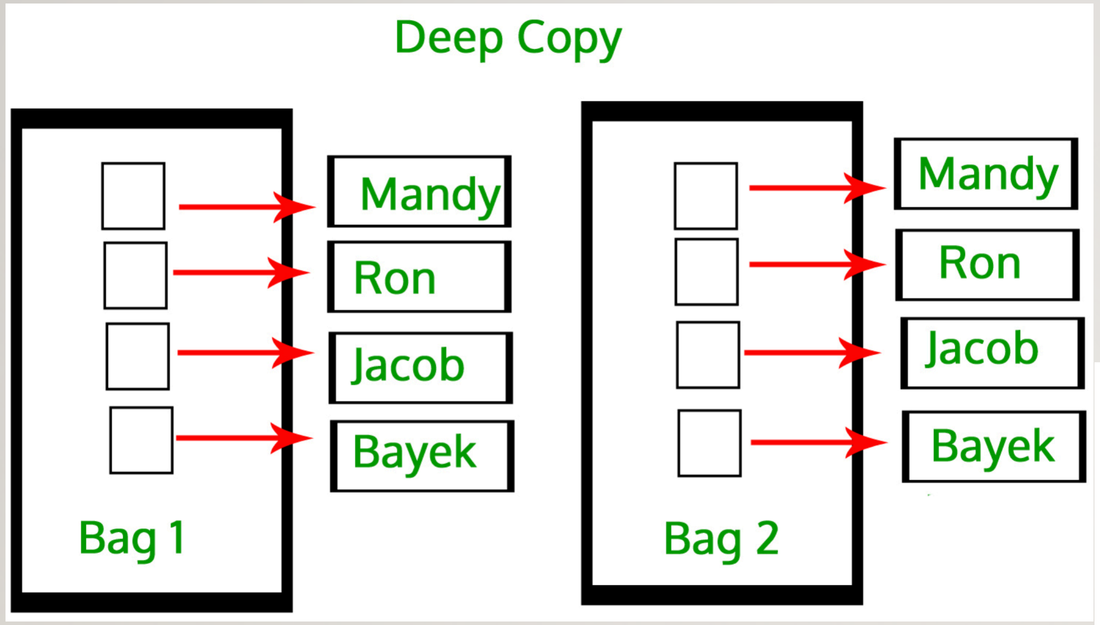
Shallow copy!!

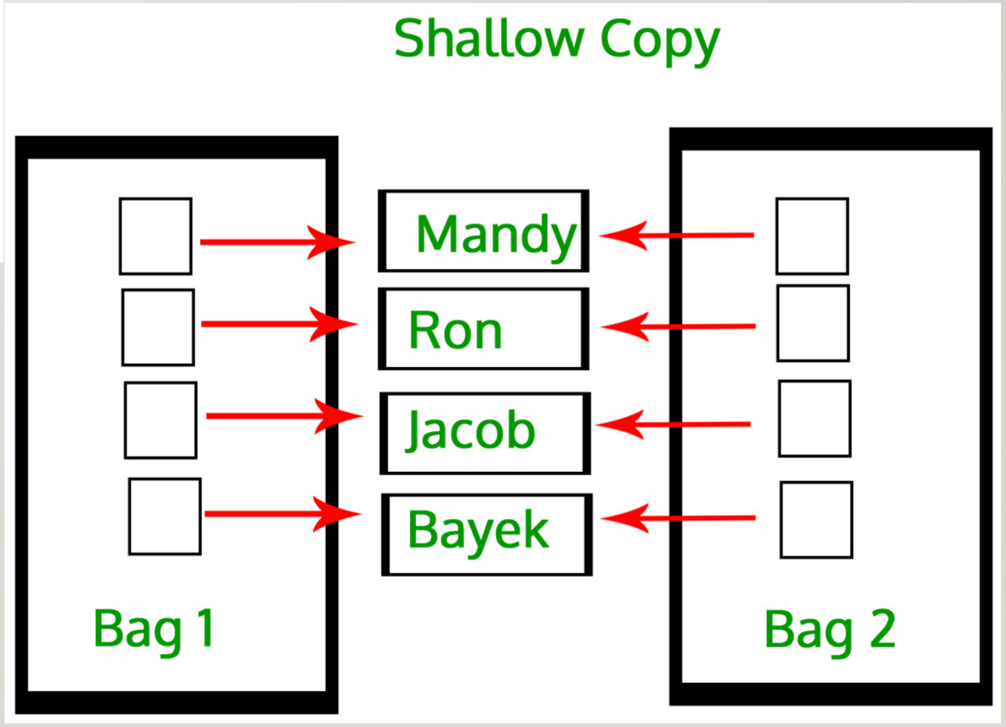
DEEP COPY VS SHALLOW COPY

```
import copy

li1 = [1, 2, [3,5], 4]
li2 = copy.copy(li1)
li3 = copy.deepcopy(li1)

print(li1)
print(li2)
print(li3)

li1[2][0] = 'NEW'

print(li1)
print(li2)
print(li3)
```

- **Deep copy**
  - Copying process occurs **recursively**
  - Construct a new collection object and then recursively populate it with **copies of the objects** in the original
- **Shallow copy**
  - Copying a collection object only
  - Construct a new collection object and then populate it with **reference** to the objects in the original

DEEP COPY VS SHALLOW COPY





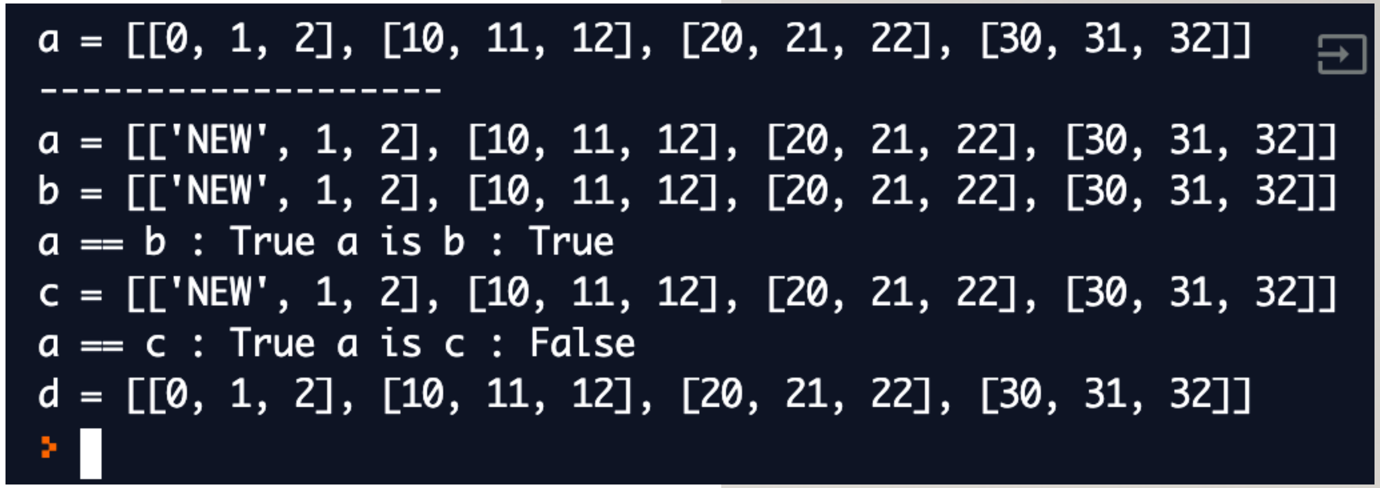
COME BACK TO CREATING 2D LIST

```python
row = 3
col = 4

a = [[0]*col]*row
print("a =", a)

print("----------------")

a[0][0] = "NEW"
print("a =", a)
```

```python
row = 3
col = 4

a = []
for i in range(row):
  a += [[0]*col]

b = [[0]*col for i in range(row)]

print("a =", a)
print("b =", b)
print("----------------")
a[0][0] = "NEW"
b[0][0] = "NEW"
print("a =", a)
print("b =", b)
```

COPYING 2D LIST

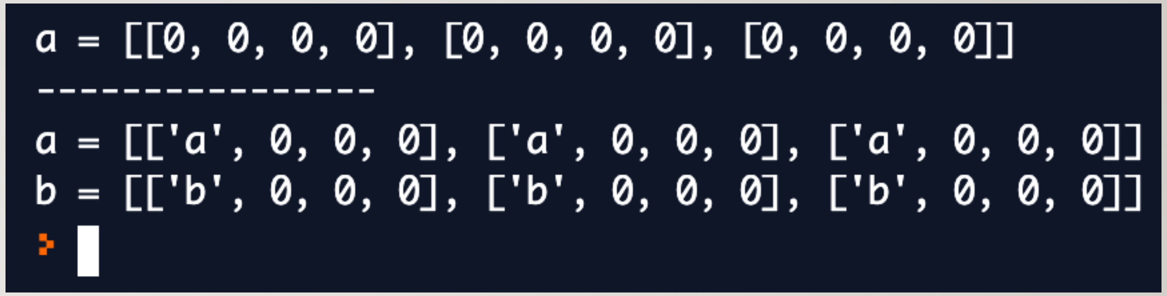
```python
import copy

row, col = 4, 3
a = [[10*i+j for j in range(col)] for i in range(row)]
print("a =", a)

print("--------------------")
b = a
c = a.copy()
d = copy.deepcopy(a);

a[0][0] = "NEW"

print("a =", a)
print("b =", b)
print("a == b :", a==b, "a is b :", a is b)
print("c =", c)
print("a == c :", a==c, "a is c :", a is c)
print("d =", d)
```



```
a = [[0, 1, 2], [10, 11, 12], [20, 21, 22], [30, 31, 32]]
--------------------
a = [['NEW', 1, 2], [10, 11, 12], [20, 21, 22], [30, 31, 32]]
b = [['NEW', 1, 2], [10, 11, 12], [20, 21, 22], [30, 31, 32]]
a == b : True a is b : True
c = [['NEW', 1, 2], [10, 11, 12], [20, 21, 22], [30, 31, 32]]
a == c : True a is c : False
d = [[0, 1, 2], [10, 11, 12], [20, 21, 22], [30, 31, 32]]
>
```

TRY!

```
import copy

row = 3
col = 4

a = [[0]*col]*row
b = copy.deepcopy(a)
print("a =", a)
print("b =", b)

print(“----------------")

a[0][0] = "a"
b[0][0] = "b"
print("a =", a)
print("b =", b)
```



```
a = [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
----------------
a = [['a', 0, 0, 0], ['a', 0, 0, 0], ['a', 0, 0, 0]]
b = [['b', 0, 0, 0], ['b', 0, 0, 0], ['b', 0, 0, 0]]
```

- Even after deepcopy you can see the result above
- Why not [[‘b’, 0, 0. 0], [0, 0, 0, 0], [0, 0, 0, 0]]?

LOOPING OVER 2D LIST

- Using nested loop

```
row = 4
col = 3

a = [[0]*col for i in range(row)]
print("a =", a)

for i in range(row):
  for j in range(col):
    a[i][j] = 10*i+j

print("----------------")
print("a =", a)
```

MORE ABOUT MULTI-DIMENSIONAL LIST

- In 2d list, each element does not have to be the same size

```
a = [[1], [2,3], [4, 5, 6], ["python", "programming"]]
print(a)
```

- 3d list : a list that has one or more 2d lists as its element

```
a = [[1, 2], [3, 4]]
b = [[5, 6, 7], [8, 9]]
c = [[10]]

d = [a, b, c]
print("d =", d)

for i in range(len(d)):
  for j in range(len(d[i])):
    for k in range(len(d[i][j])):
      print("d[%d][%d][%d] =" % (i, j, k), d[i][j][k])
```

QUESTION?