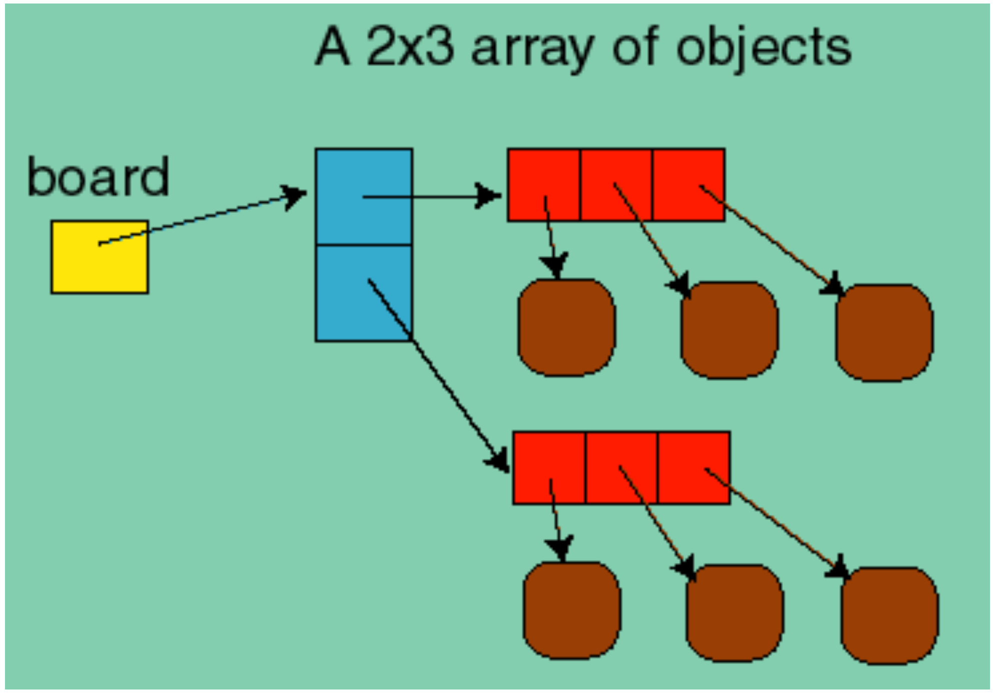
# 15-112
# Fundamentals of Programming

## Week 3 - Lecture 1:
## “2-dimensional” lists



June 5, 2017

## Tricky thing about 2d lists

1d list: references to **immutable** objects.

Aliases of elements not a problem.

2d list: references to **mutable** objects.

We must be careful about aliases of elements !!

# “Weird” Example 1

```
a = [1, 2, 3]

b = copy.copy(a)

b[0] = 0

print(a)            [1, 2, 3]

print(b)            [0, 2, 3]
```

---

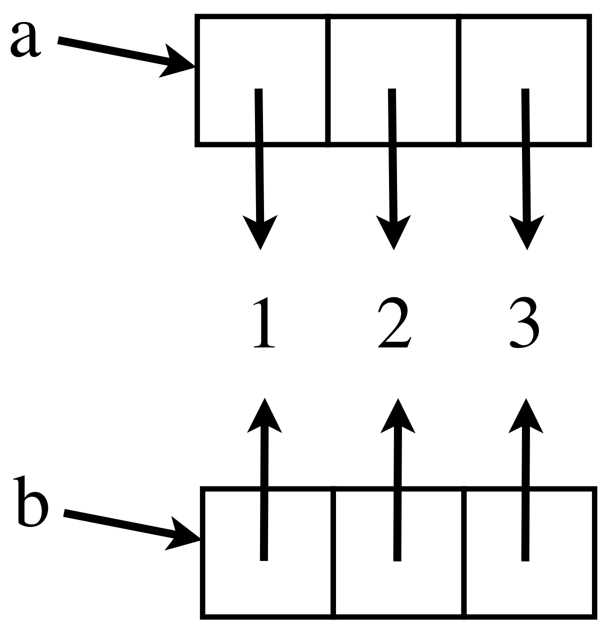
```
a = [[1, 2, 3], [4, 5, 6]]

b = copy.copy(a)

b[0][0] = 0

print(a)            [ [0, 2, 3], [4, 5, 6] ]

print(b)            [ [0, 2, 3], [4, 5, 6] ]
```

# “Weird” Example 2

```
a = [ [0]*2 ]*3
print(a)                  [ [0, 0], [0, 0], [0, 0] ]

a[0][0] = 9

print(a)                  [ [9, 0], [9, 0], [9, 0] ]
```

# Understanding Example 1

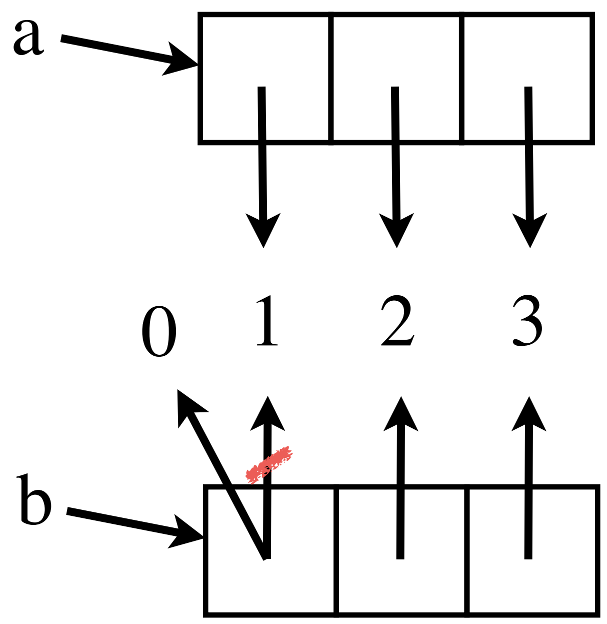
```
a = [1, 2, 3]

b = copy.copy(a)

b[0] = 0

print(a[0])

print(b[0])
```



Making a copy of the references.

# Understanding Example 1

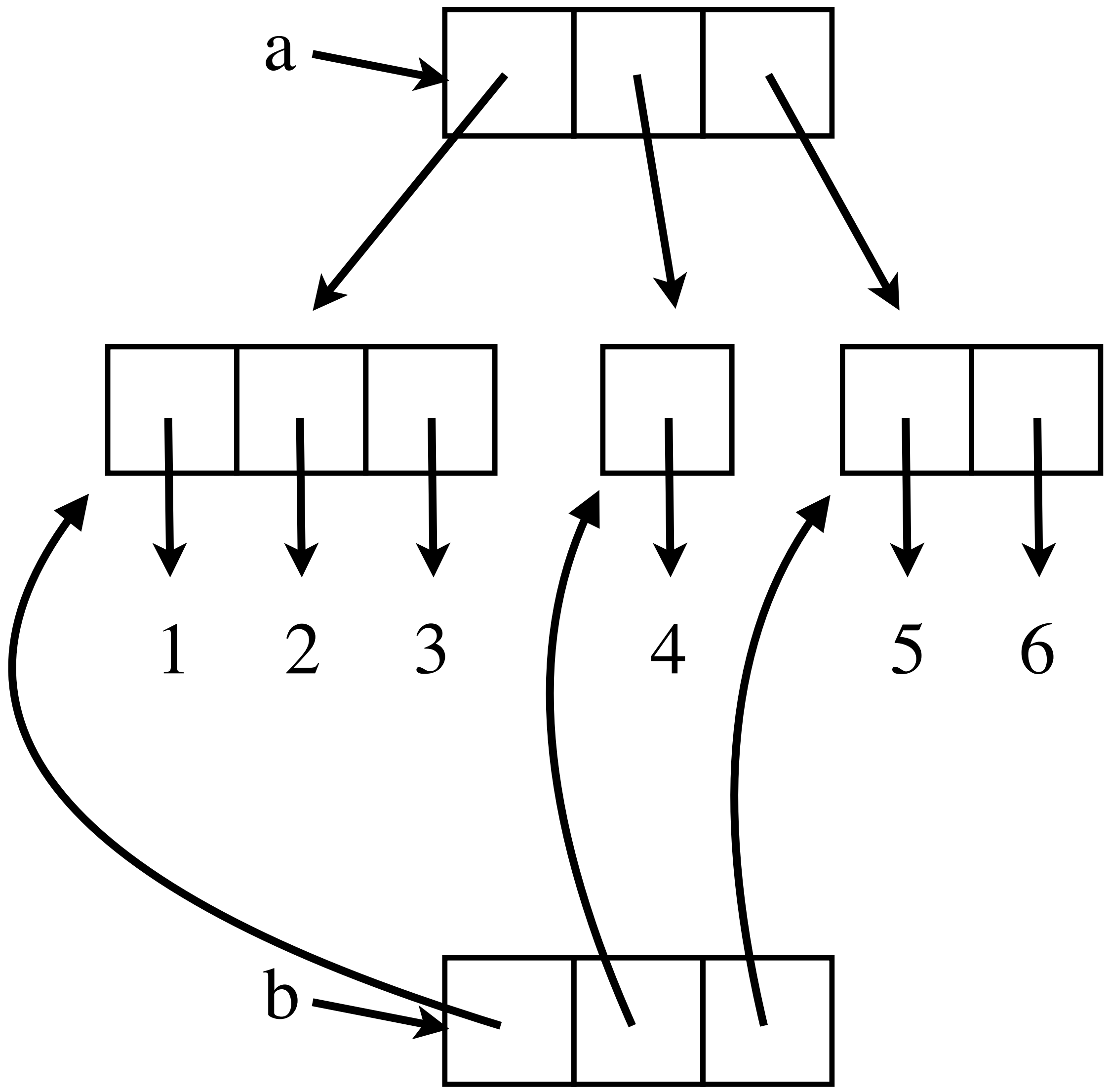
```
a = [1, 2, 3]

b = copy.copy(a)

b[0] = 0

print(a[0])

print(b[0])
```



Making a copy of the references.

# Understanding Example 1

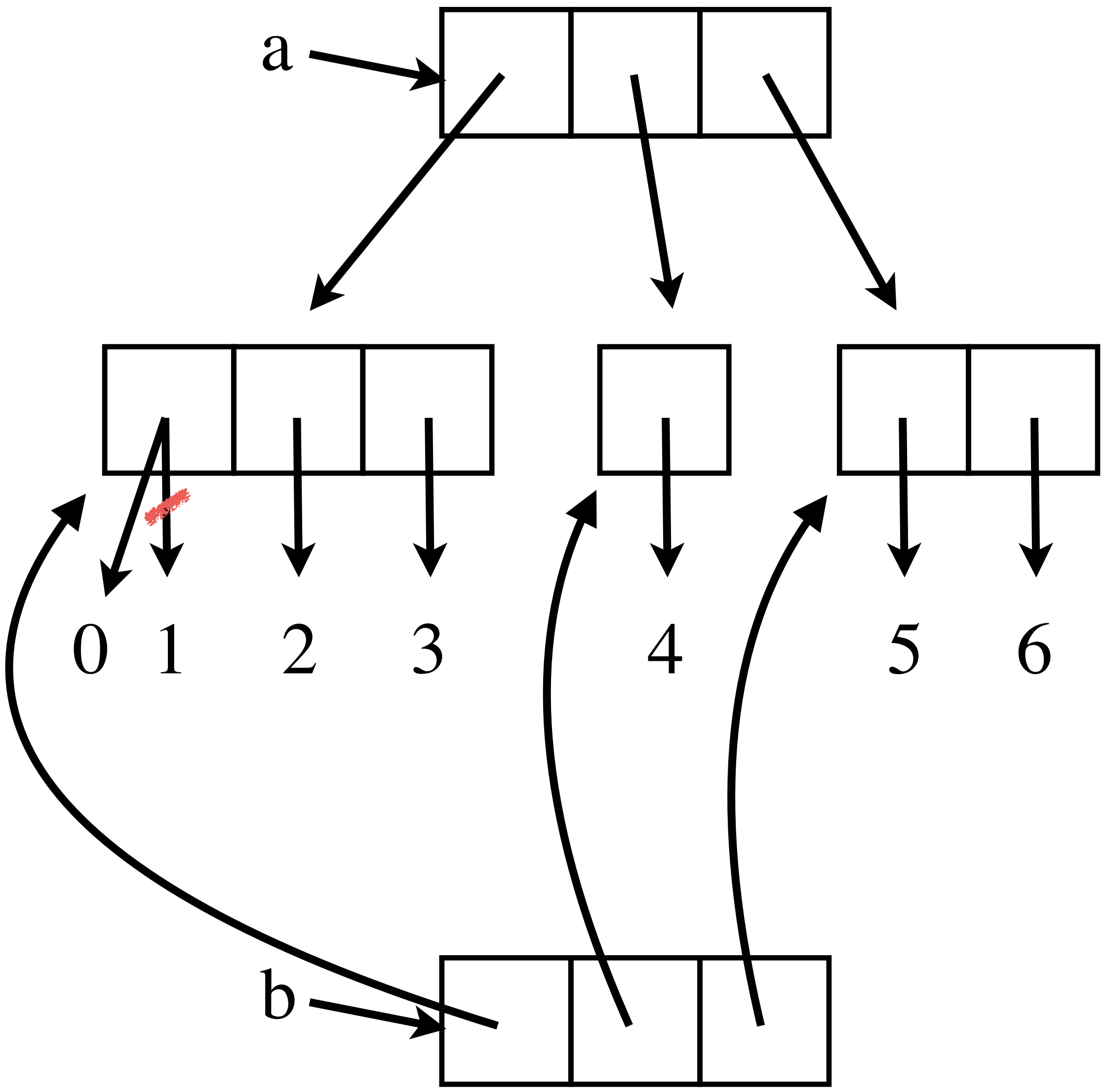
```
a = [[1, 2, 3], [4], [5, 6]]

b = copy.copy(a)

b[0][0] = 0

print(a[0][0])

print(b[0][0])
```



# Understanding Example 1

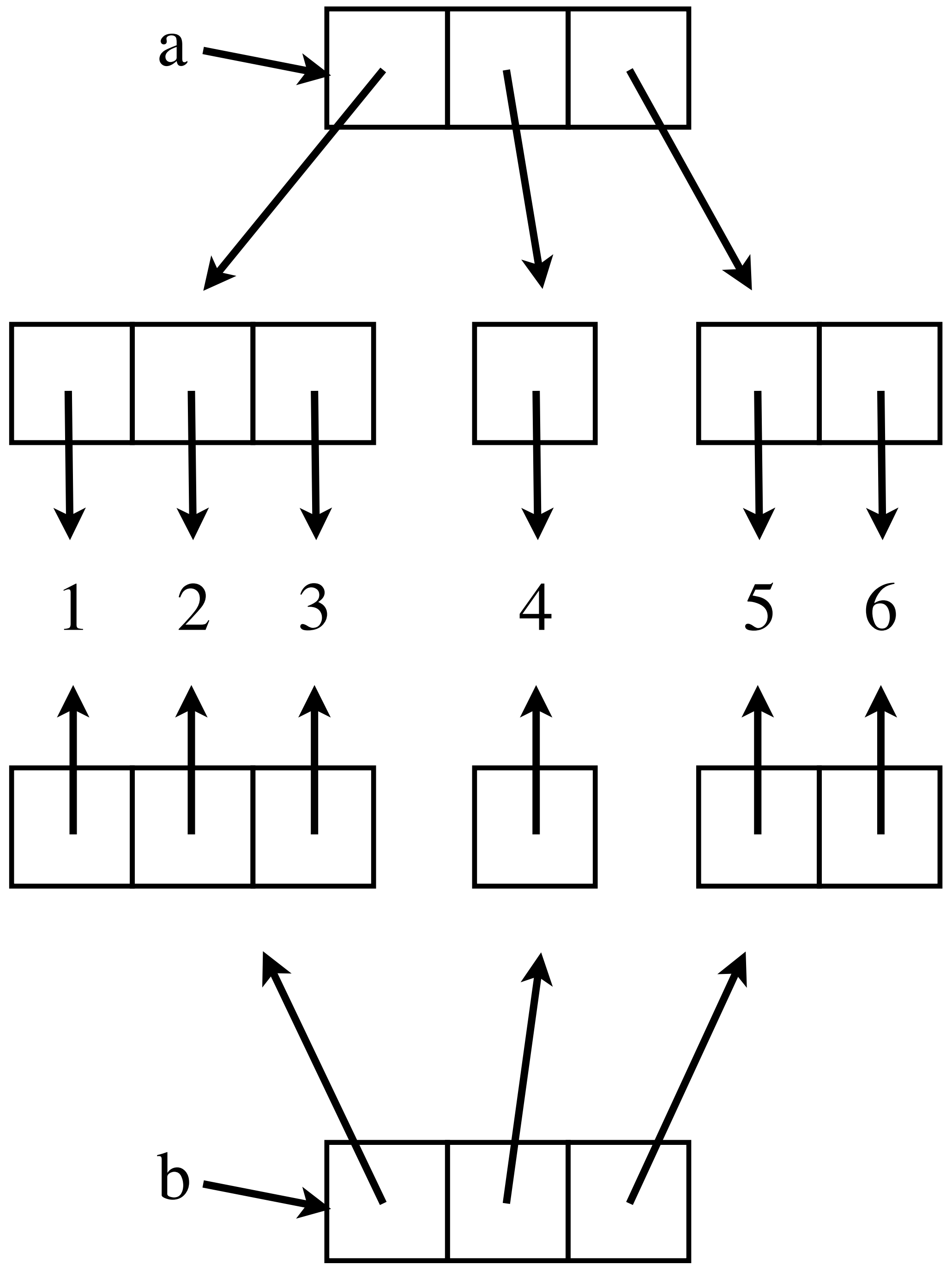
```
a = [[1, 2, 3], [4], [5, 6]]

b = copy.copy(a)

b[0][0] = 0

print(a[0][0])

print(b[0][0])
```

**Shallow copy**



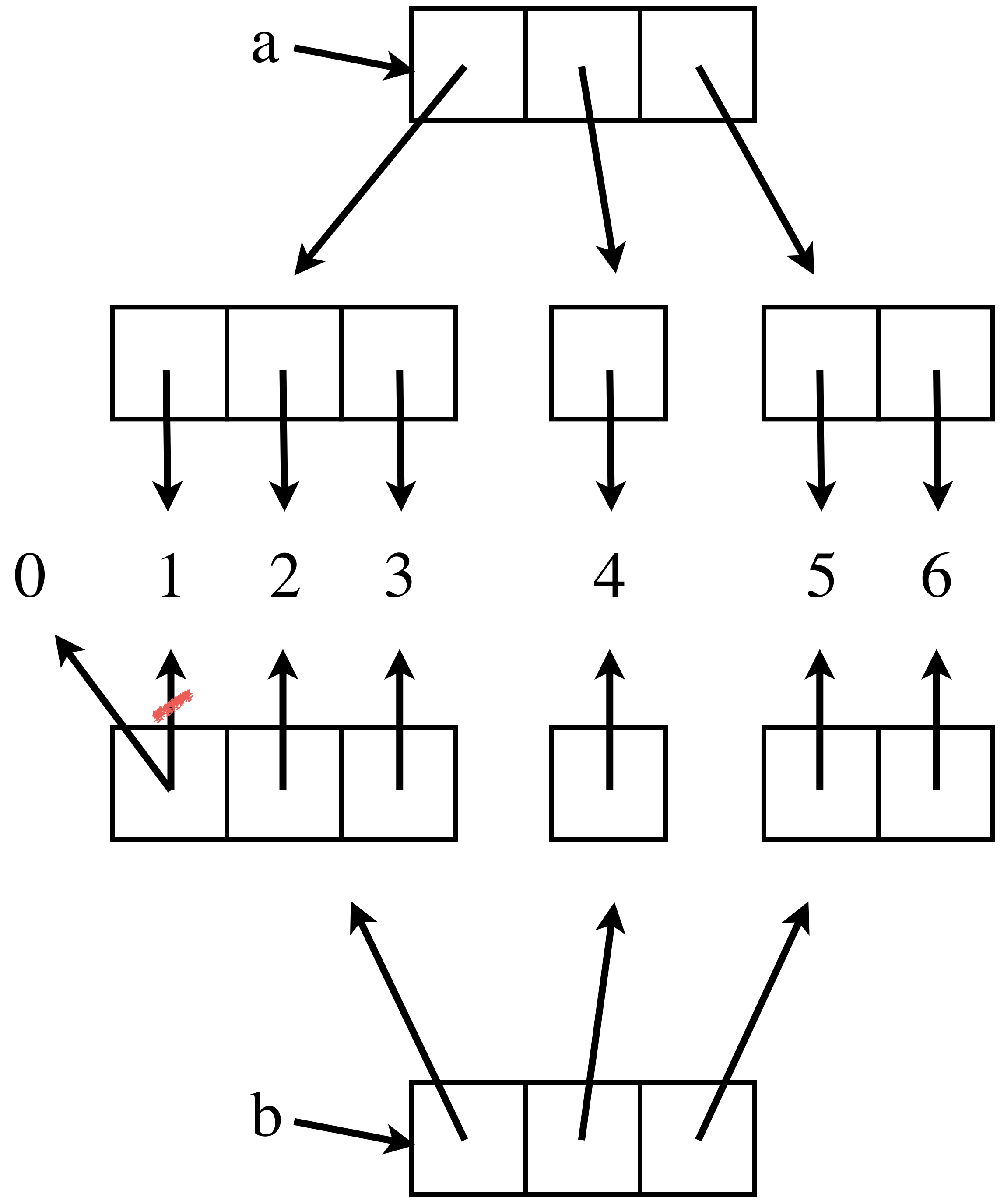
# Understanding Example 1

```
a = [[1, 2, 3], [4], [5, 6]]

b = copy.deepcopy(a)

b[0][0] = 0

print(a[0][0])

print(b[0][0])
```



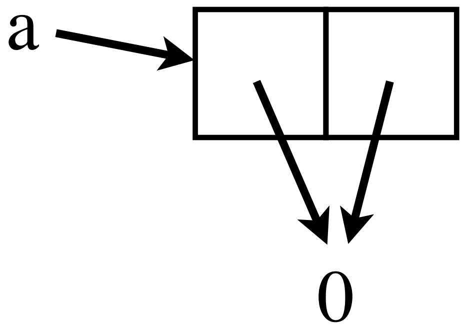
# Understanding Example 1

```
a = [[1, 2, 3], [4], [5, 6]]

b = copy.deepcopy(a)

b[0][0] = 0

print(a[0][0])

print(b[0][0])
```



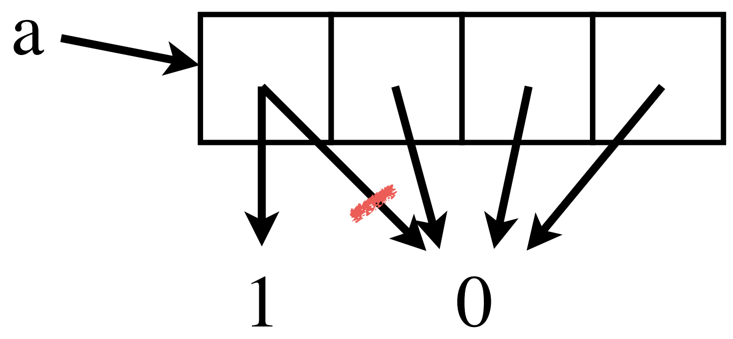
# Understanding Example 2

```
a = [0]*2
```



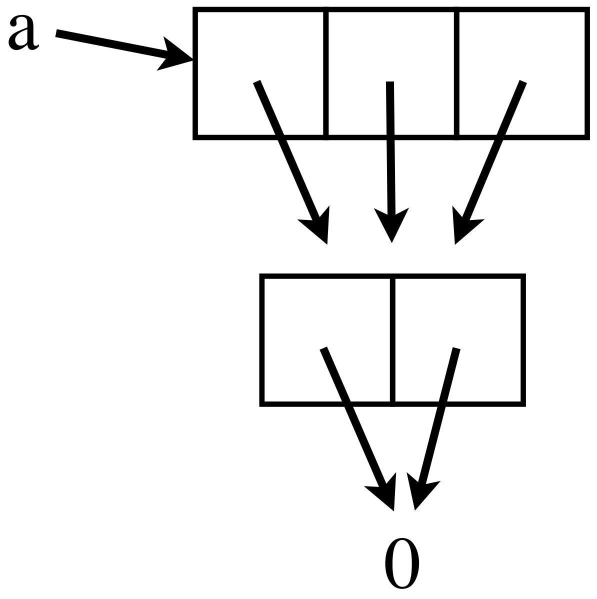
# Understanding Example 2

```
a = [0]*4

a[0] = 1
```



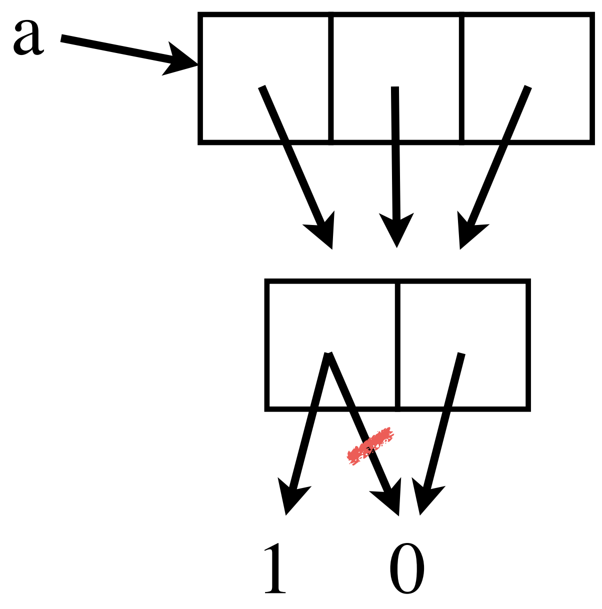
## Understanding Example 2

```
# Create a 3 by 2 list
a = [ [0]*2 ]*3
```



## Understanding Example 2

```
# Create a 3 by 2 list
a = [ [0]*2 ]*3
[ [0, 0], [0, 0], [0, 0] ]

a[0][0] = 1

print(a)

[ [1, 0], [1, 0], [1, 0] ]
```



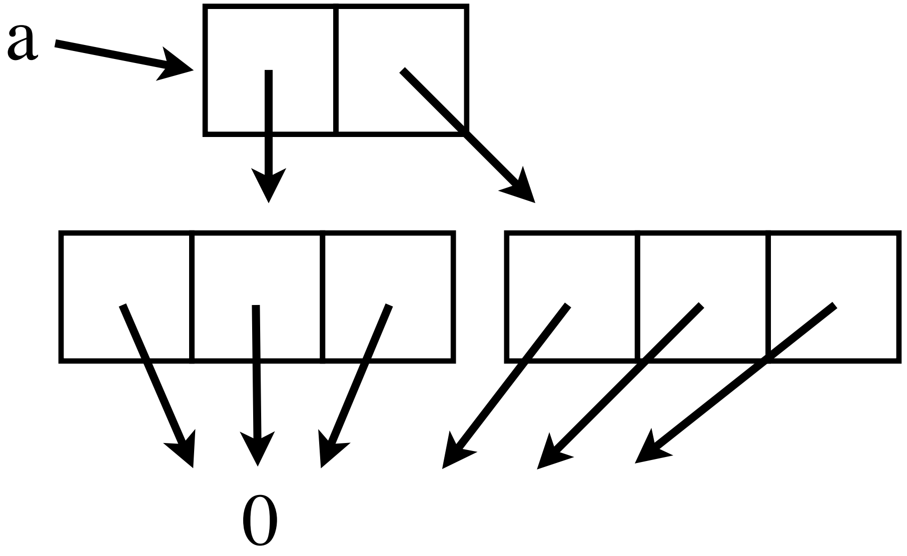
a[0], a[1], and a[2] are aliases !

* makes a shallow copy !

# Creating a *rows* by *cols* 2d list

```
rows = 2
cols = 3
a = [ ]
for row in range(rows):
    a += [[0]*cols]
```

```
a += [[0, 0, 0]]
a += [[0, 0, 0]]
```



## Creating a *rows* by *cols* 2d list

Define a function for this task.

```
def make2dList(rows, cols):
    a = []
    for row in range(rows):
        a += [[0]*cols]
    return a
```

## One more important thing

# Create a 3 by 2 list

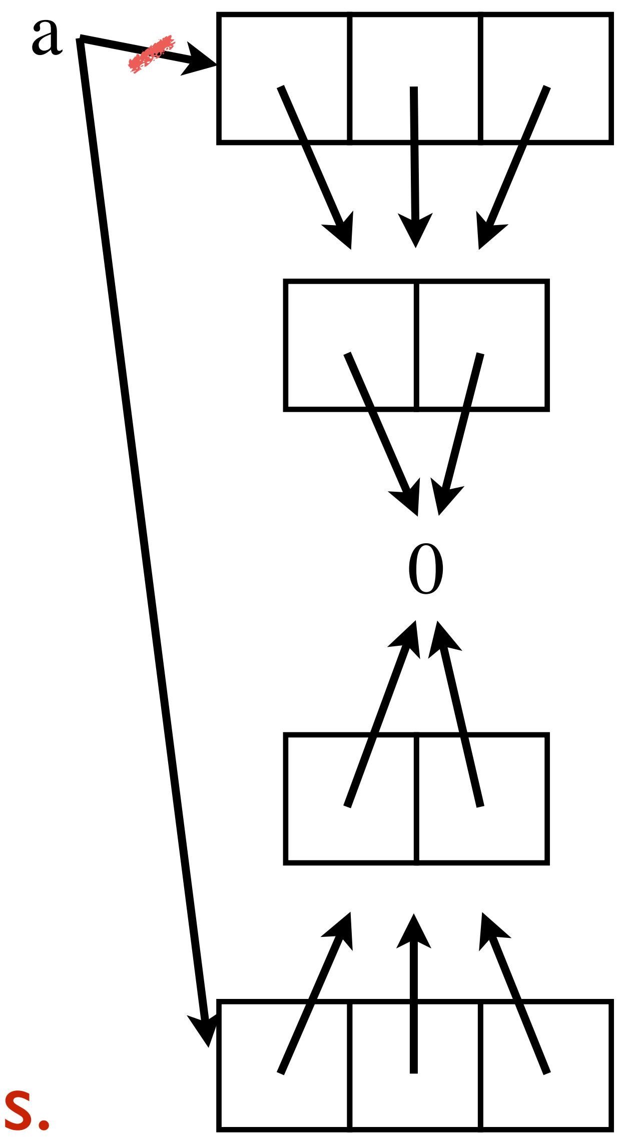
```
a = [ [0]*2 ]*3
```

Trying to break aliasing with deepcopy:

```
a = copy.deepcopy(a)
```

deepcopy preserves alias structure !!

see myDeepCopy in the notes.



## Rules

Use * only on the first level (with immutable elements)

- creates aliases

Never use copy with 2d lists.

- creates aliases
- ok to use with 1d lists since elements are immutable.

Remember: deepcopy does not break alias structure
within the list.

## 3d Lists

```
a1 = [ [ 1, 2 ],
       [ 3, 4 ] ]

a2 = [ [ 5, 6, 7 ],
       [ 8, 9 ]     ]

a3 = [ [ 10 ] ]
```

### 3d list:

```
a = [ a1, a2, a3 ]
```

### 4d list:

```
a = [ a, a ]
```

# 3d Lists

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

## Printing elements of 3d lists:

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