Digital World (2018)

Week 4, S2: Copying (Nested) Lists; Dictionaries

Chris Poskitt

SINGAPORE UNIVERSITY OF
TECHNOLOGY AND DESIGN

3. *Loops:* Write a function named `find_average` that takes in a list of lists as an input. Each sublist contains numbers. The function returns a list of the averages of each sublist, and the overall average. If the sublist is empty, take the average to 0.0.

   For example, if the input list is $[[3,4],[5,6,7],[-1,2,3]]$, the program returns the list $[3.5, 6.0, 1.333]$, and the overall average 3.625, calculated by summing all the numbers in all the sublists and dividing this total sum by the total count of all the numbers.

```
>>> ans=find_average([[3,4],[5,6,7],[-1,2,8]])
>>> print(ans)
([3.5, 6.0, 3.0], 4.25)

>>> ans=find_average([[13.13,1.1,1.1],[],[1,1,0.67]])
>>> print(ans)
```

Cohort Session Q1(a)

b.socrative.com, POSKITT5665

Specify the value of `x[0]` at the end the following code snippet.

```
x=[1,2,3]
x[0]=0
y=x
y[0]=1
```

Cohort Session Q1(a)

b.socrative.com, POSKITT5665

Specify the value of `x[0]` at the end the following code snippet.

```
x=[1,2,3]
x[0]=0
y=x
y[0]=1
```

this is called **aliasing**

Cohort Session Q1(c)

b.socrative.com, POSKITT5665

What is the value of `a[0][0][0][0]` after executing the following code snippet? Write 'E' if there are any errors.

```
x=[1,2,3]
y=[x]
a=[y,x]
y[0][0] = (1,2)
```

Cohort Session Q1(c)

b.socrative.com, POSKITT5665

What is the value of `a[0][0][0][0]` after executing the following code snippet? Write 'E' if there are any errors.

```
x=[1,2,3]
y=[x]
a=[y,x]
y[0][0] = (1,2)
```

this is a **tuple**

i.e. an ordered sequence of values; like a list but immutable

(where have we seen tuples before?)

Cohort Session Q1(d)

b.socrative.com, POSKITT5665

Specify the values of expressions (a), (b), (c) and (d) in the following code.

```
x=[1,2,3]
y1=[x,0]
y2=y1[:]
y2[0][0]=0
y2[1]=1
y1[0][0]  # (a)
y1[1]  # (b)
y2[0][0]  # (c)
y2[1]  # (d)
```

Cohort Session Q1(d)

b.socrative.com, POSKITT5665

Specify the values of expressions (a), (b), (c) and (d) in the following code.

```
x=[1,2,3]
y1=[x,0]
y2=y1[:]
y2[0][0]=0
y2[1]=1
y1[0][0]  # (a)
y1[1]  # (b)
y2[0][0]  # (c)
y2[1]  # (d)
```

this is called a **shallow copy**

we create a new list **y2** *by copying the references of* **y1**

Shallow vs. deep copy

```
x = [1, 2, 3]
y1 = [x, 0]
y2 = y1[:]
```

```
import copy
x = [1, 2, 3]
y1 = [x, 0]
y2 = copy.deepcopy(y1)
```

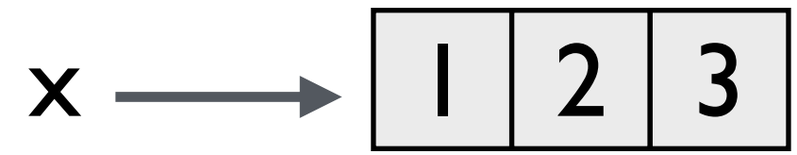
Shallow vs. deep copy

```
x = [1, 2, 3]
y1 = [x, 0]
y2 = y1[:]
```

```
import copy
x = [1, 2, 3]
y1 = [x, 0]
y2 = copy.deepcopy(y1)
```

Shallow vs. deep copy

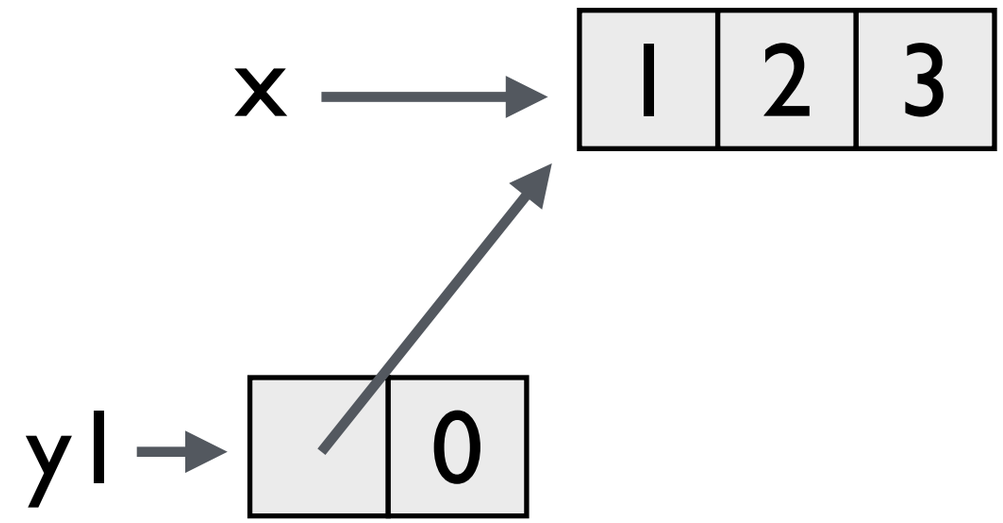
```
x = [1, 2, 3]
y1 = [x, 0]
y2 = y1[:]
```




```
import copy
x = [1, 2, 3]
y1 = [x, 0]
y2 = copy.deepcopy(y1)
```

Shallow vs. deep copy

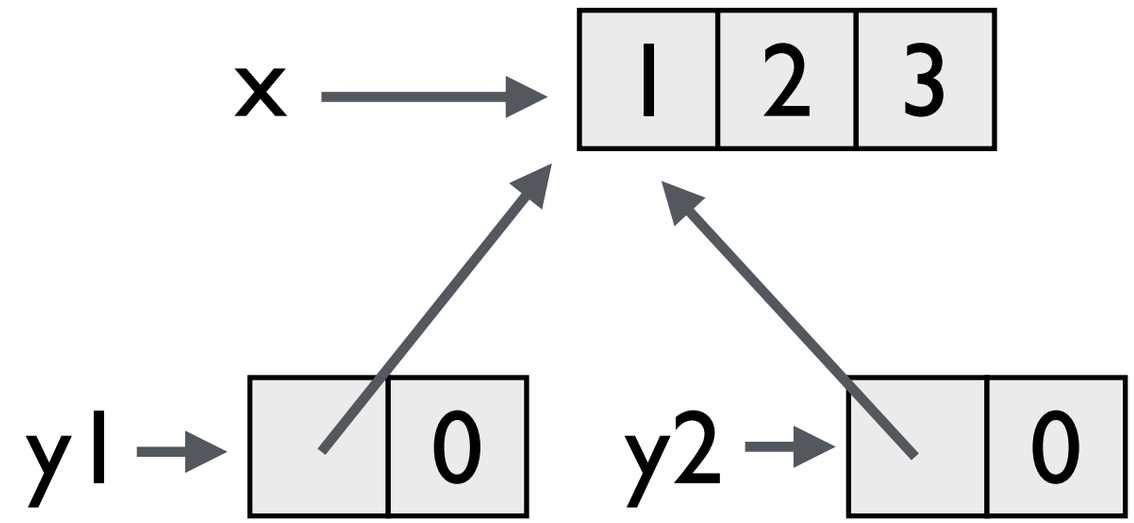
```
x = [1, 2, 3]
y1 = [x, 0]
y2 = y1[:]
```




```
import copy
x = [1, 2, 3]
y1 = [x, 0]
y2 = copy.deepcopy(y1)
```

Shallow vs. deep copy

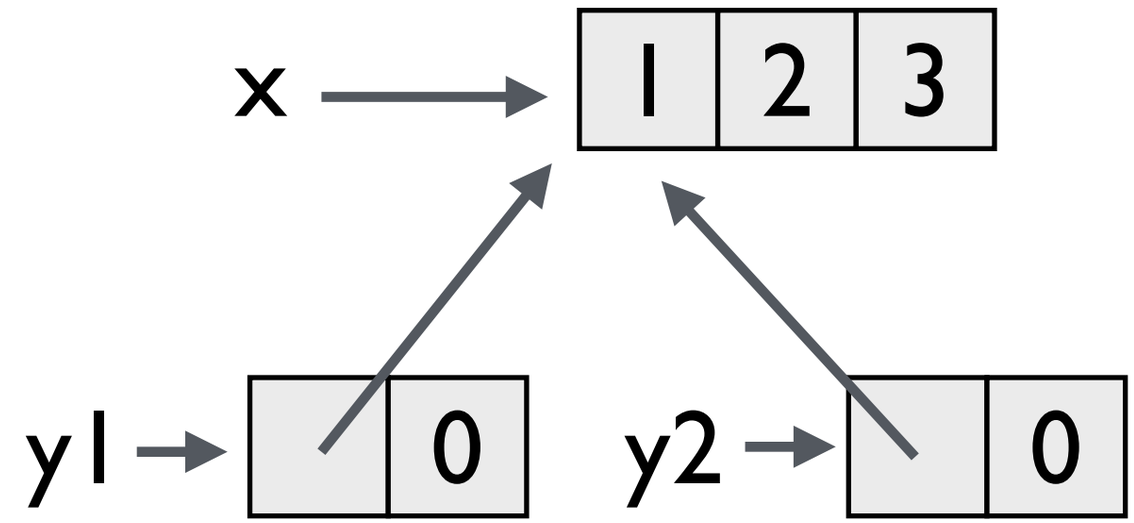
```
x = [1, 2, 3]
y1 = [x, 0]
y2 = y1[:]
```




```
import copy
x = [1, 2, 3]
y1 = [x, 0]
y2 = copy.deepcopy(y1)
```

Shallow vs. deep copy

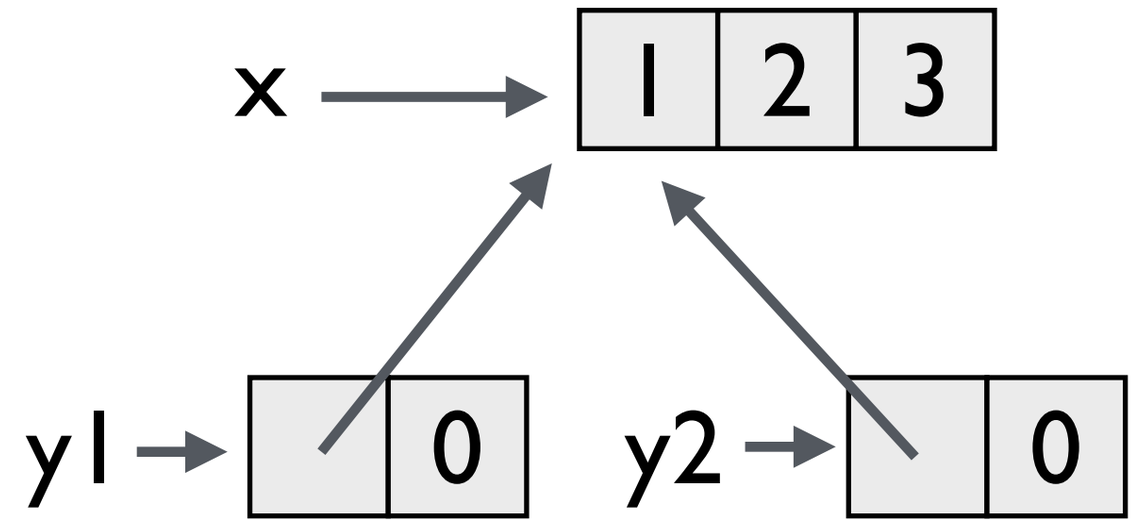
```
x = [1, 2, 3]
y1 = [x, 0]
y2 = y1[:]
```




```
import copy
x = [1, 2, 3]
y1 = [x, 0]
y2 = copy.deepcopy(y1)
```

Shallow vs. deep copy

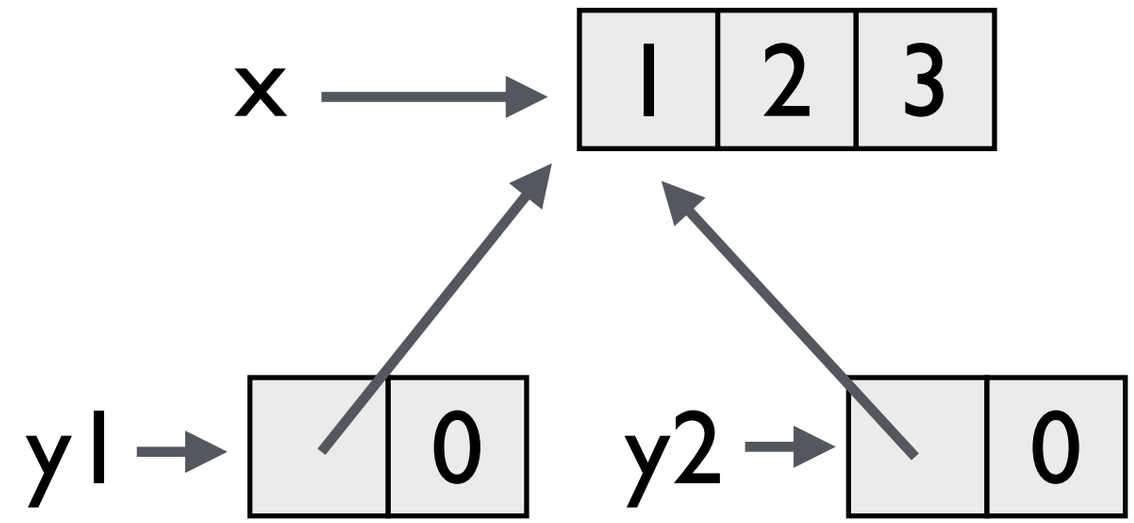
```
x = [1, 2, 3]
y1 = [x, 0]
y2 = y1[:]
```



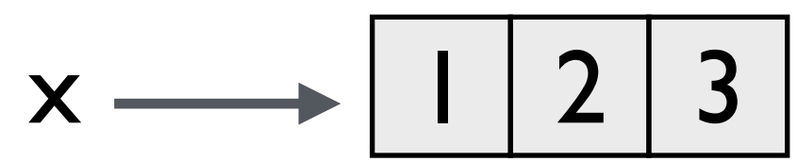

```
import copy
x = [1, 2, 3]
y1 = [x, 0]
y2 = copy.deepcopy(y1)
```

Shallow vs. deep copy

```
x = [1, 2, 3]
y1 = [x, 0]
y2 = y1[:]
```

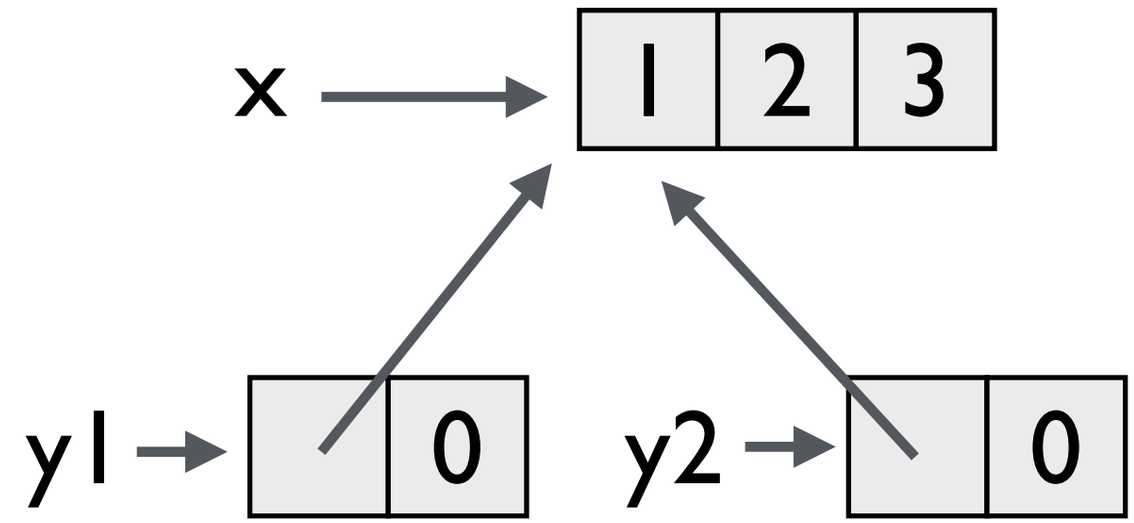



```
import copy
x = [1, 2, 3]
y1 = [x, 0]
y2 = copy.deepcopy(y1)
```

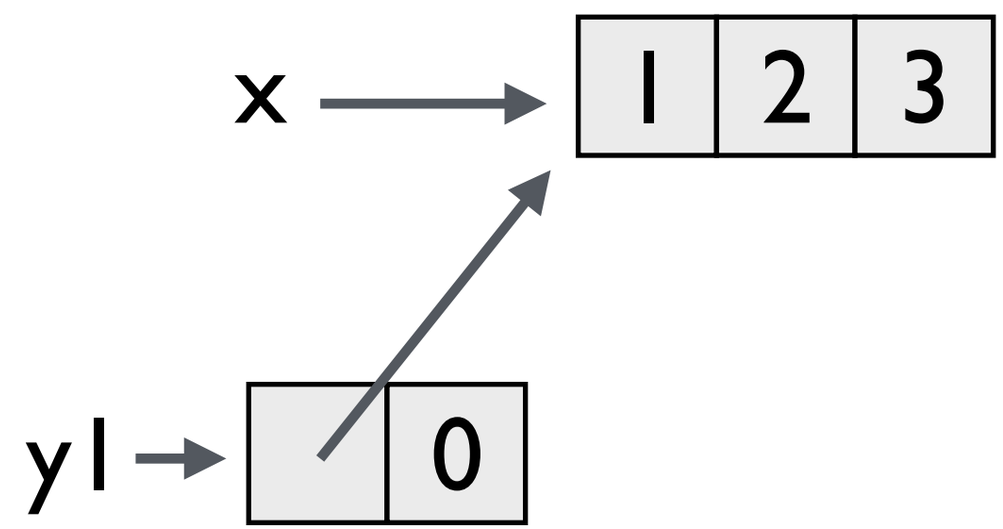


Shallow vs. deep copy

```
x = [1, 2, 3]
y1 = [x, 0]
y2 = y1[:]
```

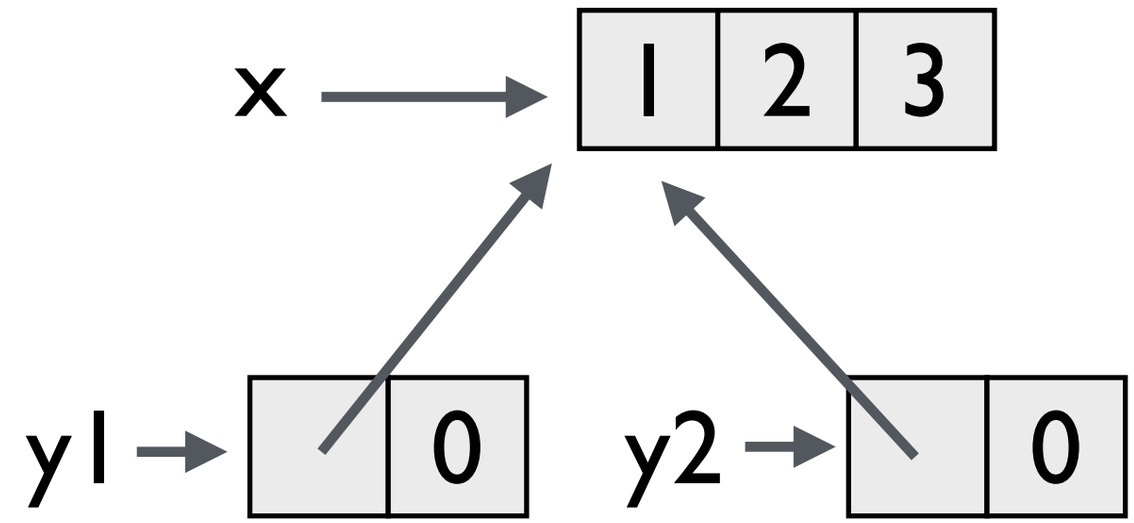



```
import copy
x = [1, 2, 3]
y1 = [x, 0]
y2 = copy.deepcopy(y1)
```

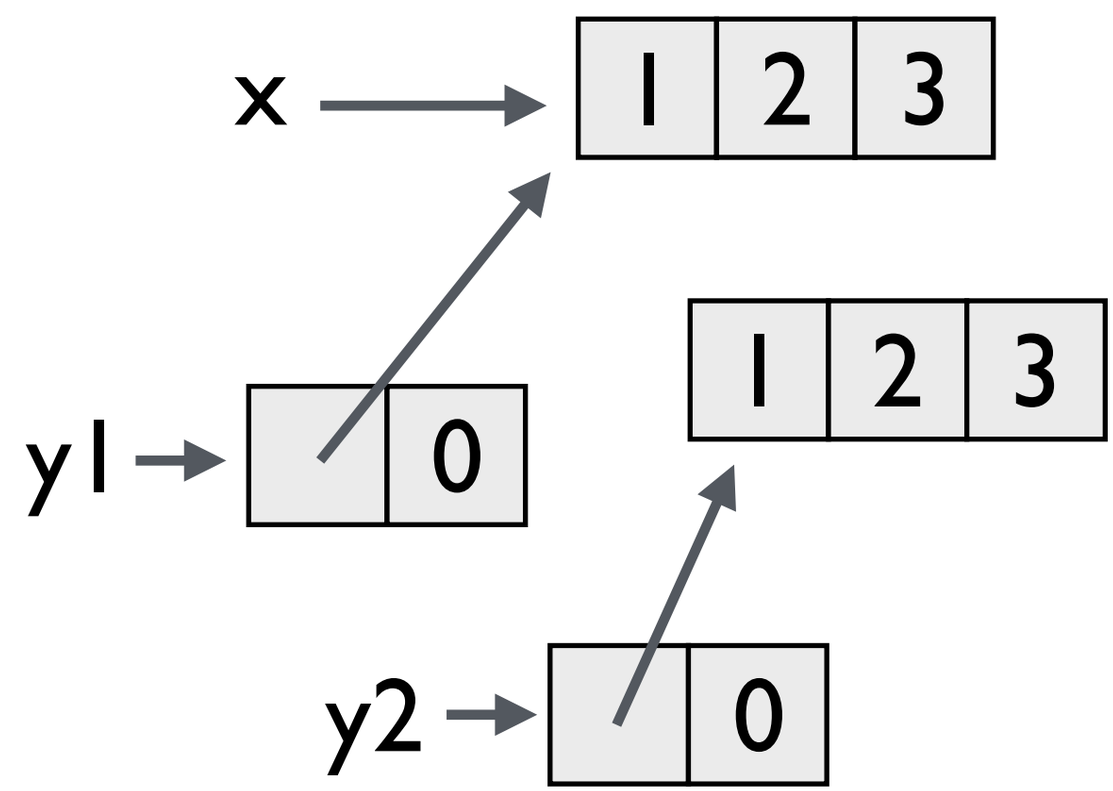


Shallow vs. deep copy

```
x = [1, 2, 3]
y1 = [x, 0]
y2 = y1[:]
```




```
import copy
x = [1, 2, 3]
y1 = [x, 0]
y2 = copy.deepcopy(y1)
```



Deep copy preserves the graphical structure

```
import copy
x = [1,2]
y1 = [x, x]
y2 = copy.deepcopy(y1)
```

Deep copy preserves the graphical structure

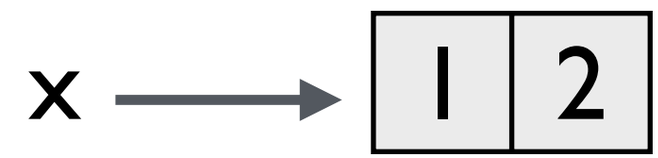
```
import copy
x = [1,2]
y1 = [x, x]
y2 = copy.deepcopy(y1)
```

Deep copy preserves the graphical structure

```
import copy
x = [1,2]
y1 = [x, x]
y2 = copy.deepcopy(y1)
```

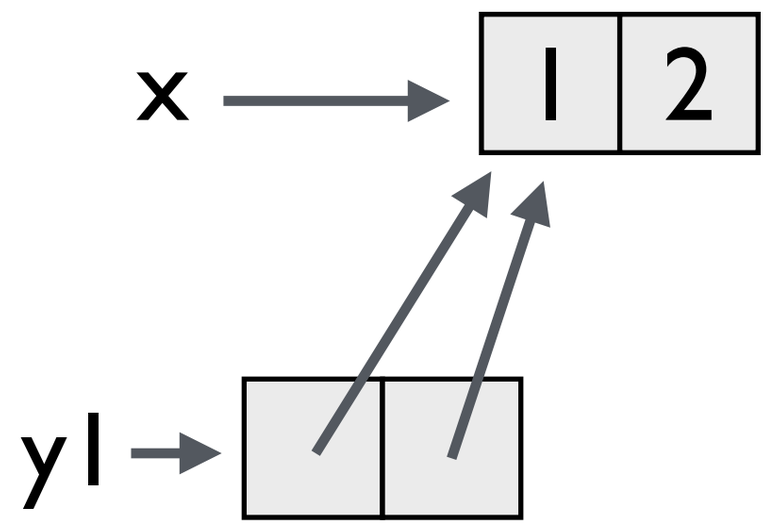
Deep copy preserves the graphical structure

```
import copy
x = [1,2]
y1 = [x, x]
y2 = copy.deepcopy(y1)
```



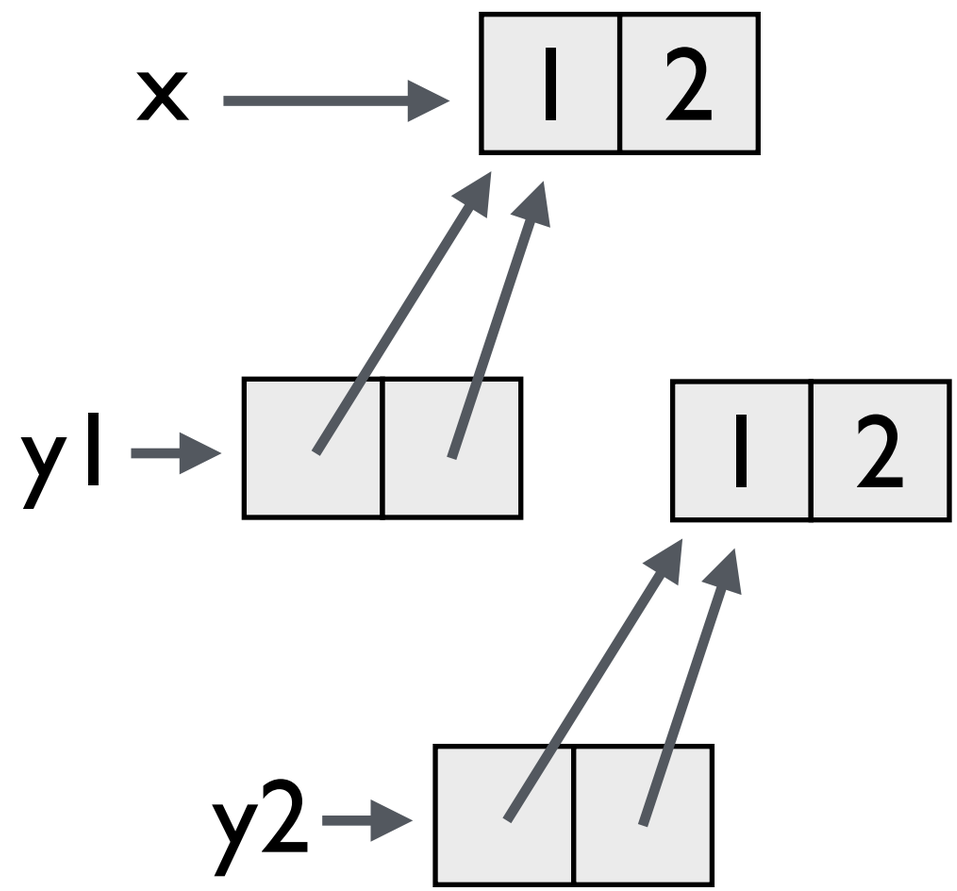
Deep copy preserves the graphical structure

```
import copy
x = [1,2]
y1 = [x, x]
y2 = copy.deepcopy(y1)
```



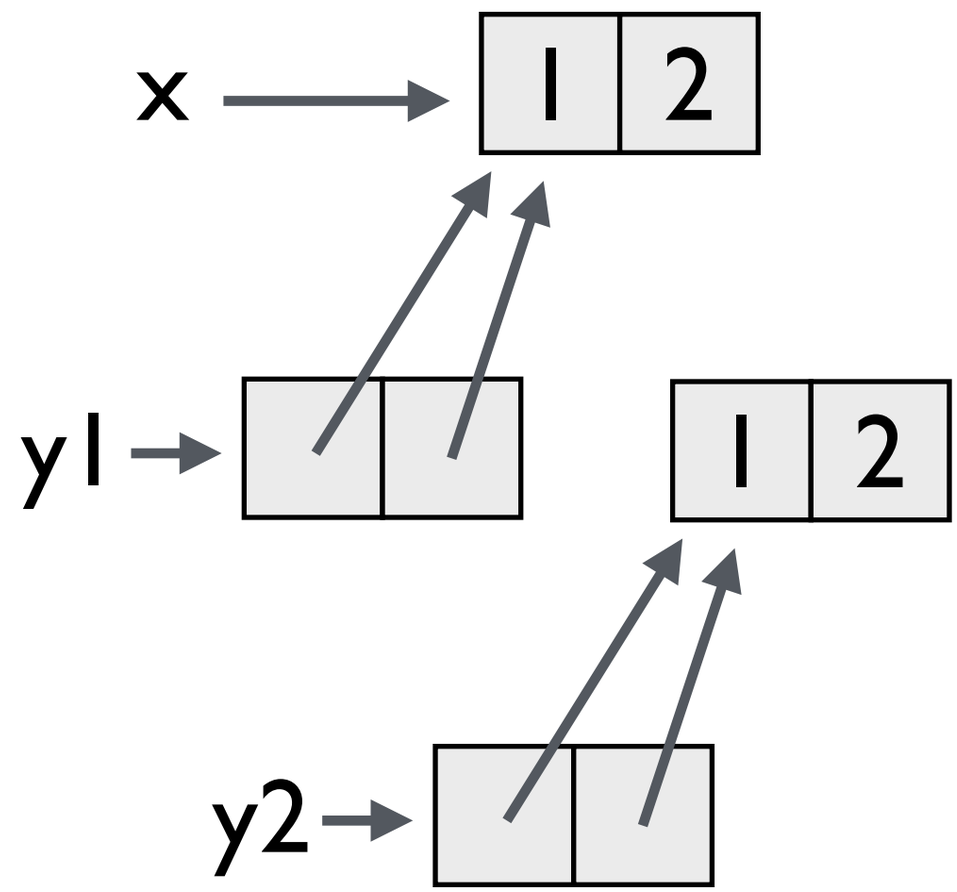
Deep copy preserves the graphical structure

```
import copy
x = [1,2]
y1 = [x, x]
y2 = copy.deepcopy(y1)
```



Deep copy preserves the graphical structure

```
import copy
x = [1,2]
y1 = [x, x]
y2 = copy.deepcopy(y1)
```



still uncertain? *try animating your own programs at:*
http://pythontutor.com/visualize.html

(b) Specify the value of `x[0]` after the following code snippet.

```
x=[1,2,3]
def f(l):
  l[0]='a'
f(x)
```

(e) Specify the values of expressions (a), (b), (c) and (d) in the following code.

```
import copy
x=[1,2,3]
y1=[x,0]
y2=copy.deepcopy(y1)
y2[0][0]=0
y2[1]=1
y1[0][0]  # (a)
y1[1]  # (b)
y2[0][0]  # (c)
y2[1]  # (d)
```

(f) What is the value of `l` after steps (a), (b), (c) and (d) below?

```
l=[1,2,3]
l[2:3]=4  # (a)
l[1:3]=[0]  # (b)
l[1:1]=1  #(c)
l[2:]=[]  # (d)
```

From lists to dictionaries

From lists to dictionaries

- a list maps indices to values
- a dictionary maps (immutable) keys to values

From lists to dictionaries

- a list maps indices to values

- a dictionary maps (immutable) keys to values

d =

'Rowan Atkinson'	→	*'+44 1356 345867'*
'John Cleese'	→	*'+44 1904 534534'*
'Cersei Lannister'	→	*'+353 112 112'*
'Chris Poskitt'	→	*'+65 8888 8888'*

From lists to dictionaries

- a list maps indices to values
- a dictionary maps (immutable) keys to values

d =

'Rowan Atkinson'	→	*'+44 1356 345867'*
'John Cleese'	→	*'+44 1904 534534'*
'Cersei Lannister'	→	*'+353 112 112'*
'Chris Poskitt'	→	*'+65 8888 8888'*

d['key'] = value d.keys() d.values() d.items()

d.copy() copy.deepcopy(d) x in d.keys() x not in d.values()

5. *Dictionary:* Write a function named `get_details` that takes in a name, a key search, and a list. The list contains a list of phone book entries, where each entry is a dictionary. For example

```
>>> phonebook=[{'name':'Andrew', 'mobile_phone':9477865, '
    office_phone':6612345, 'email':'andrew@sutd.edu.sg'},{'name':'
    Bobby','mobile_phone':8123498, 'office_phone':6654321, 'email': '
    bobby@sutd.edu.sg'}]
```

The function returns the value of the key search requested for that particular name. It should return `None` if either the name or the key is not found. For example:

```
>>> print(get_details('Andrew', 'mobile_phone', phonebook))
9477865
>>> print(get_details('Andrew', 'email', phonebook))
andrew@sutd.edu.sg
>>> print(get_details('Bobby', 'office_phone', phonebook))
6654321
>>> print(get_details ('Chokey', 'office_phone', phonebook))
None
```

6. *Dictionary:* Write a function named `get_base_counts` that takes a DNA string as an input. The input string consists of letters A, C, G, and T (upper case only). The function returns the count of the number of times each of the four letters A, C, G, and T appear in the input string, in the form of a dictionary. For any input string with letters other than A, C, T, and G, the function will return 'The input DNA string is invalid'. **Test Cases:**

Test case 1

Input: 'AACCGT'

Output: {'A': 2, 'C': 2, 'G': 1, 'T': 1}

Test case 2

Input: 'AACCG'

Output: {'A': 2, 'C': 2, 'G': 1, 'T': 0}

Test case 3

Input: 'AAB'

Output: 'The input DNA string is invalid'

Test case 4

Input: 'AaCaGT'

Output: 'The input DNA string is invalid'

Summary

- several ways to copy a list: aliasing vs. shallow vs. deep
- tuples are immutable sequences of values
- dictionaries map immutable keys to values
- their keys, values, and (key, value) pairs can be iterated over