CS 1110:

Introduction to Computing Using Python

Lecture 7

Objects

[Andersen, Gries, Lee, Marschner, Van Loan, White]

Lecture 7 Announcements

- Please check the ***end*** of the Lecture 6 slides (slides 25-29) for many announcements:

http://www.cs.cornell.edu/courses/cs1110/2017sp/lectures/02-14-17/presentation-06.pdf

- Incorrect link for how to break up long lines in Section 10 of Assignment 1. Watch course website for announcements about A1:

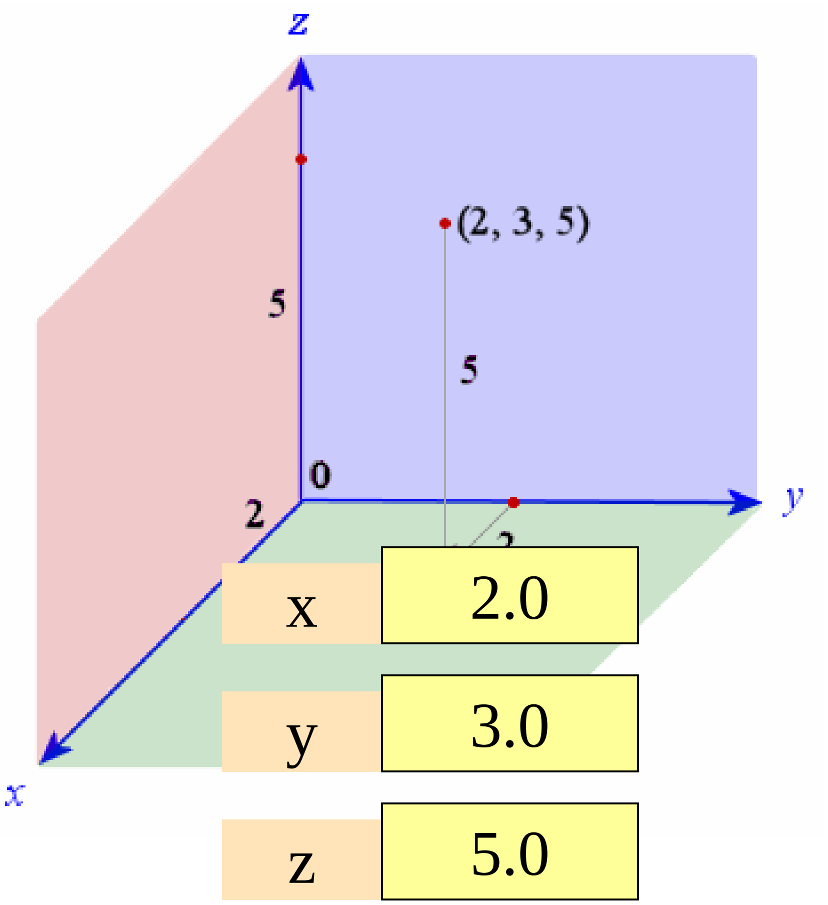
http://www.cs.cornell.edu/courses/cs1110/2017sp/announcements.php

Review: Types

- Type **int**:
  - **Values**: integers
  - **Ops**: +, –, *, /, %, **
- Type **float**:
  - **Values**: real numbers
  - **Ops**: +, –, *, /, **
- Type **bool**:
  - **Values**: **True** and **False**
  - **Ops**: not, and, or
- Type **str**:
  - **Values**: string literals
    - Double quotes: "abc"
    - Single quotes: 'abc'
  - **Ops**: + (concatenation)

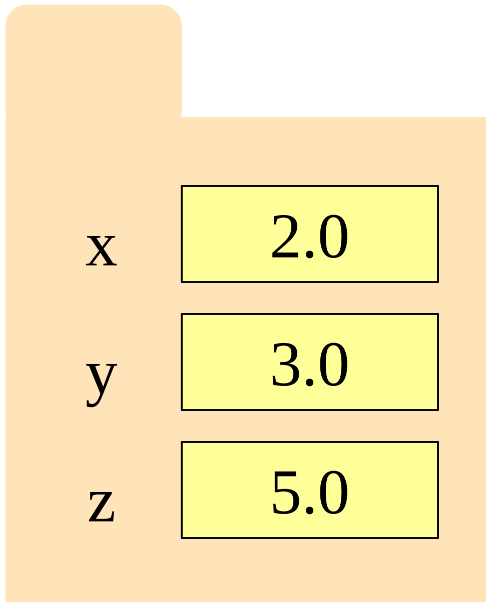
Built-in Types are not "Enough"

- Want a point in 3D space
  - We need three variables
  - *x*, *y*, *z* coordinates
- What if have a lot of points?
  - Vars x0, y0, z0 for first point
  - Vars x1, y1, z1 for next point
  - …
  - This can get really messy
- How about a single variable that represents a point?



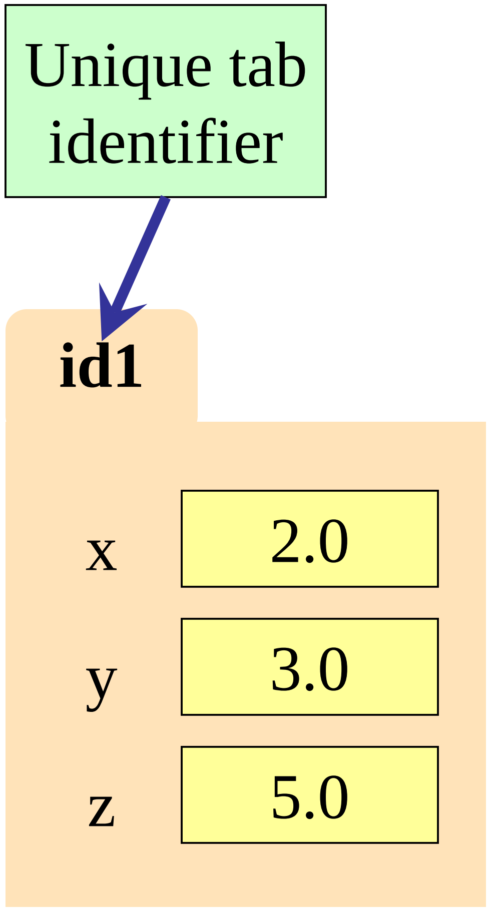
Built-in Types are not "Enough"

- Want a point in 3D space
  - We need three variables
  - *x*, *y*, *z* coordinates
- What if have a lot of points?
  - Vars x0, y0, z0 for first point
  - Vars x1, y1, z1 for next point
  - …
  - This can get really messy
- How about a single variable that represents a point?
- Can we stick them together in a "folder"?
- Motivation for **objects**



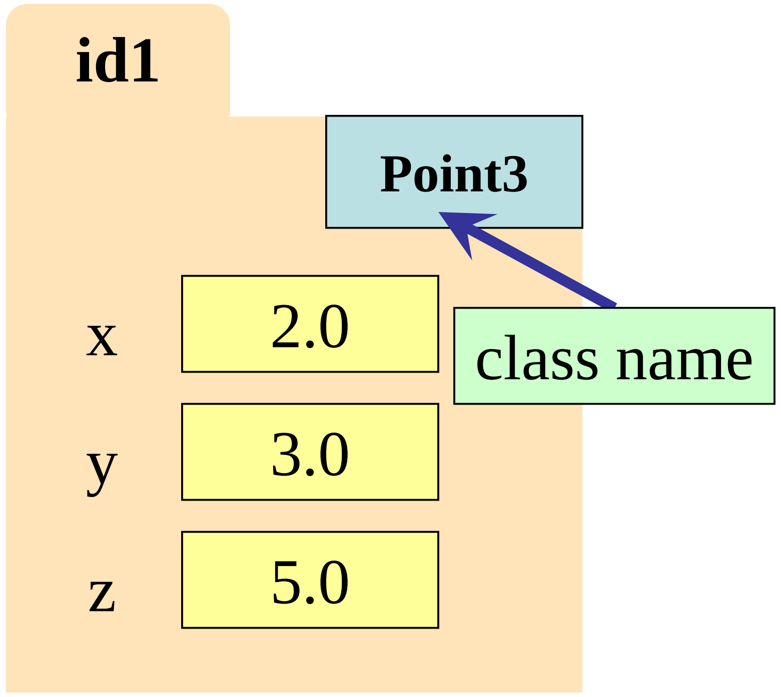
Objects: Organizing Data in Folders

- An object is like a **manila folder**
- It contains other variables
  - Variables are called **attributes**
  - These values can change
- It has an **ID** that identifies it
  - Unique number assigned by Python (just like a NetID for a Cornellian)
  - Cannot ever change
  - Has no meaning; only identifies



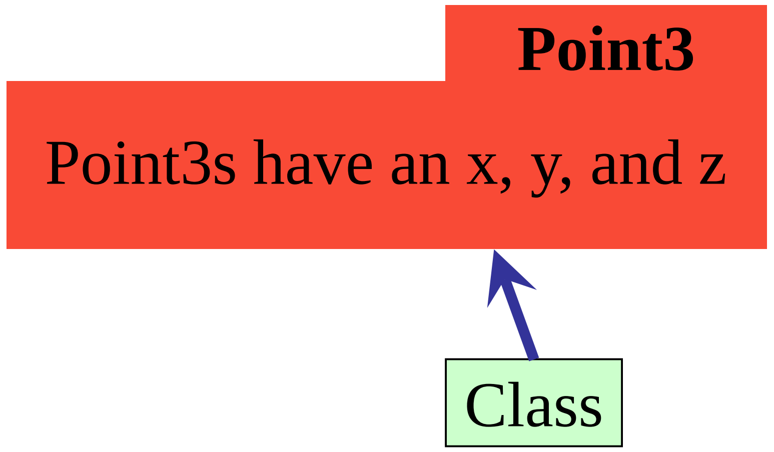
Classes: Types for Objects

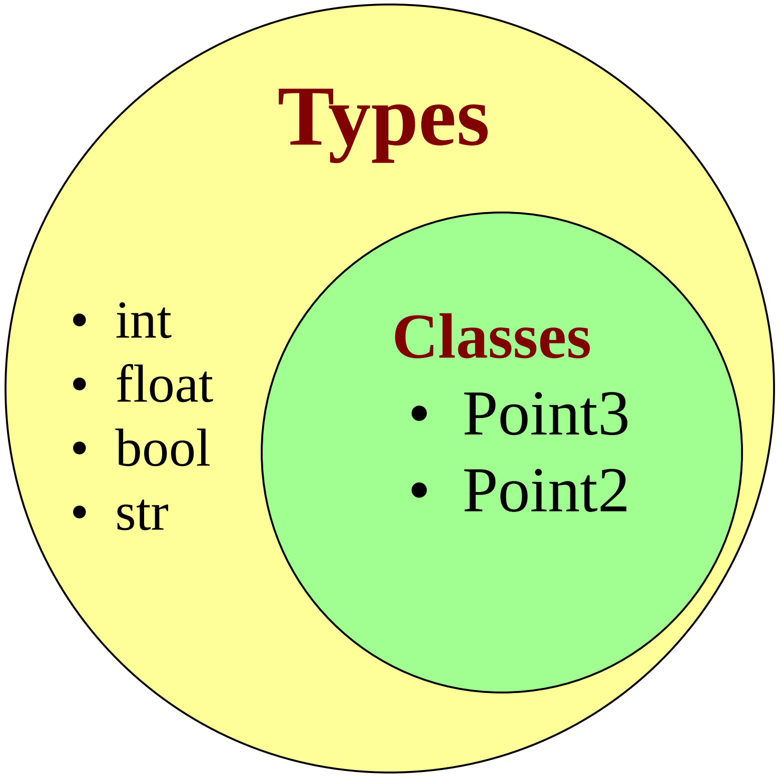
- Values must have a type
  - An object is a **value**
  - Object type is a **class**
- **Modules** provide classes
  - Will show how later
- **Example**: geom
  - Classes: Point2, Point3



Classes: Types for Objects

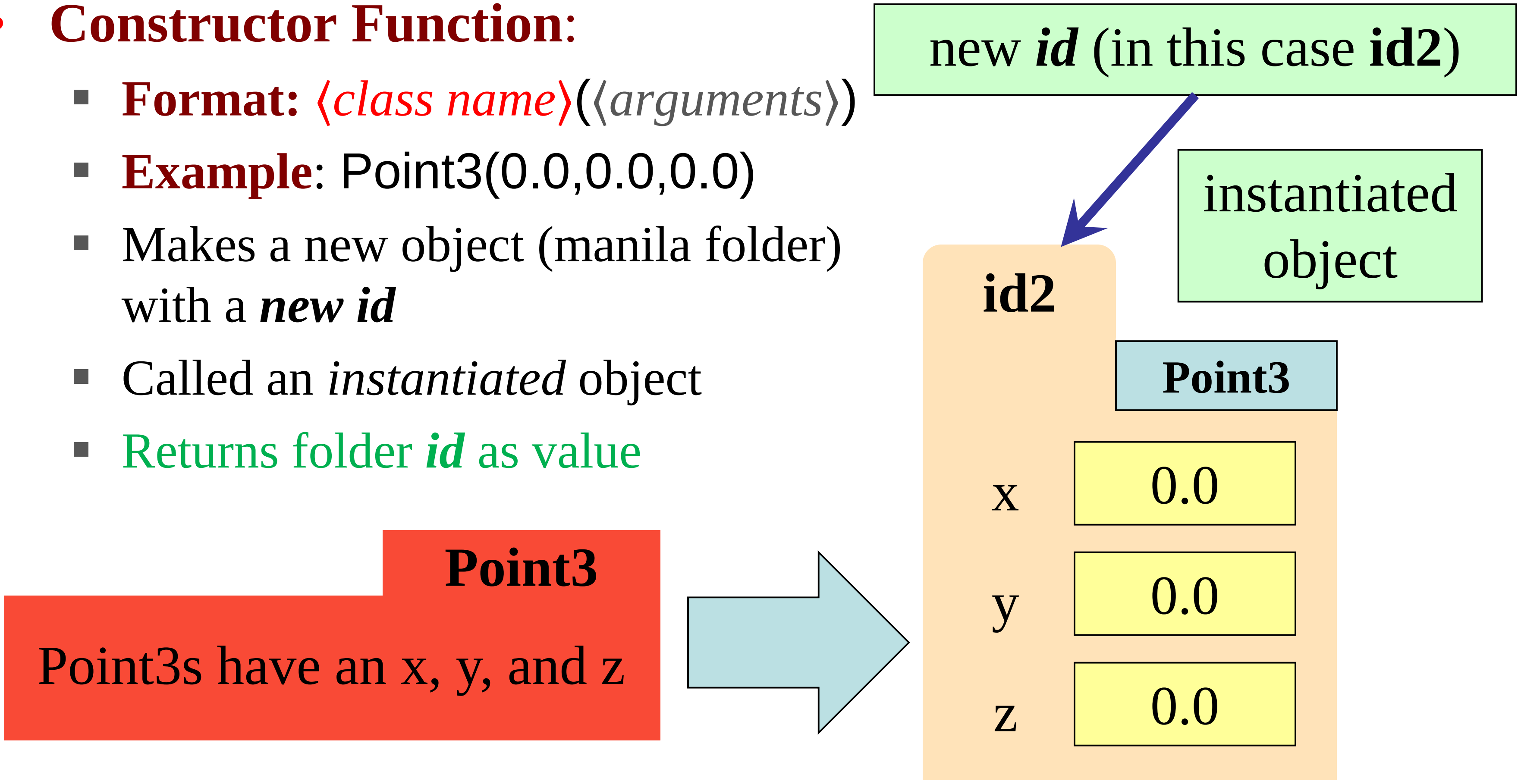
- Classes are how we add new types to Python
- Sort of like a template





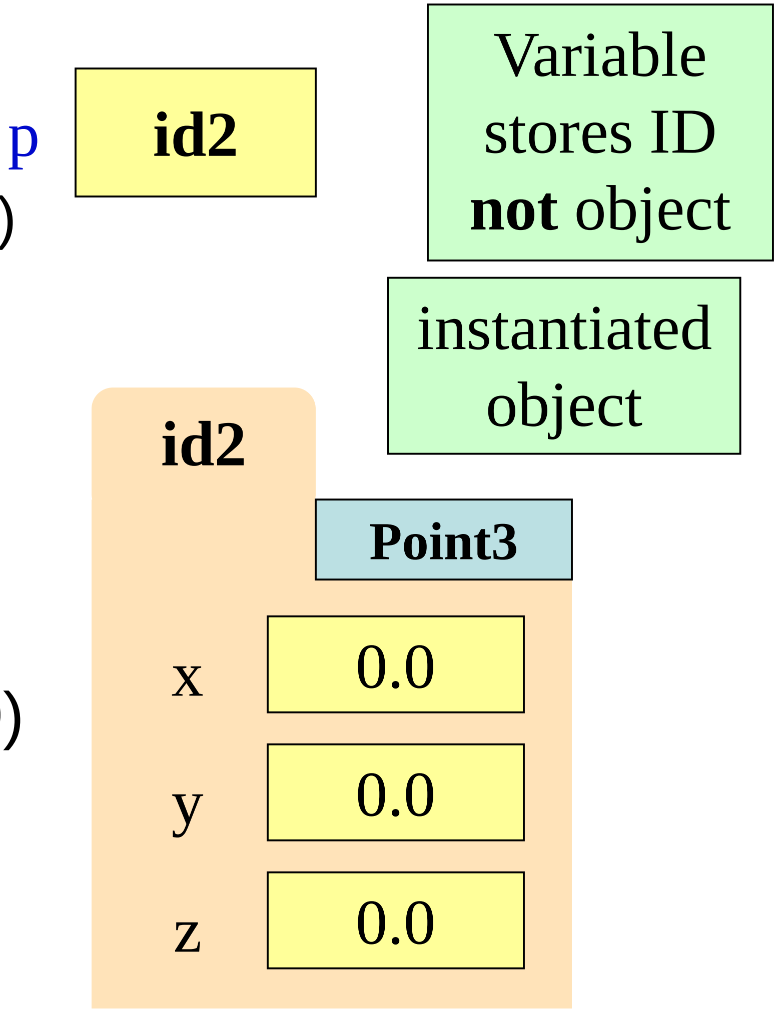
Constructor: Function to make Objects

- How do we create objects?
- **Constructor Function**:
  - **Format:** ⟨*class name*⟩(⟨*arguments*⟩)
  - **Example**: Point3(0.0,0.0,0.0)
  - Makes a new object (manila folder) with a ***new id***
  - Called an *instantiated* object
  - Returns folder ***id*** as value


new ***id*** (in this case **id2**)
instantiated object
id2
Point3
x
0.0
y
0.0
z
0.0
Point3
Point3s have an x, y, and z

Constructor: Function to make Objects

- How do we create objects?
- **Constructor Function:**
  - **Format:** ⟨*class name*⟩(⟨*arguments*⟩)
  - **Example**: Point3(0.0,0.0,0.0)
  - Makes a new object (manila folder) with a ***new id***
  - Called an *instantiated* object
  - Returns folder ***id*** as value
- **Example**: p = Point3(0.0, 0.0, 0.0)
  - Creates a Point object
  - Stores object's ***id*** in p



Like a Greek god!

Constructors and Modules

```
>>> import geom
```

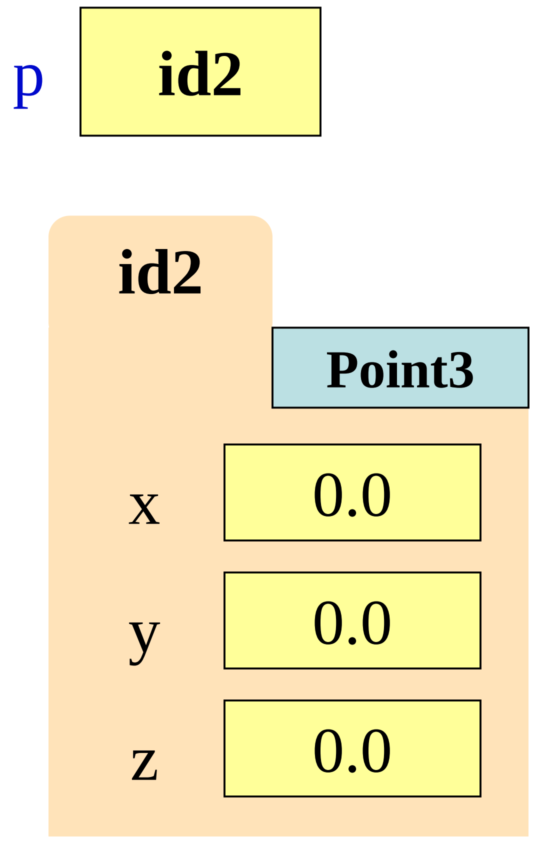
Need to import module that has Point class.

```
>>> p = geom.Point3(0.0,0.0,0.0)
```

Constructor is function. Prefix w/ module name.

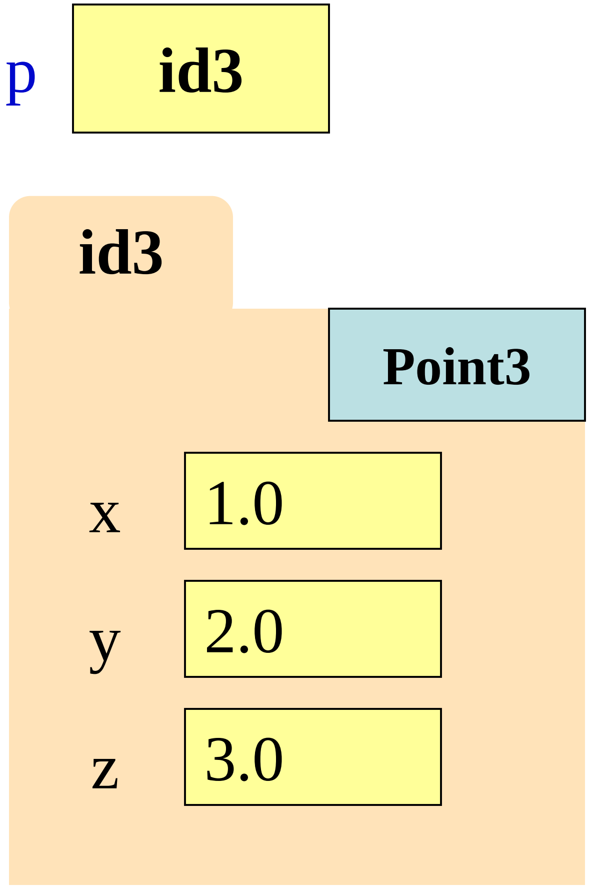
```
>>> id(p)
```

Shows the ***id*** of p.



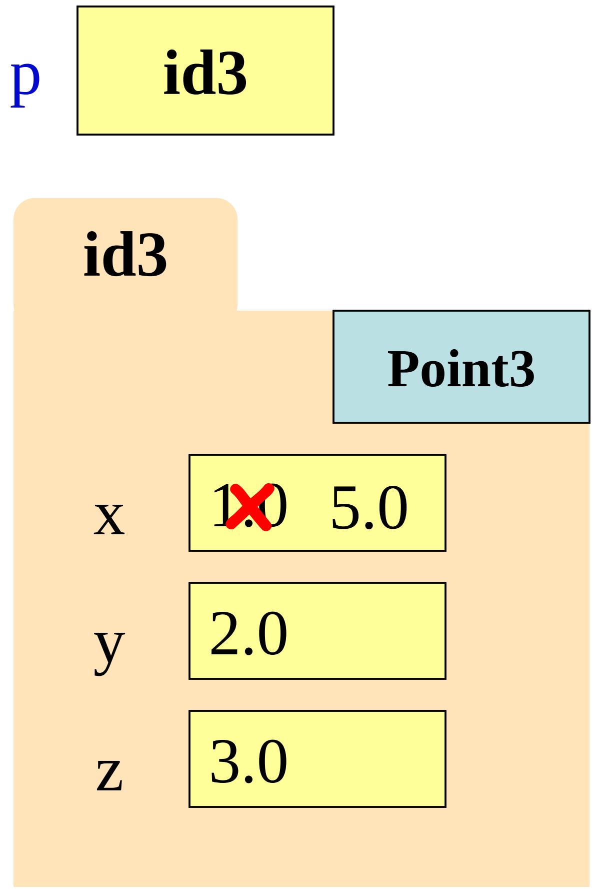
Accessing Attributes

- Attributes are variables that live inside of objects
  - Can **use** in expressions
  - Can **assign** values to them
- **Format**: ⟨*variable*⟩.⟨*attribute*⟩
  - **Example**: p.x
  - Look like module variables
- To evaluate p.x, Python:
  1. finds folder with ***id*** stored in p
  2. returns the value of x in that folder



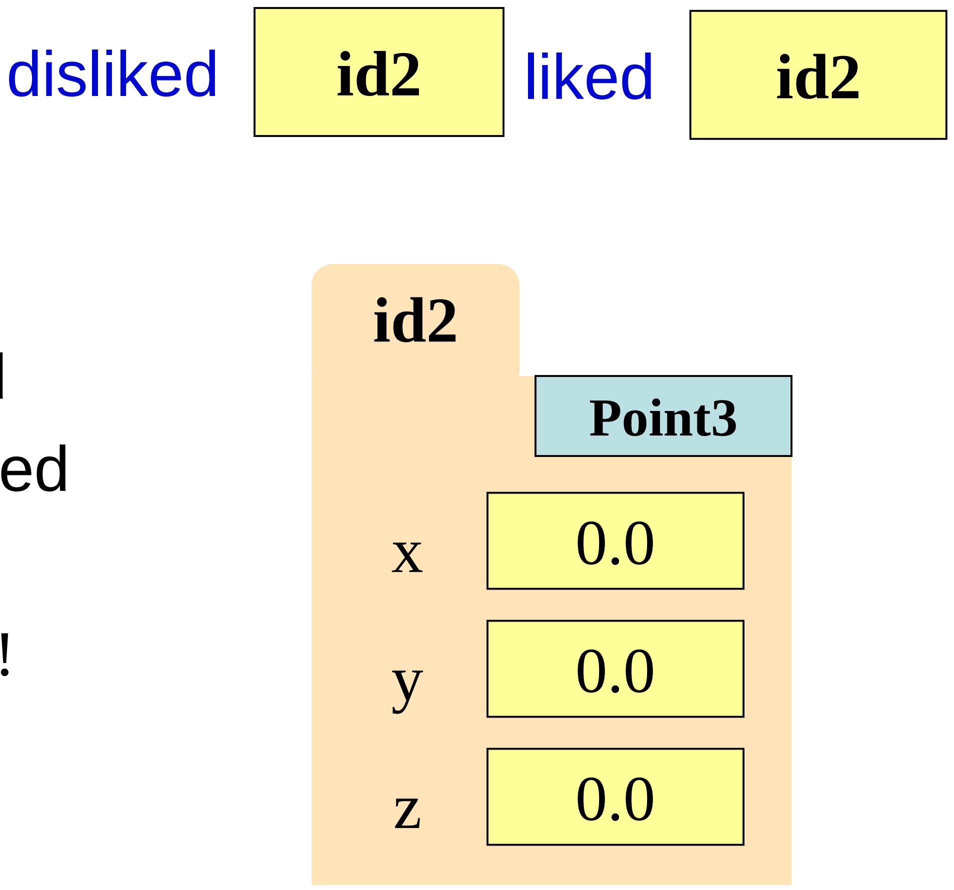
Accessing Attributes

- **Example:**
  - p = geom.Point3(1.0, 2.0, 3.0)
  - p.x = p.y + p.z



Object Variables

- Variable stores object ***id***
  - **Reference** to the object
  - Reason for folder analogy
- Assignment uses object ***id***
  - **Example**: `liked = disliked`
  - Takes contents from `disliked`
  - Puts contents in `liked`
  - Does not make new folder!
- This is the cause of many mistakes in this course



Exercise: Attribute Assignment

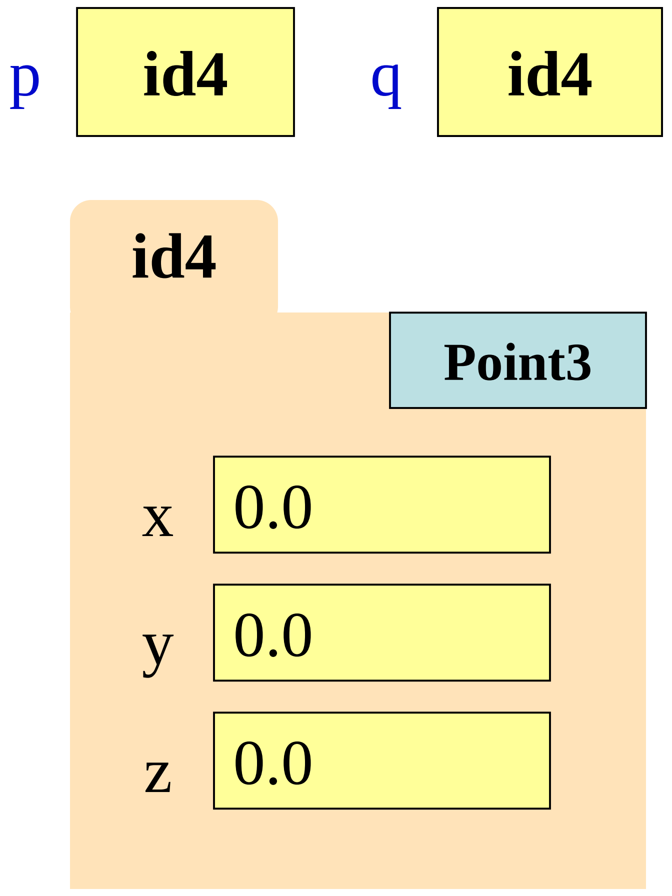
```
>>> p = geom.Point3(0,0,0)
>>> q = p
```

- Execute the assignments:

```
>>> p.x = 5.6
>>> q.x = 7.4
```

- What is value of p.x?

A: 5.6
B: 7.4 **CORRECT**
C: **id4**
D: I don't know



Exercise: Attribute Assignment

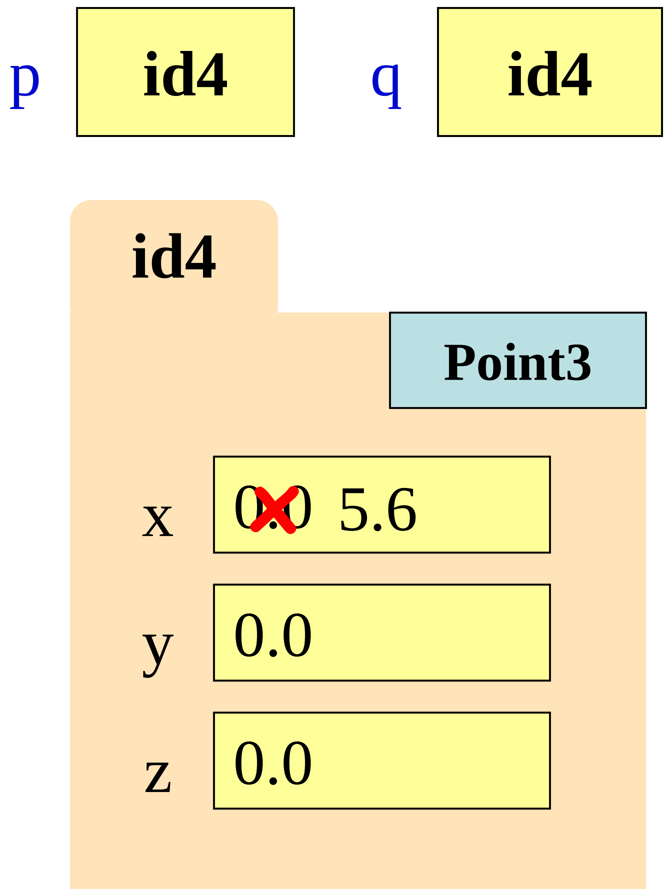
```
>>> p = geom.Point3(0,0,0)
>>> q = p
```

- Execute the assignments:

```
  >>> p.x = 5.6
  >>> q.x = 7.4
```

- What is value of p.x?

A: 5.6
B: 7.4 **CORRECT**
C: **id4**
D: I don’t know



Exercise: Attribute Assignment

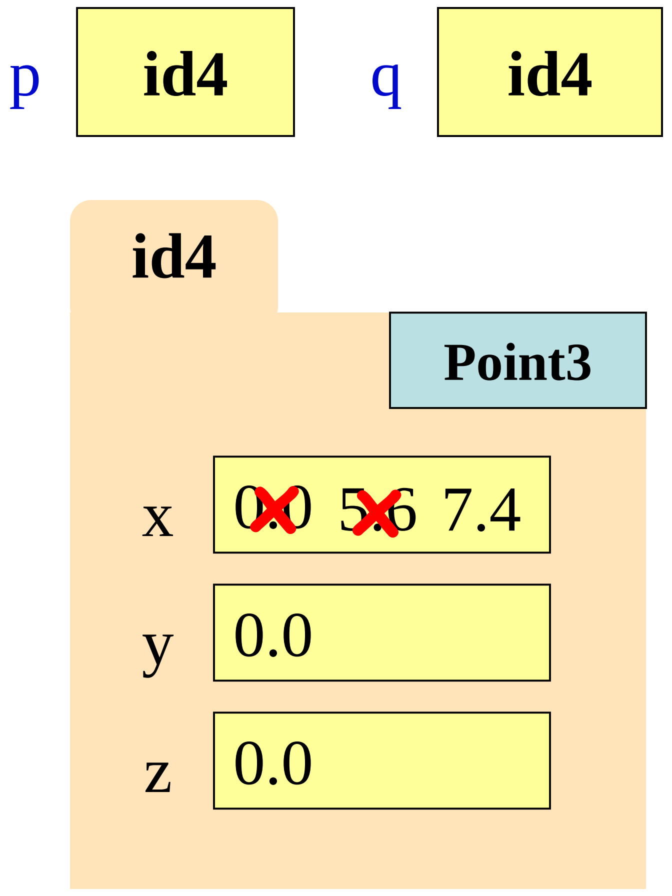
```
>>> p = geom.Point3(0,0,0)
>>> q = p
```

- Execute the assignments:

```
>>> p.x = 5.6
>>> q.x = 7.4
```

- What is value of p.x?

A: 5.6
B: 7.4 **CORRECT**
C: **id4**
D: I don't know



Assignment and Attribute Oddness

```
>>> p = 5.0
>>> q = p
>>> p = 4.0
>>> q
5.0
```

```
>>> from geom import *
>>> p = Point3(1.0,2.0,3.0)
>>> q = p
>>> p.x = 4.0
>>> q.x
4.0
```

!!

The rules of variables have not changed! However, combining variable assignment with object references can be confusing.

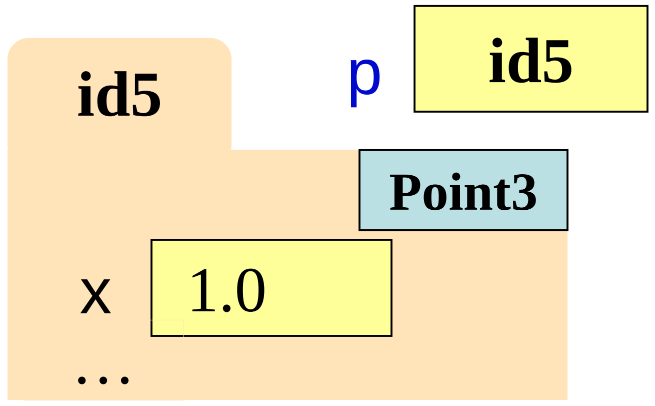
Call Frames and Objects

- Objects can be altered in a function call
  - Object variables hold ***id***s!
  - Folder can be accessed from global variable or parameter
- **Example**:

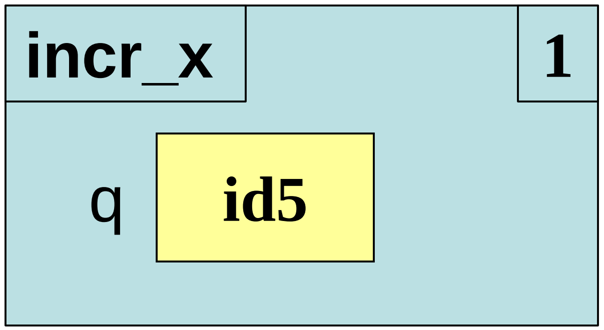
```
  def incr_x(q):
1     q.x = q.x + 1.0
```

```
>>> p = geom.Point3(1.0, 2.0, 3.0)
>>> incr_x(p)
```

Global **STUFF**



Call Frame

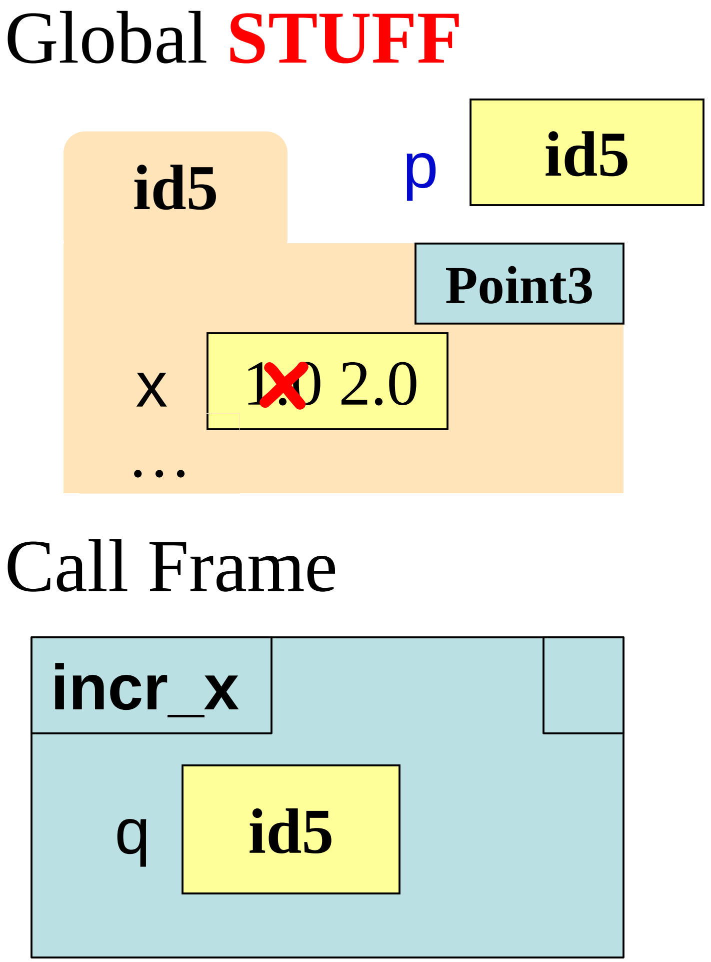


Call Frames and Objects

- Objects can be altered in a function call
  - Object variables hold ***id***s!
  - Folder can be accessed from global variable or parameter
- **Example**:

```
def incr_x(q):
    q.x = q.x + 1.0
```

```
>>> p = geom.Point3(1.0, 2.0, 3.0)
>>> incr_x(p)
```



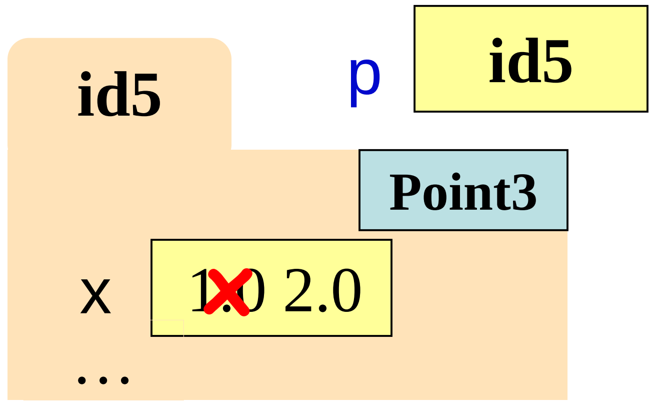
Call Frames and Objects

- Objects can be altered in a function call
  - Object variables hold ***id***s!
  - Folder can be accessed from global variable or parameter
- **Example**:

```
def incr_x(q):
1       q.x = q.x + 1.0
```

```
>>> p = geom.Point3(1.0, 2.0, 3.0)
>>> incr_x(p)
```

Global **STUFF**



Call Frame

Exercise: Attribute Assignment

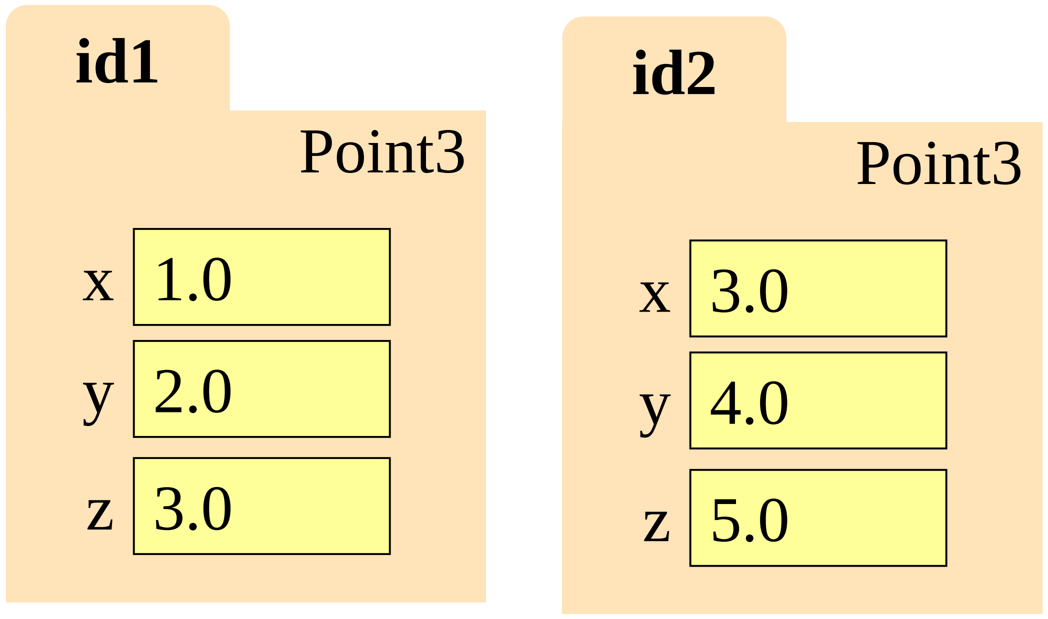
```
import geom
p = geom.Point3(1.0,2.0,3.0)
q = geom.Point3(3.0,4.0,5.0)
```

Draw everything that gets created.
How many folders get drawn?

Exercise: Attribute Assignment

```
import geom
p = geom.Point3(1.0,2.0,3.0)
q = geom.Point3(3.0,4.0,5.0)
```

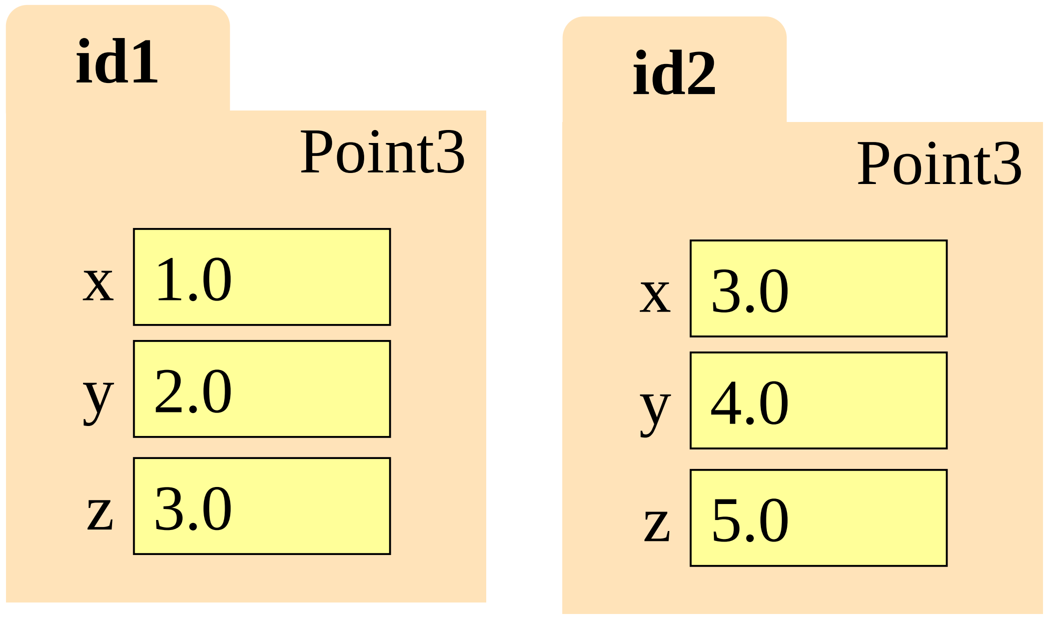
Draw everything that gets created.
How many folders get drawn?



Exercise: Attribute Assignment

```
import geom
p = geom.Point3(1.0,2.0,3.0)
q = geom.Point3(3.0,4.0,5.0)
```

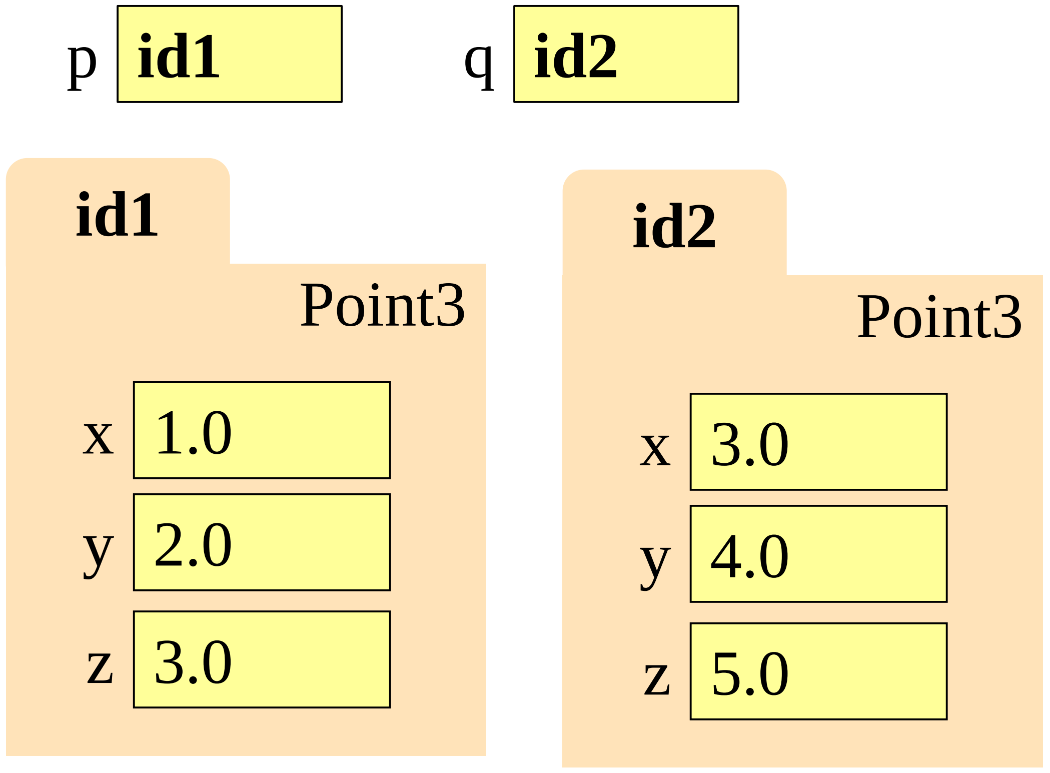
Draw everything that gets created.
How many folders get drawn?
What else gets drawn?



Exercise: Attribute Assignment

```
import geom
p = geom.Point3(1.0,2.0,3.0)
q = geom.Point3(3.0,4.0,5.0)
```

Draw everything that gets created.
How many folders get drawn?
What else gets drawn?



Exercise: Attribute Assignment

```
import geom
p = geom.Point3(1.0,2.0,3.0)
q = geom.Point3(3.0,4.0,5.0)
swap_x(p, q)
```

```
def swap_x(p, q):
1     t = p.x
2     p.x = q.x
3     q.x = t
```

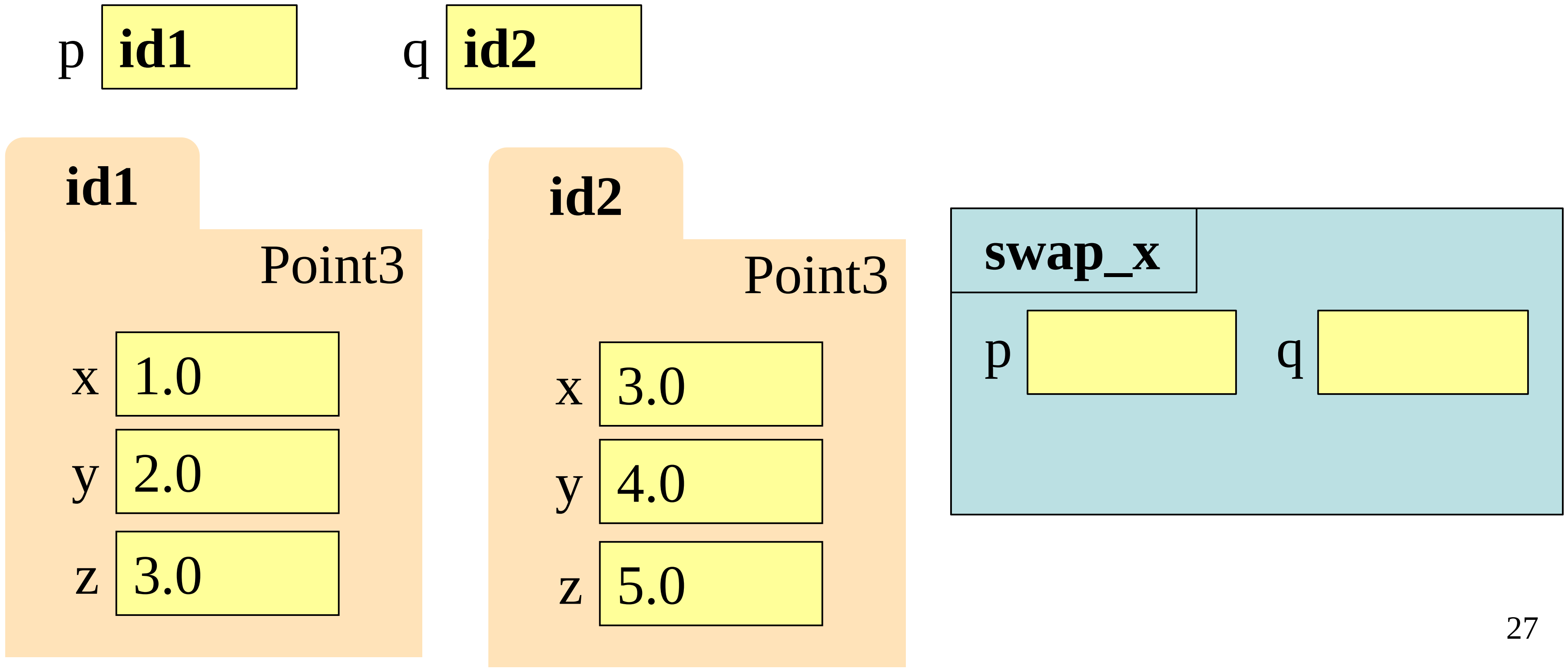
Execute swap_x on what we just drew.
There should be a call frame.
What is in p.x at the end?

A: 1.0
B: 2.0
C: 3.0
D: I don’t know

Exercise: Attribute Assignment

```
import geom
p = geom.Point3(1.0,2.0,3.0)
q = geom.Point3(3.0,4.0,5.0)
swap_x(p, q)
```

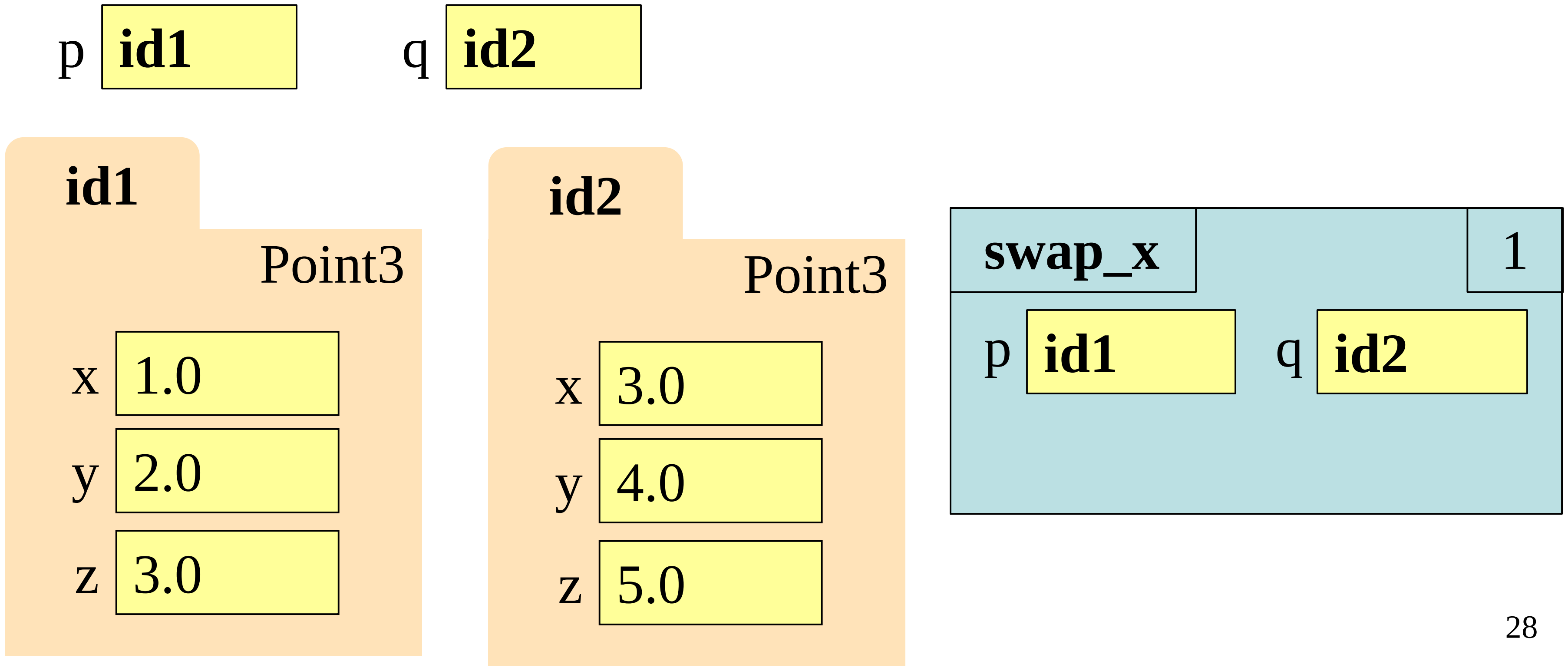
```
def swap_x(p, q):
1     t = p.x
2     p.x = q.x
3     q.x = t
```



Exercise: Attribute Assignment

```
import geom
p = geom.Point3(1.0,2.0,3.0)
q = geom.Point3(3.0,4.0,5.0)
swap_x(p, q)
```

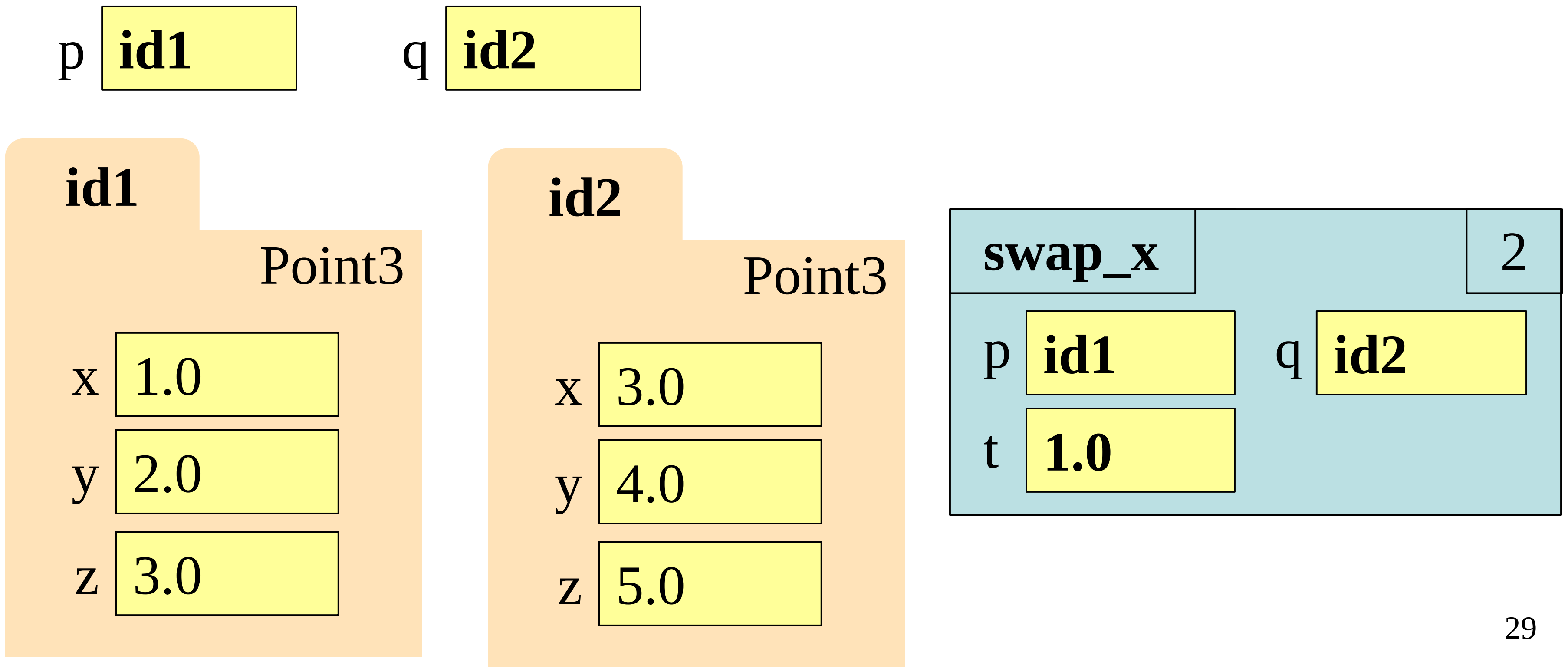
```
def swap_x(p, q):
1    t = p.x
2    p.x = q.x
3    q.x = t
```



Exercise: Attribute Assignment

```
import geom
p = geom.Point3(1.0,2.0,3.0)
q = geom.Point3(3.0,4.0,5.0)
swap_x(p, q)
```

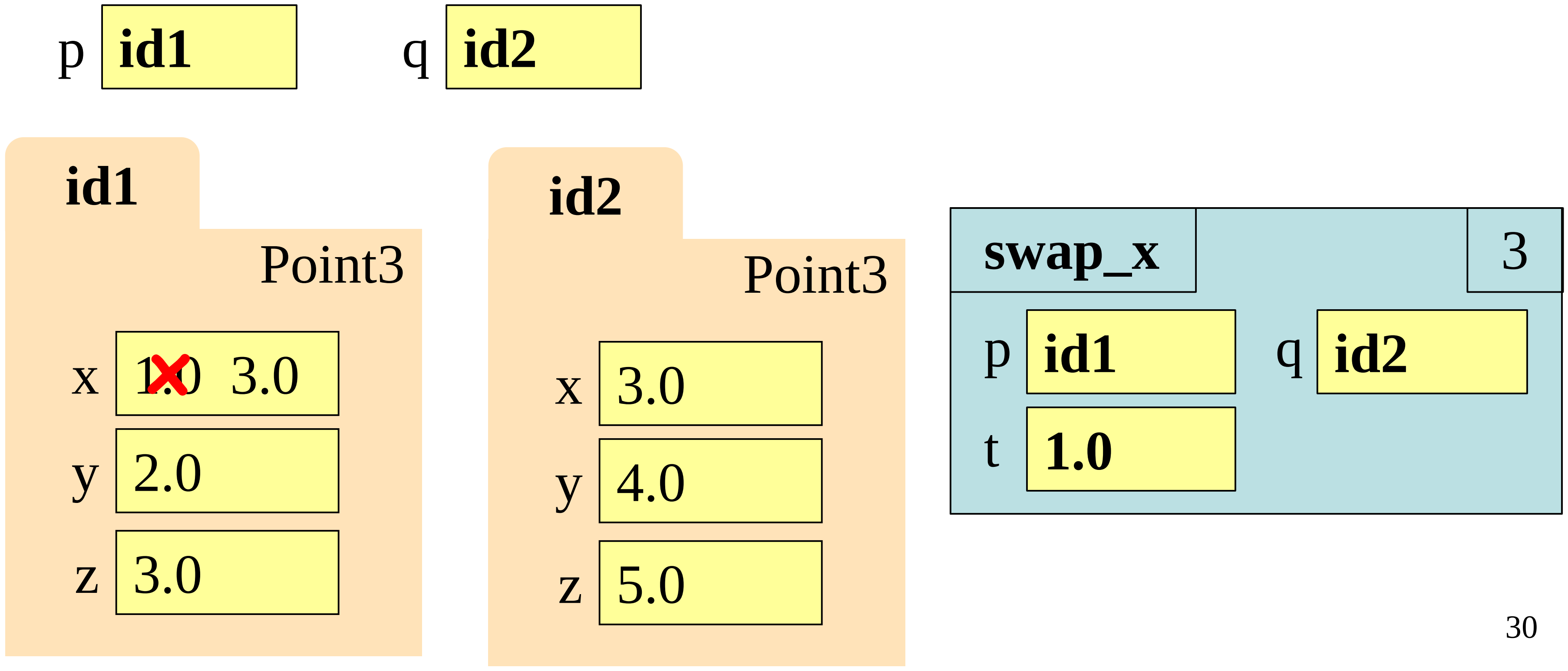
```
def swap_x(p, q):
1    t = p.x
2    p.x = q.x
3    q.x = t
```



Exercise: Attribute Assignment

```
import geom
p = geom.Point3(1.0,2.0,3.0)
q = geom.Point3(3.0,4.0,5.0)
swap_x(p, q)
```

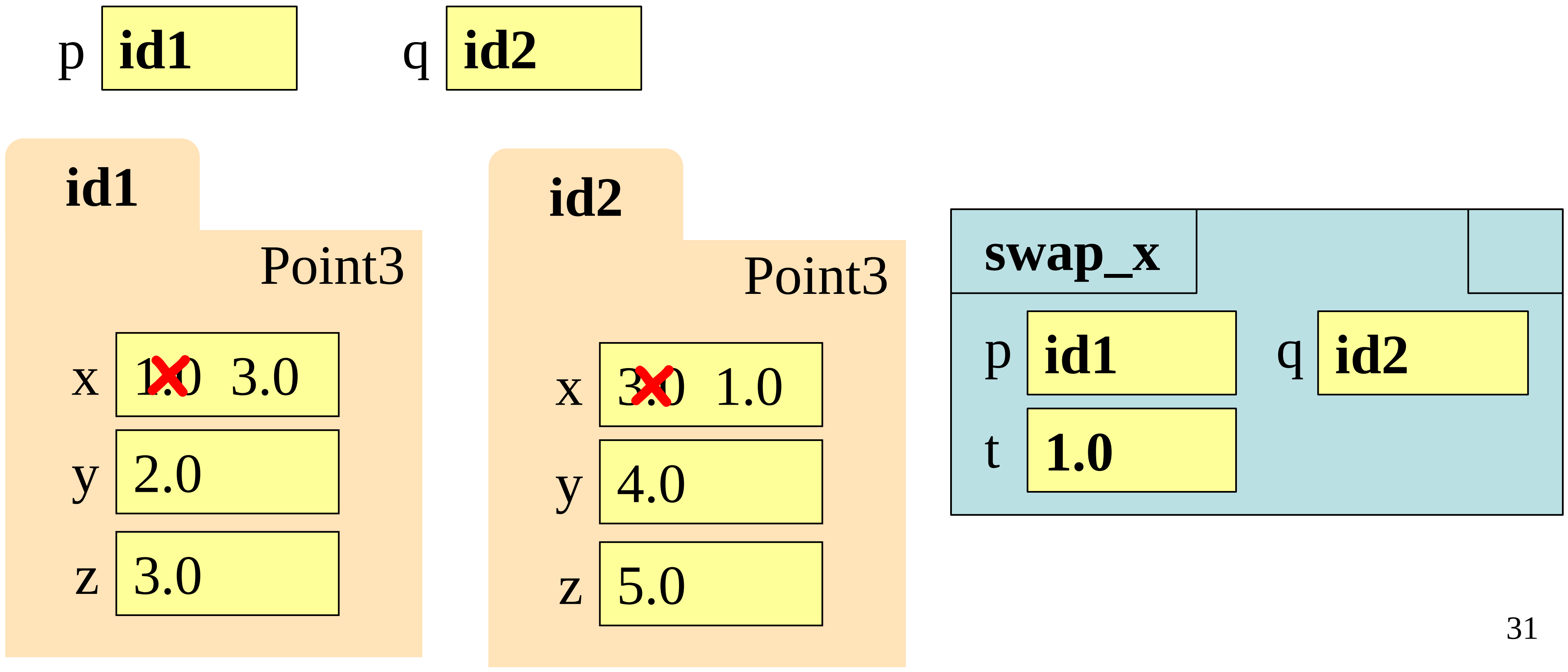
```
def swap_x(p, q):
1    t = p.x
2    p.x = q.x
3    q.x = t
```



Exercise: Attribute Assignment

```
import geom
p = geom.Point3(1.0,2.0,3.0)
q = geom.Point3(3.0,4.0,5.0)
swap_x(p, q)
```

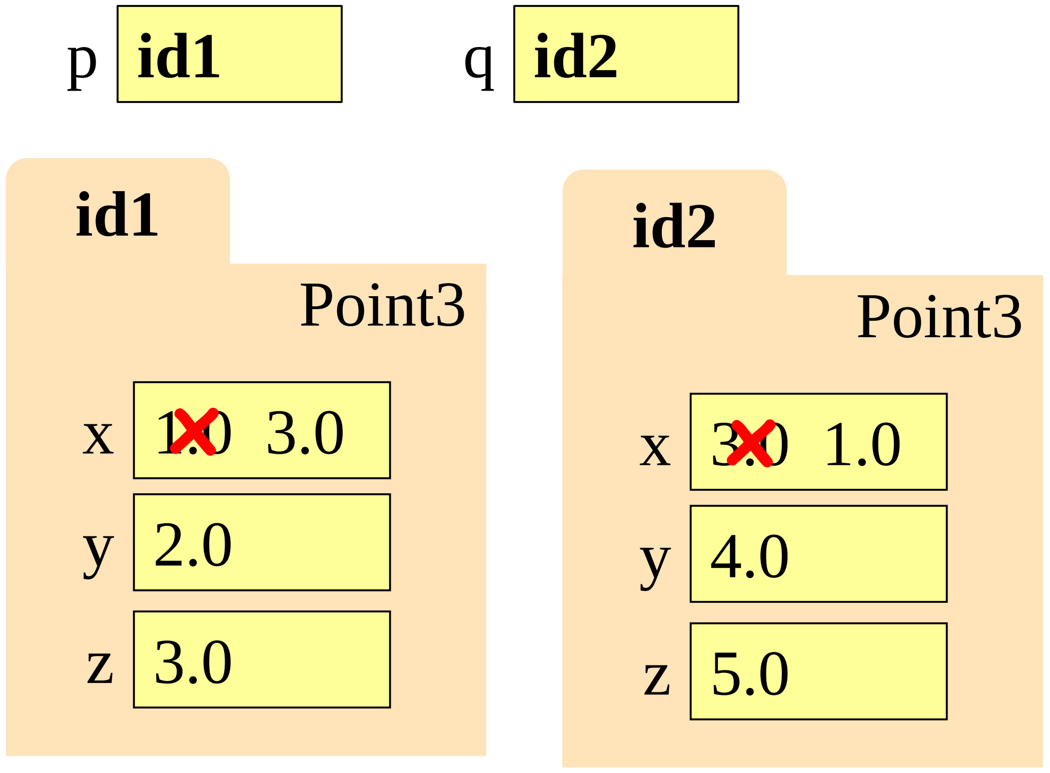
```
def swap_x(p, q):
1    t = p.x
2    p.x = q.x
3    q.x = t
```



Exercise: Attribute Assignment

```
import geom
p = geom.Point3(1.0,2.0,3.0)
q = geom.Point3(3.0,4.0,5.0)
swap_x(p, q)
```

```
def swap_x(p, q):
1    t = p.x
2    p.x = q.x
3    q.x = t
```



ERASE WHOLE FRAME

Exercise: Attribute Assignment

```
import geom
p = geom.Point3(1.0,2.0,3.0)
q = geom.Point3(3.0,4.0,5.0)
swap_x(p, q)
```

```
def swap_x(p, q):
1    t = p.x
2    p.x = q.x
3    q.x = t
```

Execute swap_x on what we just drew.
There should be a call frame.
What is in p.x at the end?

A: 1.0
B: 2.0
C: 3.0 **CORRECT**
D: I don’t know

Exercise: Attribute Assignment

```
import geom
p = geom.Point3(1.0,2.0,3.0)
q = geom.Point3(3.0,4.0,5.0)
swap(p, q)
```

```
def swap(p, q):
    t = p
    p = q
    q = t
```

Before calling swap(p, q):

p **id1**   q **id2**

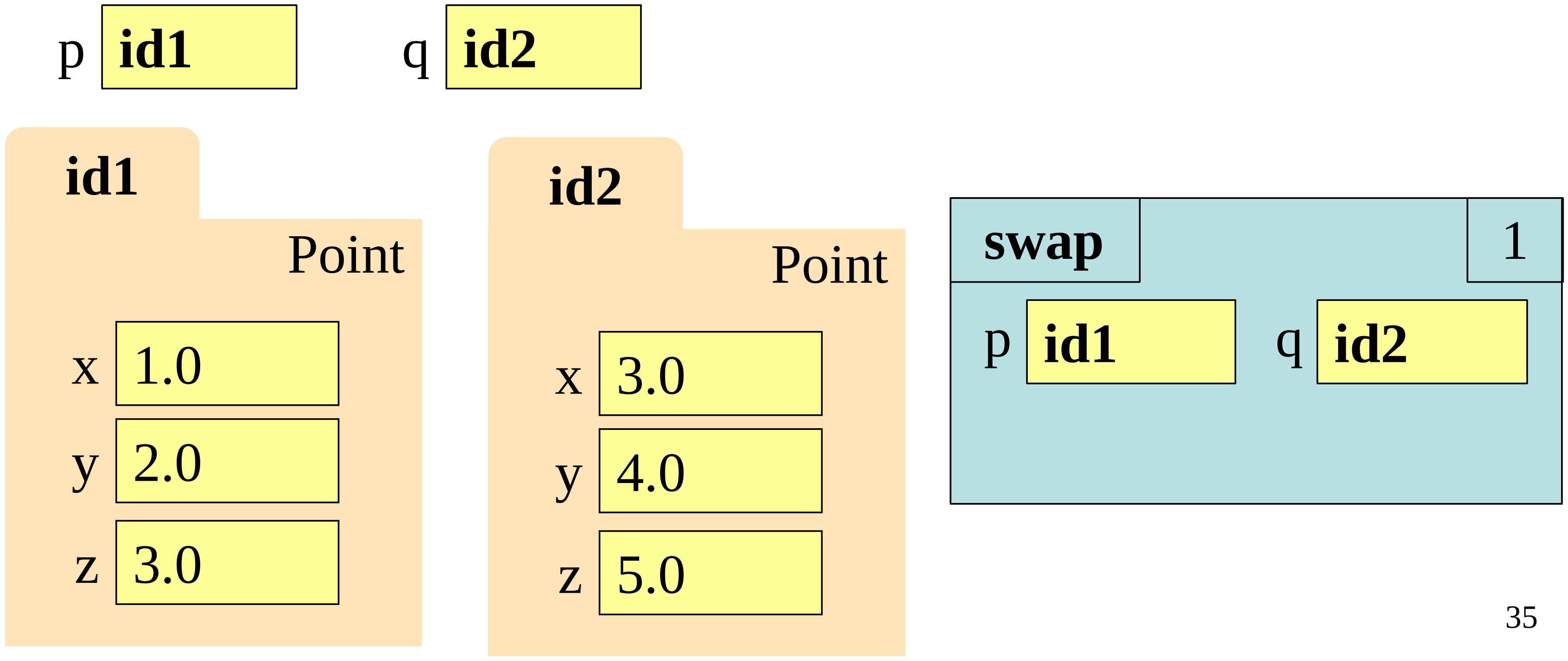
What is in global p after calling swap?

A: **id1**
B: **id2**
C: I don’t know

Exercise: Attribute Assignment

```
import geom
p = geom.Point3(1.0,2.0,3.0)
q = geom.Point3(3.0,4.0,5.0)
swap(p, q)
```

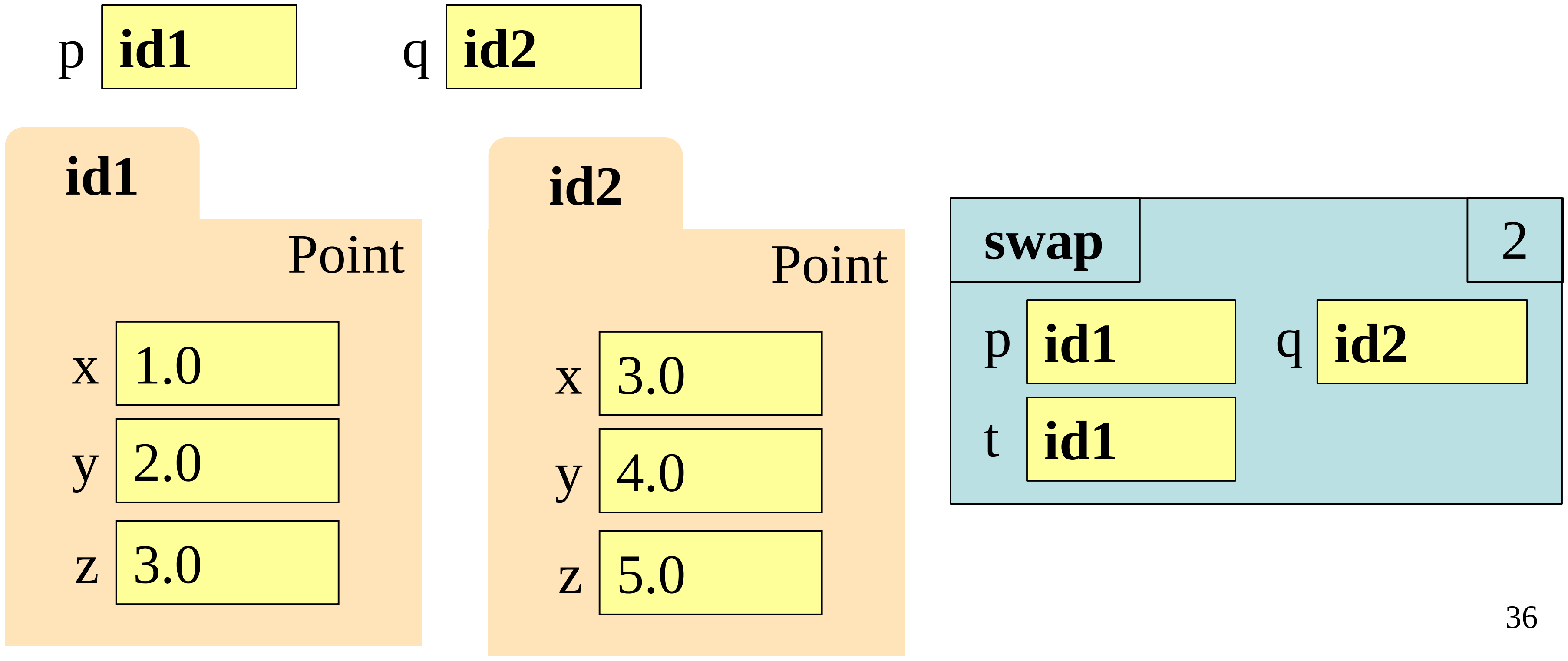
```
def swap(p, q):
  1   t = p
  2   p = q
  3   q = t
```



Exercise: Attribute Assignment

```
import geom
p = geom.Point3(1.0,2.0,3.0)
q = geom.Point3(3.0,4.0,5.0)
swap(p, q)
```

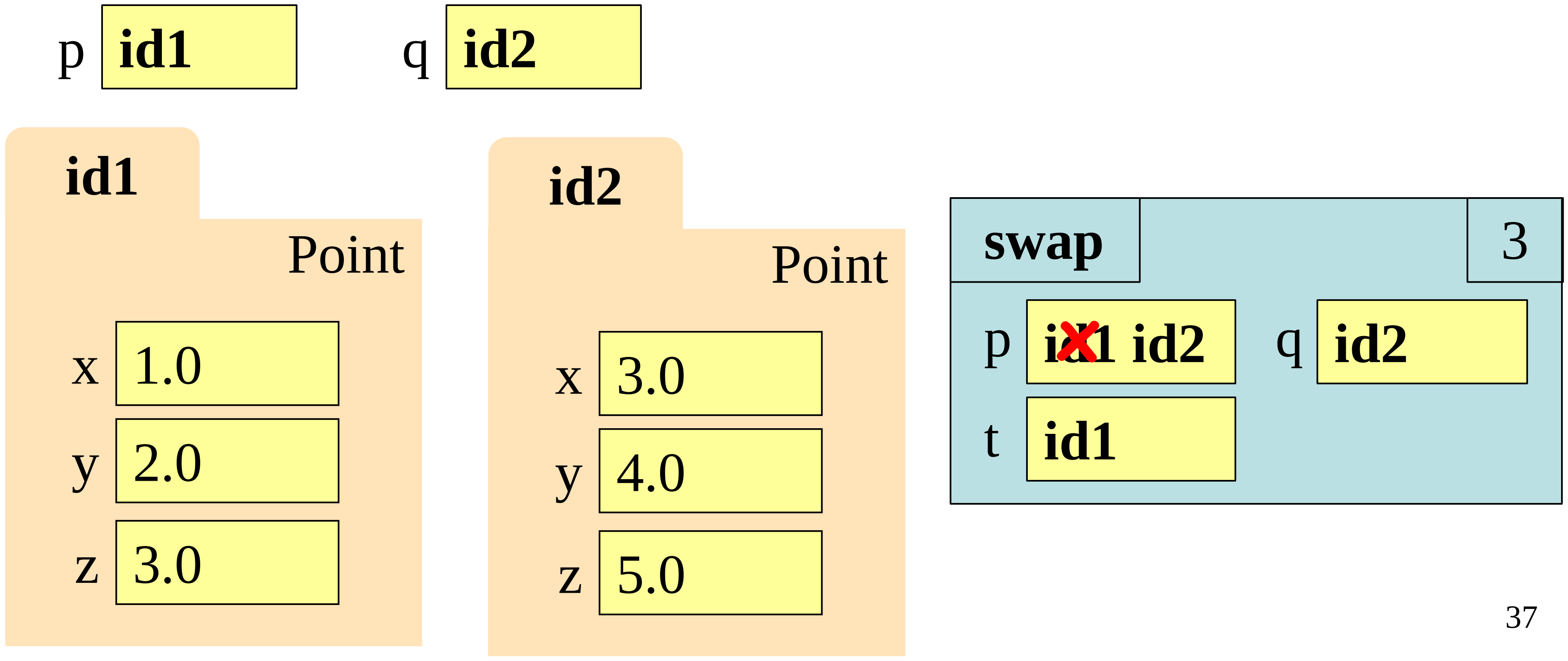
```
def swap(p, q):
1    t = p
2    p = q
3    q = t
```



Exercise: Attribute Assignment

```
import geom
p = geom.Point3(1.0,2.0,3.0)
q = geom.Point3(3.0,4.0,5.0)
swap(p, q)
```

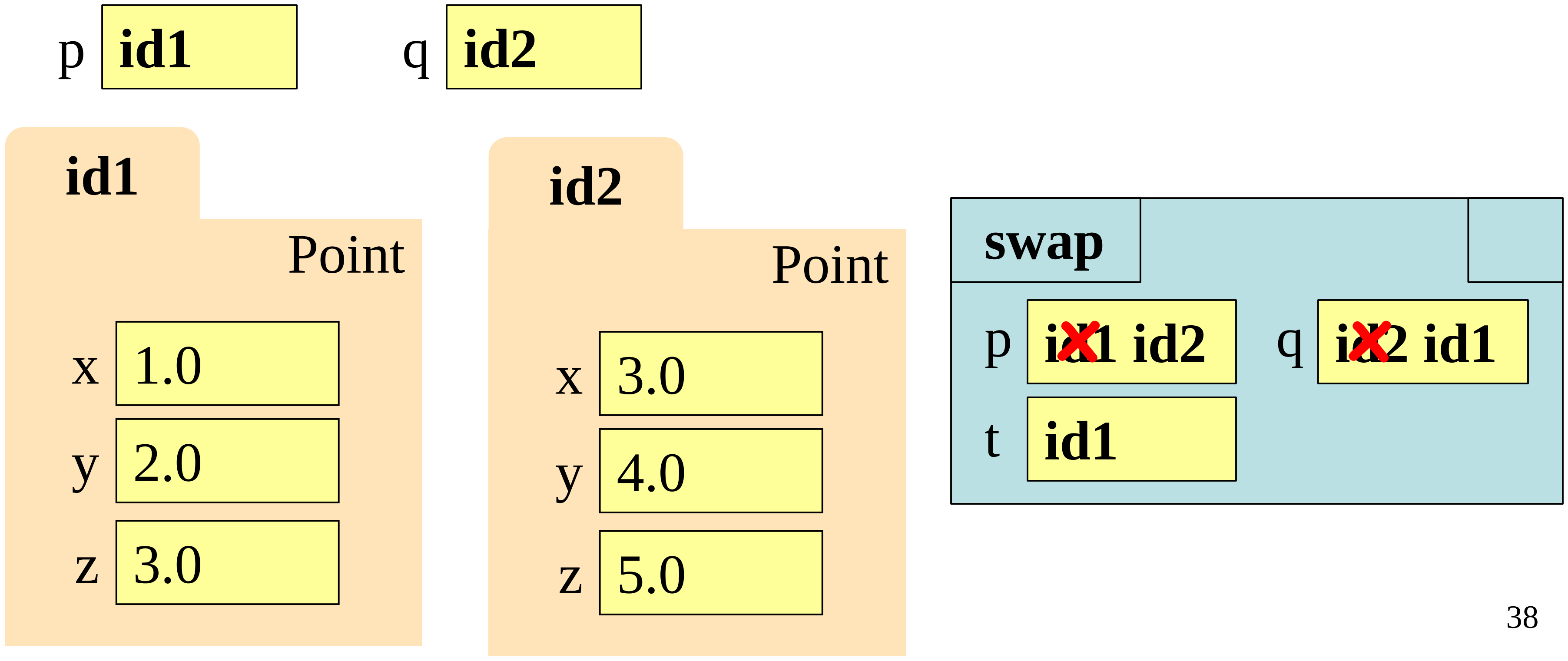
```
def swap(p, q):
1    t = p
2    p = q
3    q = t
```



Exercise: Attribute Assignment

```
import geom
p = geom.Point3(1.0,2.0,3.0)
q = geom.Point3(3.0,4.0,5.0)
swap(p, q)
```

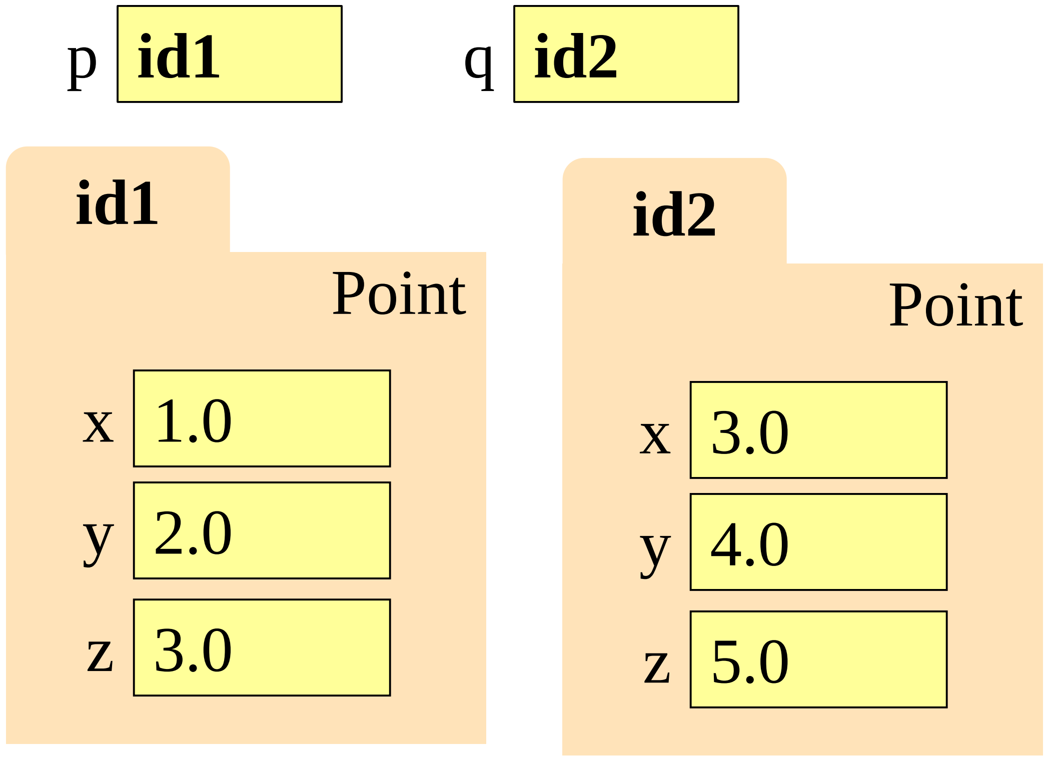
```
def swap(p, q):
    t = p
    p = q
    q = t
```



Exercise: Attribute Assignment

```
import geom
p = geom.Point3(1.0,2.0,3.0)
q = geom.Point3(3.0,4.0,5.0)
swap(p, q)
```

```
def swap(p, q):
    t = p
    p = q
    q = t
```



ERASE WHOLE FRAME

Exercise: Attribute Assignment

```
import geom
p = geom.Point3(1.0,2.0,3.0)
q = geom.Point3(3.0,4.0,5.0)
swap(p, q)
```

```
def swap(p, q):
    t = p
    p = q
    q = t
```

p **id1**   q **id2**

What is in global p after calling swap?

A: **id1** CORRECT
B: **id2**
C: I don’t know