03-1 Classes and Objects

CSI 500

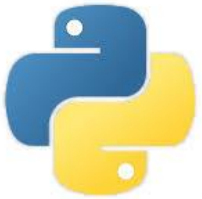
Spring 2018

Course material derived from:
Downey, Allen B. 2012. "Think Python, 2nd Edition". O'Reilly Media Inc., Sebastopol CA.

"How to Think Like a Computer Scientist" by Peter Wentworth, Jeffrey Elkner, Allen B. Downey, and Chris Meyers. Oct 2012
http://openbookproject.net/thinkcs/python/english3e/index.html

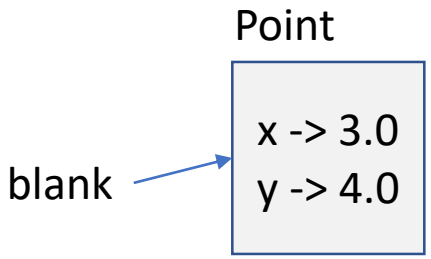
Programmer defined types

- We've used lots of Python built-in types - now let's create our own type
  - Point - represents a point in 2-D space
- How to do it?
  - manage coordinates in x and y variables
  - store coordinates in a list or tuple
  - create a new type, called a **class**
- Create instance of the class using Point()
  - makes an "object"



```
class Point:
        """ Represents a point in 2-D space """

Point
<class '__main__.Point'>

blank = Point()       # create instance
blank
<__main__.Point object at 0x00227B4DC1C18>
```

Attributes



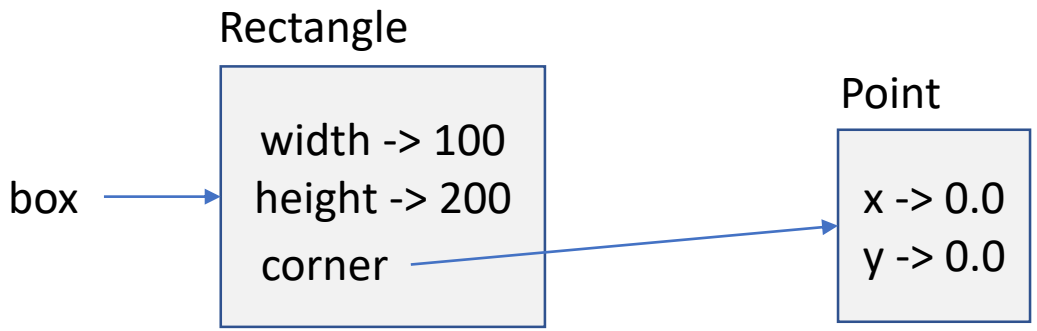
- Assign values to an instance using "dot notation"
  - called "**attributes**"
- can access attributes just like regular variables
  - use in expressions
  - use in functions

```
# continue our example
 blank.x = 3.0
 blank.y = 4.0

# we can access instance variables as well
 print( blank.y )
4.0

 spam = blank.x
 spam
3.0

# use instance vars w dot notation like regular
 print( '(%g %g)' % (blank.x, blank.y))
(3 4)

 dist = math.sqrt( blank.x**2 + blank.y**2)
 dist
5.0

 def print_point( p ):
                print('(%g %g)' % (p.x, p.y)
 print_point( blank )
(3 4)
```

Rectangles

- When designing classes, you have to consider how the attributes should work
- Consider a Rectangle class
  - need length, width
  - need position in 2D space
- How to implement?
  - Could specify one corner (or center), width and height
  - Could specify two opposing corners

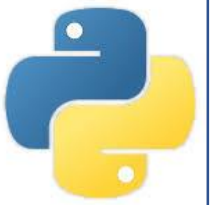
```
# here's a basic Rectangle class
class Rectangle:
    """ implements a 2D rectangle
    attributes: width, length, corner
    """


box = Rectangle()

box.width = 100
box.height = 200

box.corner = Point()
box.corner.x = 0.0
box.corner.y = 0.0
```



Instances as return values

- Functions can return instances of classes
  - example: find the center of a box (Rectangle), and return a Point

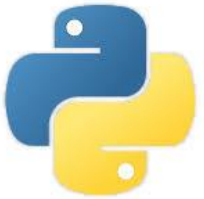
```
# continue our box example…
 def find_center( rect ):
     p = Point()
     p.x = (rect.corner.x + rect.width)/2
     p.y = (rect.corner.y + rect.height)/2
     return p

 center = find_center( box )
 print_point( center )
( 50 100)
```



Objects are mutable

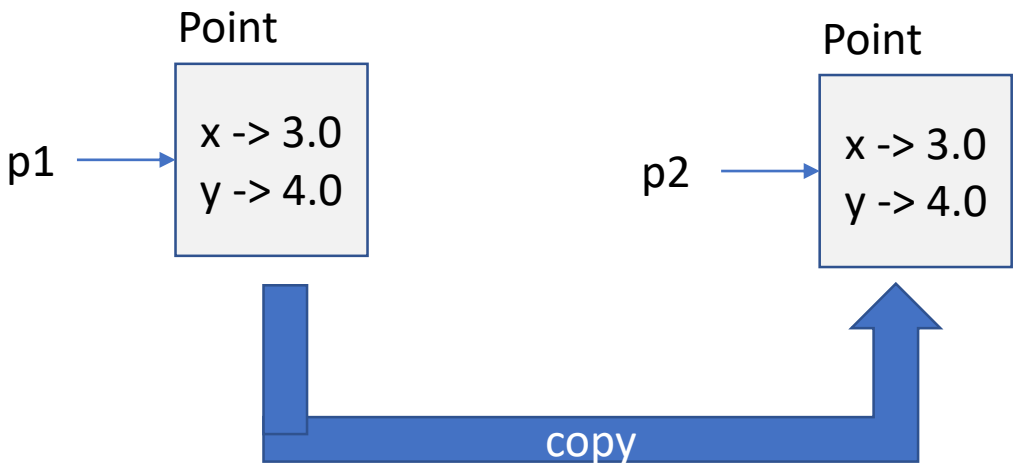
- We can change objects by altering their attributes
- Example: let's expand a Rectangle w/o changing its position

```
# continue with boxing…

 box.width += 50
 box.height += 100

# let's write a function to expand a Rectangle
 def grow_rectangle( rect, dwidth, dheight):
     rect.width += dwidth
     rect.height += dheight

# let's try it out
 box.width, box.height
(100, 200)

 grow_rectangle( box, 50, 100)
 box.width, box.height
(150, 300)
```



Copying

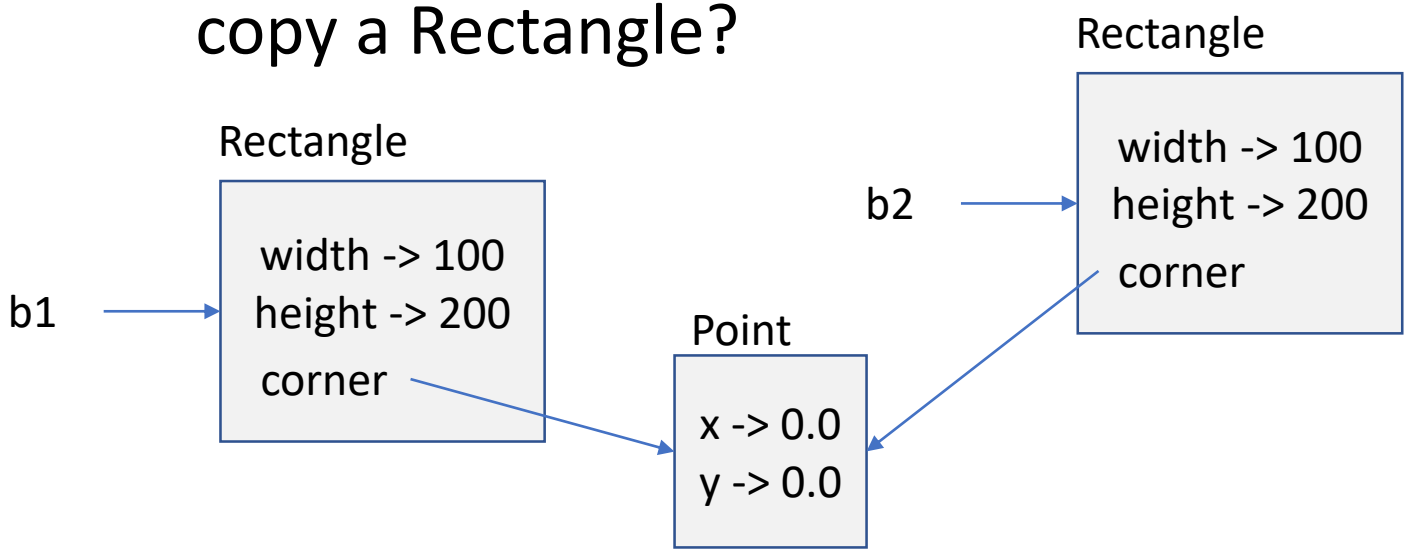
- Aliasing can make it hard to keep track of variables
- Copying is an alternative to aliasing
  - uses the **copy** module
  - creates separate copy of any object
  - copies are considered distinct (== will fail)




```
# use the copy module
 import copy

 p1 = Point()
 p1.x = 3.0
 p1.y = 4.0

 p2 = copy.copy( p1 )

# p1 and p2 contain same data,
# but are NOT the same instance (object)

 print_point( p1 )
(3, 4)
 print_point( p2 )
(3, 4)
 p1 is p2   # checks instances only
False
 p1 == p2 # checks instances only
False
```

Copies (cont)

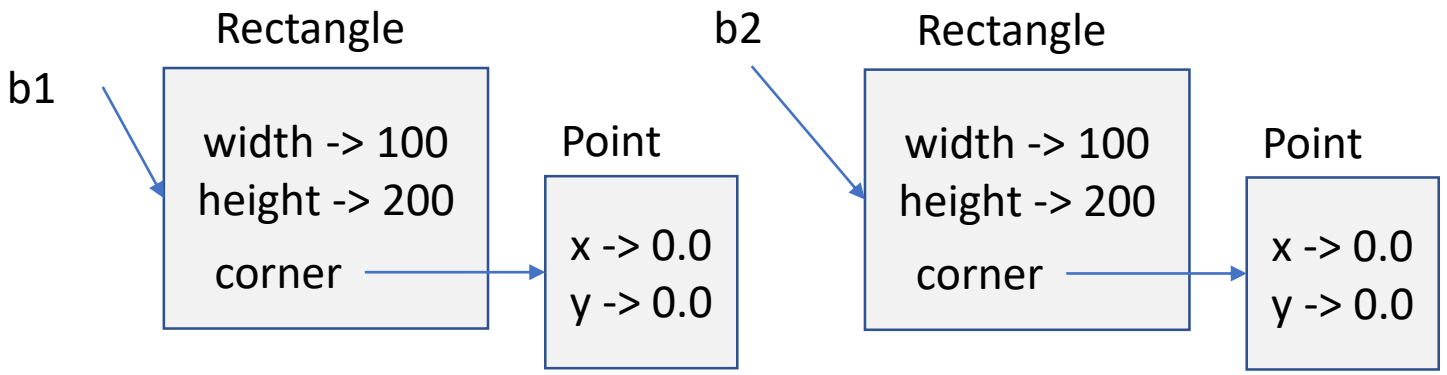
- The copy operator will copy all attributes
  - will not copy embedded classes
  - called "shallow copy"
- Example: what happens if we copy a Rectangle?




```
# more use of the copy module
 b1 = Rectangle()
 b1.width = 100
 b1.height = 200
 b1.corner = Point()
 b1.corner.x= 0.0
 b1.corner.y = 0.0

 b2 = copy.copy( b1 )

 b2 is b1
False

# but we share the corner!
 b2.corner is b1.corner
True
```

Copies (still more cont)

- The deepcopy operator will copy all attributes
  - will copy embedded classes
  - called "deep copy"
- Example: what happens if we deepcopy a Rectangle?




```
# more use of the copy module
 b1 = Rectangle()
 b1.width = 100
 b1.height = 200
 b1.corner = Point()
 b1.corner.x= 0.0
 b1.corner.y = 0.0

 b2 = copy.deepcopy( b1 )

 b2 is b1
False

# now we don't share the corner!
 b2.corner is b1.corner
False
```

Summary

- You can define your own data types in Python using the class operator
  - data values associated with your class are called "attributes"
  - functions associated with your class are called "methods"
- Python copy module provides several ways to copy classes
  - the copy() method does a "shallow copy"
  - the deepcopy() method does a "deep copy"