CS1 Lecture 26 Mar. 24, 2017

- HW6 due next Wed.
  - For question 2
    - Ensure legal moves – i.e. if user enters an illegal choice, print something appropriate and ask for a new choice.
    - Computer gameplay can be random (but must be legal). You can use, for instance, random.randint(…) for choosing number of balls. It's not required, but you can make computer smarter than random if you wish.
  - Question 3 – simple use of inheritance – will be added today

Last time

- Ch 15 and 16: classes and objects

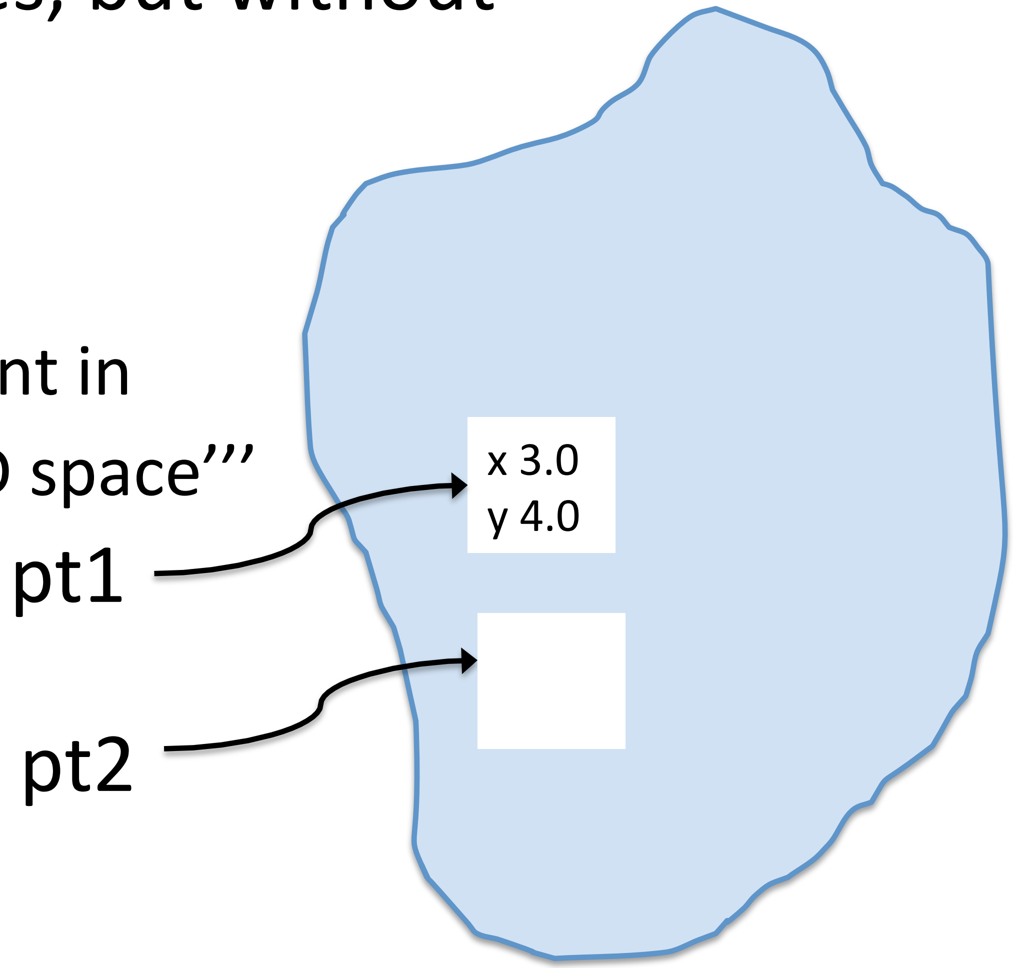
Today

- Chapter 17: classes and methods

Ch 15

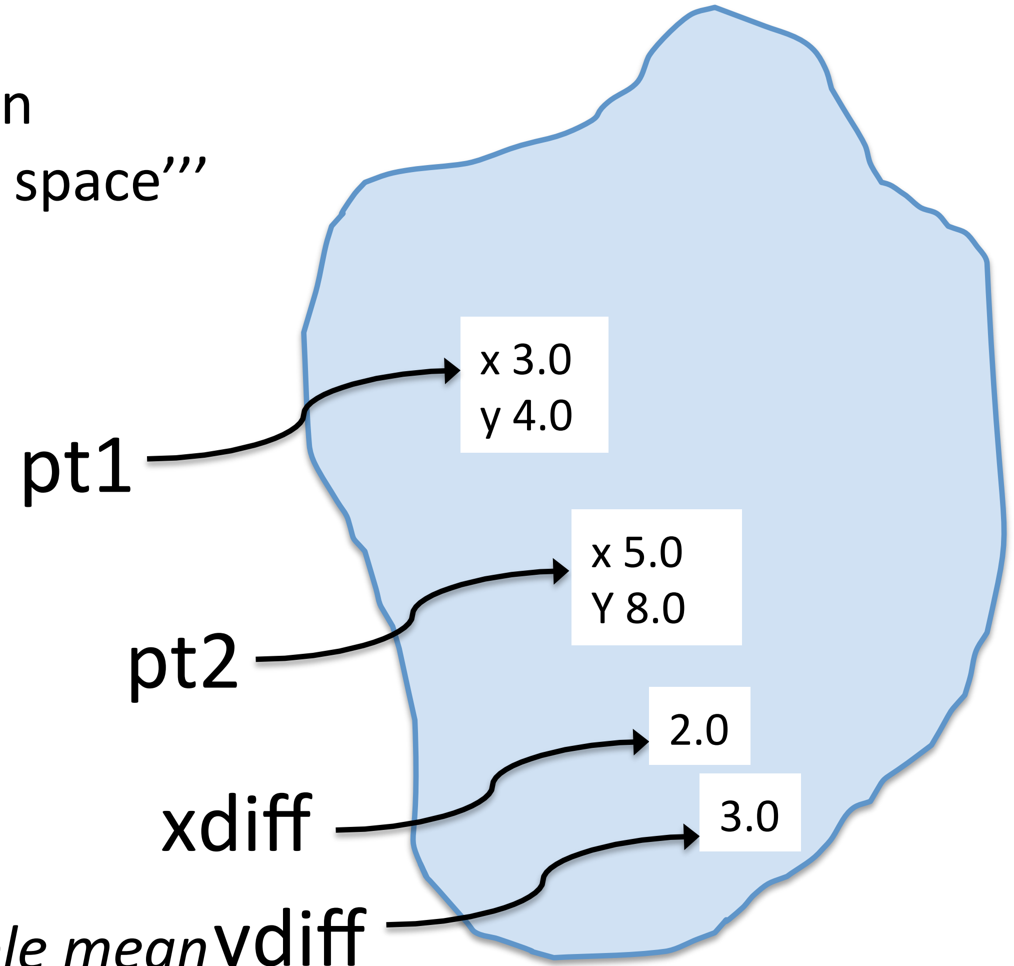
- Demonstrates classes as simple containers of attributes, but without methods.

E.g.
```
>>> class Point:
        '''represents a point in
                      2D space'''

    >>> pt1 = Point()
    >>> pt1.x = 3.0
    >>> pt1.y = 4.0
    >>> pt2 = Point()
```



Ch 15

```
E.g. >>> class Point:
                '''represents a point in
                                    2D space'''

     >>> pt1 = Pt()
     >>> pt1.x = 3.0
     >>> pt1.y = 4.0
     >>> pt2 = Pt()
     >>> pt2.x = 5.0
     >>> pt2.y = 8.0
     >>> xdiff = pt2.x – pt1.x
     >>> ydiff = pt2.y – pt1.y
```



This style is not really what people mean by "object-oriented programming." But, it is important to know how to access attributes since the methods you write in class definitions will access attributes directly

Ch 16 – using classes in functions

- Mainly contains some examples of functions that do things with instances of user-defined classes like those in Ch15 (that have no methods).
- Again, we don't really want to do things this way – writing top-level functions that access object attributes directly (point.x, etc.)
- HOWEVER, it can be useful transition
- Last time, demonstrated a Time class using the Ch 15/16 approach and then converted it to more standard object-oriented style of Ch 17.

  ch16time.py vs. ch17time.py

Ch 17

- This is the chapter to study most carefully
- General rule for defining classes:
  - define an init method initializing values for all properties/attributes (e.g. hour, minutes, seconds for time)
  - define methods that represent the “public interface” to the class. Users should work with instances of the class only via these methods rather than by accessing object attributes directly

Init methods

```
class Time:
    def __init__ (self, hour = 0,
                        minutes = 0,
                         second = 0):
        self.hour = hour
        self.minutes = minutes
        self.seconds = seconds
```

When you create an object using “constructor”: e.g. Time(…)

1. Python first creates empty object
2. Passes that empty object to __init__ with any additional arguments provided to constructor
3. returns the new object (even though there is no “return” line in init)

Notes on development of classes

- __repr__ and __str__ methods: used to define how object displays/gets converted to string. Book discusses similar/related __str__. Many Python programmers don't know the distinction between the two. You don't need to know. If you're only going to define one, define __repr__. (However, many say best practice is: __repr__ should produce string that is what you would type in to create object similar object. Not always followed ...) lec26ch17time.py

Notes on development of classes

- Very nice feature: you can "overload" operators. That is, you can define how +, -, <, etc. apply to instances of classes you define
  - __add__ for +
  - __lt__ for <
  - __eq__ for ==, etc.

    lec26ch17time.py
- AGAIN, avoid directly accessing object properties. Use only methods. This allows changing internal object implementation. lec26ch17timeAlt.py

HW 6

Q1 – 3D Box class.

Demo similar class: lec26circle.py

Next time

Finish our quick look at object-oriented programming:

Ch 18 – inheritance