Mostly skipping chapters 7 and 8, but worth reading anyway!

More Python features

- Key items from chapter 7:
  - Deeper while loop discussion (see pp. 243-247, and review chapter 5 notes ("more control structures")
  - Additional file processing (see pp. 253-256)
- Frequency counting in chapter 8 (pp. 271-276)
- Other good parts of chapter 8 (*but not on exam*):
  - String method `join` – opposite of `split`

```
>>> " ".join(mySet) # uses " " to separate items
'bat hat cat’
```

  - Pattern matching and regular expressions (pp. 291-302) – about more complicated searching

Starting chapter 9 (just covering through p. 315 though)

Recursive functions

- Definition: functions that call themselves, directly or indirectly
- But *proper* recursive functions also stop!

```
def factorial(n): # return n! = n(n-1)(n-2)…1
    if n > 1: # recursive step: call self for n-1
        return n * factorial(n-1)
    return 1  # base case: stop recursion if n < 2
```

  – Must have (at least) one base case, and the recursive step must converge on a base case
    - Otherwise "infinite recursion"
- See/try first two functions in …/demos/ recursive.py

Recursive drawing examples

- Listing 9.2 (also in recursive.py) – uses drawSquare function from chapter 2

  ```
  def nestedBox(aTurtle,side):
      if side >= 1: # recursive step
          drawSquare(aTurtle,side)   # A
          nestedBox(aTurtle,side-5)  # B
      # base case: do nothing (side too small to draw)
  ```

  - Note: switch lines A and B – will draw smallest first
- Draw tick marks on a ruler (recursive.py again)
- Listing 9.4 – draw nested triangles
  - Note demo introduces command line argument too
- Listing 9.3 (and exercises 9.11-9.13) – draw tree

Introducing chapter 10 (won't be on exam)

OOP and Python classes

- Essence of object-oriented programming:
  - An object is an instance of a class
  - The class defines what data an object knows, and what operations an object can carry out
    - Instance data – what an object knows: its state
    - Methods – what an object can do
- Objects of Python's class Turtle for example:
  - Instance data include color, heading, position
  - Methods include forward, backward, penup

Example: class Planet

- In Python – a class's constructor defines what an object of the class will know

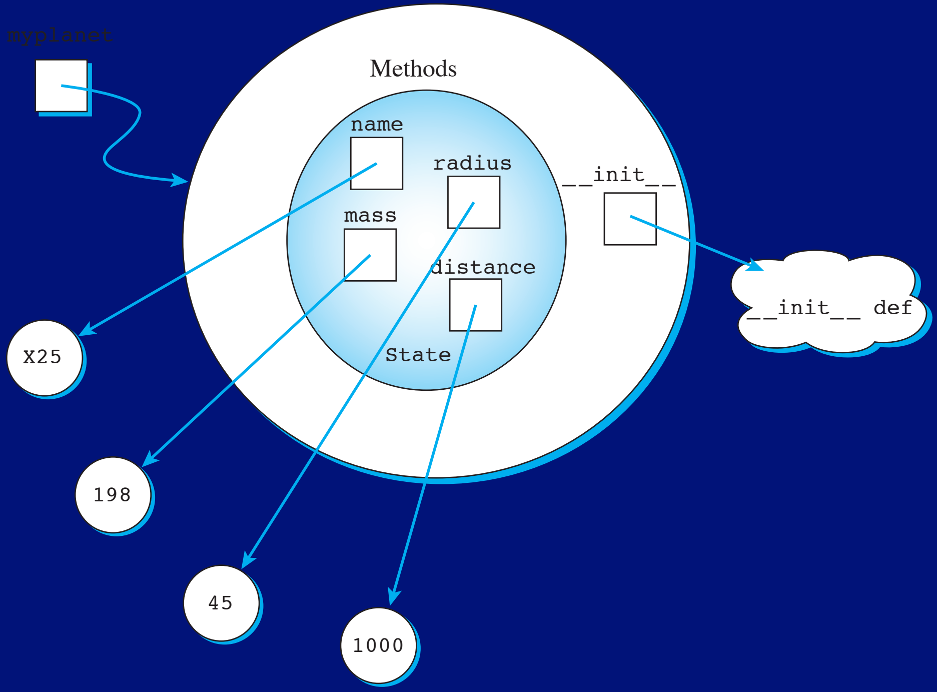
```
class Planet:
    def __init__(self, iname, irad, im, idist):
        self.name = iname
        self.radius = irad
        self.mass = im
        self.distance = idist
    ...
```

- A Planet object will know its own name, radius, mass, and distance from the sun

Constructing a Planet object

- Creating an object invokes the constructor

```
>>> myplanet = Planet('X25',45,198,1000)
```



Adding some Planet methods

- Accessor methods to access the data values

```
def getName(self):
    return self.name
```

  - Also `getRadius`, `getMass`, `getDistance`
- Mutator methods to change the data values

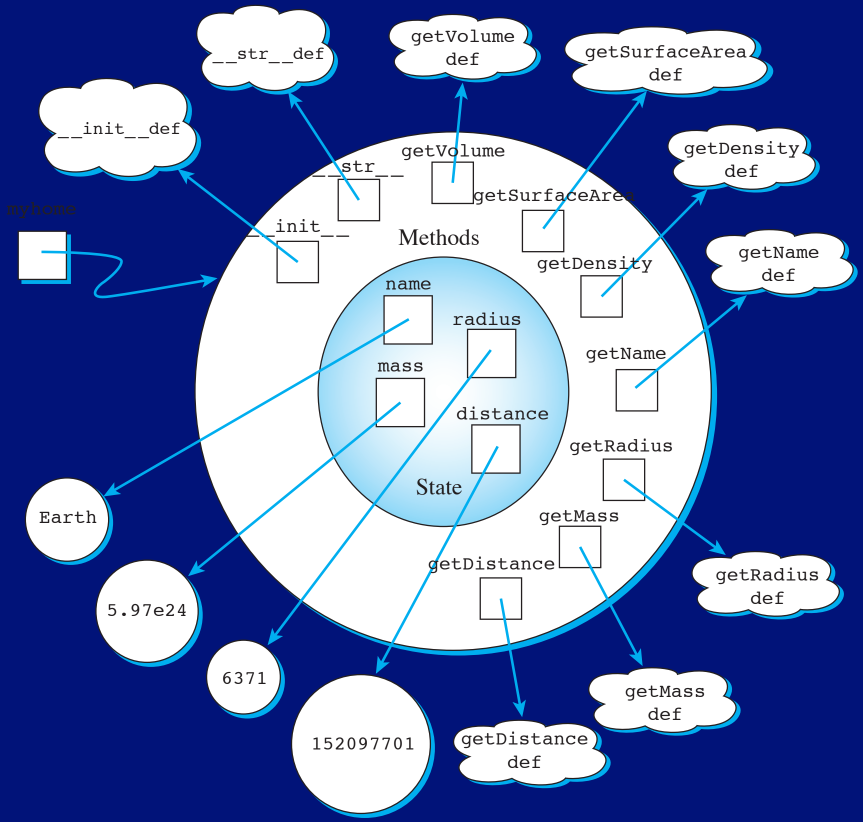
```
def setName(self, newname):
    self.name = newname
```

  - Also `setRadius`, `setMass`, `setDistance`
- Special method for converting Python object to str

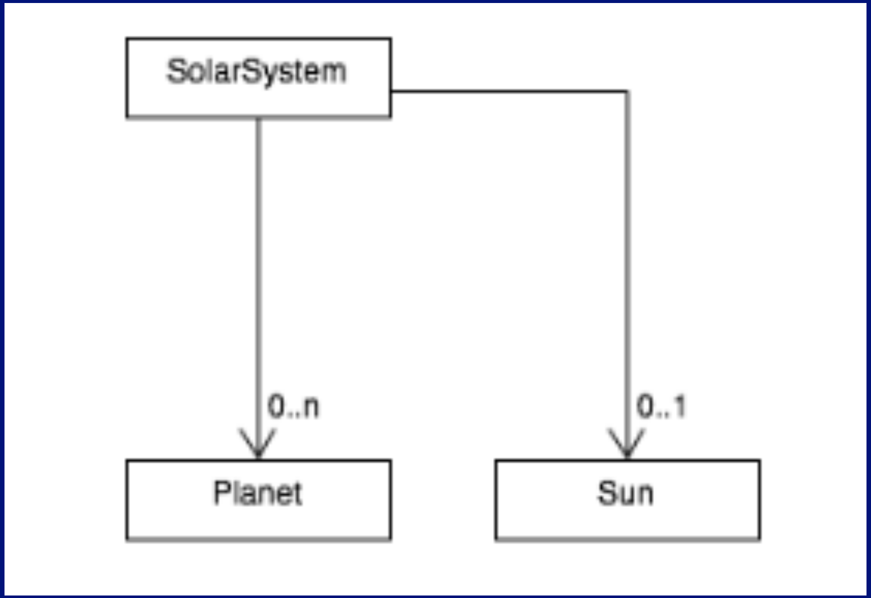
```
def __str__(self):
    return self.name
```

class Planet

A more complete Planet object



OOP example:
Modeling a solar system



Animated solar system - Planet, Sun and SolarSystem classes, plus a function to create and animate the parts