Agenda

- Announcements
- Snarf code “CompSci101Recursion”
Recursion

- Recursion: Se recursion.
Recursion

• Functions can call other functions, but they can also call themselves — that is recursion!
Recursion

• It is not a function or library
• It is a new way of structuring your code to solve problems.
• Recursion is based on the divide and conquer way of thinking:

  - Solve 1 small piece of the problem and call yourself with the rest of it.
Recursion example # 1

- A simple recursive function that counts down to “Blastoff”

```
def countdown(n):
    if n == 0:
        print 'Blastoff!'
    else:
        print n
        countdown(n - 1)

countdown(5)
```
Try example
Recursion example #2

• How to count files and folders?
Recursion example # 2

• Solve a big problem by
  – Dividing it into a smaller problem that can be solved. Identical in structure.
Recursion on a structure

• Do something (like counting) on the place were you are…

• …then call your self on a new element.
Recursion on a collection

• If else structure:
  – If the problem is small enough to be solved, then do it, otherwise divide it up and call yourself.

• Do something to a small part of the problem, then divide and call yourself on the rest

```python
def countdown(n):
    if n == 0:
        print 'Blastoff!'
    else:
        print n
        countdown(n - 1)

countdown(5)
```
Question

What is printed?

A) 100.0  7.25 ,  
    1000.0  72.5 ,  
    10.0  0.725 ,  
    Done

B) 100.0  72.5 ,  
    1000.0  725.0 ,  
    10.0  7.25 ,  
    Done

C) Error

```python
def vatCalculator( meatLst ):
    vat = 7.25

    if len( meatLst ) == 0:
        print "Done"
    else:
        pay = meatLst[0]
        tax = (pay * vat) / 100.0
        print pay, tax, ",”
        vatCalculator(meatLst)

lionFood = [10.0, 1000.0, 100.0 ]
vatCalculator( lionFood )
```
Recursion in sorting

- Quicksort can be solved via recursion