CSC148 fall 2013
more recursion, testing
week 4

Danny Heap
heap@cs.toronto.edu
BA4270 (behind elevators)
http://www.cdf.toronto.edu/~heap/148/F13/
416-978-5899

October 3, 2013
Outline

Class design for cheese

Recursion on nested lists

Testing, big and small
Separation of concerns

DomainStools $\rightarrow$ ManualController $\leftarrow$ CheeseView
Define the nesting-depth of L as 1 plus the maximum nesting depth of L’s elements if L is a list, otherwise 0.

- the definition is almost exactly the Python code you write!

- start by writing return and pythonese for the definition:
  
  ```python
  return 1 + max([nesting_depth(x) for x in L]) if ... 
  ```

- deal with the special case of a non-list
maximum number in nested list

Use the built-in max much like sum

- how would you find the max of non-nested list?
  \[ \text{max}(\ldots) \]

- how would you build that list using a comprehension?
  \[ \text{max}([\ldots]) \]

- what would you do with list items that were themselves lists?
  \[ \text{max}([\text{rec_max}(x) \ldots]) \]

- get some intuition by tracing through flat lists, lists nested one deep, then two deep...
get some turtles to draw

Spawn some turtles, point them in different directions, get them to draw a little and then spawn again...
This is a job for recursion:

\[ M(n) = \begin{cases} 
1 & n == 1 \\
\min \left\{ 1 \leq i < n \mid 2 \times M(n - i) + 2^i - 1 \right\} & \text{otherwise.}
\end{cases} \]

That's a recursive formula. Python has a built-in function \texttt{min}. You probably want to combine (tuple?) the minimum number of moves with the split \((i)\) that produces it.
before and after coding:

Test your docstring examples automatically:

```python
if __name__ == '__main__':
    import doctest
    doctest.testmod()
```

For more thorough testing, use `unittest`