CSC148 winter 2014
more recursion
week 4

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

January 29, 2014
Outline

Class design for cheese wrangling

Recursion on nested lists

Testing, big and small
Separation of concerns

Tour $\leftarrow$ TOAHModel $\rightarrow$ ConsoleController
This is a job for recursion:

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

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.
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([nested_depth(x) for x in L] + [0])
    if isinstance(L, list) else 0)
  ```

- deal with the special case of a non-list
trace to understand recursion

Trace in increasing complexity; at each step fill in values for recursive calls that have (basically) already been traced

- Trace `nested_depth([])`
- Trace `nested_depth(17)`
- Trace `nested_depth([3, 17, 1])`
- Trace `nested_depth([5, [3, 17, 1], [2, 4], 6])`
- Trace `nested_depth([14, 7, [5, [3, 17, 1], [2, 4], 6], 9])`
maximum number in nested list

Use the built-in max much like sum

- how would you find the max of non-nested list?
  max(...)

- how would you build that list using a comprehension?
  max([...])

- what would you do with list items that were themselves lists?
  max([rec_max(x) ...])

- get some intuition by tracing through flat lists, lists nested one deep, then two deep...
trace the recursion

trace from simple to complex; fill in already-solved recursive calls

► trace \text{rec\_max}([3, 5, 1, 3, 4, 7])

► trace \text{rec\_max}([4, 2, [3, 5, 1, 3, 4, 7], 8])

► trace \text{rec\_max}([6, [4, 2, [3, 5, 1, 3, 4, 7], 8], 5])
get some turtles to draw

Spawn some turtles, point them in different directions, get them to draw a little and then spawn again...

Try out tree_burst.py
You will have noticed that a recursive function has a conditional structure that specifies how to combine recursive subcalls (general case), and when/how to stop (the base case, or cases).

What happens if you leave out the base case?
before and after coding:

Test your docstring examples automatically:

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

For more thorough testing, use `unittest`