Recursion (continued)

CS 101 - Spring 2016
Creating a recursive solution

Base case:
- A trivial and easily solvable instance of the problem

Recursive case:
- Break the problem up into solvable problems and smaller versions of the same problem [must make progress toward the base case]
- Make the problem smaller by looking at smaller numbers, less data, or fewer choices
- Figure out how to combine the solutions to smaller problems to get the solution to the overall problem
Examples today:

- Countspace, Fibonacci
- Recursive Spirals
- Towers of Hanoi
def spaceCount(text):
    spaceLocation = text.find(' ') # Find index of first space
    if spaceLocation == -1:
        return 0 # Return 0 if no spaces
    else:
        return 1 + spaceCount(text[spaceLocation+1:]) # Recursively count spaces after first one
def spaceCount(text):
    spaceLocation = text.find(' ')  
    if spaceLocation == -1:
        return 0
    else:
        return 1 + spaceCount(text[spaceLocation+1:])
def spaceCount(text):
    spaceLocation = text.find('  ')
    if spaceLocation == -1:
        return 0
    else:
        return 1 + spaceCount(text[spaceLocation+1:])
Recursive Function Definitions

- Example 2: n\textsuperscript{th} value in Fibonacci sequence
  - 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

- Recursive definition of Fibonacci:
  \[
  \begin{aligned}
  \text{Fib}(n) &= \begin{cases} 
  0 & \text{if } n = 0 \\
  1 & \text{if } n = 1 \\
  \text{Fib}(n-1) + \text{Fib}(n-2) & \text{if } n > 1
  \end{cases}
  \end{aligned}
  \]
Recursive functions in Python

```python
def factorial(n):
    if n <= 1:
        return 1
    else:
        return n * factorial(n-1)

def fib(n):
    if n <= 1:
        return n
    else:
        return fib(n-1) + fib(n-2)
```
def spiral(len, level):
    if level > 0:
        turtle.forward(len)
        turtle.left(30)
        spiral(len * 0.95, level-1)
def spiral(len, level):
    if level > 0:
        turtle.forward(len)
        turtle.left(30)
        spiral(len * 0.95, level-1)
        turtle.right(30)
        turtle.backward(len)
def spiral(len, level):
    if level > 0:
        turtle.forward(len)
        turtle.left(30)
        spiral(len * 0.95, level-1)
        turtle.right(30)
        turtle.backward(len)

Call 3 times:
spiral(50, 50)
turtle.left(120)
Drawing a Tree
def drawTree(levels, len, angle, shrink):
    if levels > 0:
        t.forward(len)
        t.left(angle)
        drawTree(levels-1, shrink * len, angle, shrink)
        t.right(2*angle)
        drawTree(levels-1, shrink * len, angle, shrink)
        t.left(angle)
        t.backward(len)
Towers of Hanoi

Move $n$ disks from pole A to pole B:

- move 1 disk at a time
- never place a larger disk on top of a smaller one
- use the extra pole for “temporary storage”
# Towers of Hanoi

How many moves to solve puzzle for n disks?

<table>
<thead>
<tr>
<th>n</th>
<th># moves</th>
<th>[= 2^n - 1 ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>4</td>
</tr>
</tbody>
</table>
Towers of Hanoi

Move $n$ disks from pole A to pole B:

1. Move top $n-1$ disks from A to C
2. Move largest disk from A to B
3. Move $n-1$ disks from C to B