Recursion (continued)

CS 101 – Fall 2017
Creating a recursive solution

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

Recursive case:
- A set of steps that divide the current problem into smaller versions of the same problem [this dividing must make progress toward the base case]
- Make the problem smaller in some way (e.g. by looking at smaller numbers, less data, or fewer choices, etc)
- Figure out how to combine the solutions to the smaller problems to get the solution to the overall problem
def go_back(n):
    if n == 0:
        print("Stop!")
    else:
        print("Go", n)
        go_back(n-1)
        print("Back", n)
Today:

- I will pose a problem
- You will help me come to a solution
- Before we begin: any general questions?
Count Spaces

Write a recursive function `count_spaces(text)` which takes a string as an argument, and returns back an integer representing the number of spaces in the string.

e.g.

```python
>>> count_spaces(“Hello CS 101!”)
2
```
Fibonacci

Write a recursive function fib(n), which takes in an integer n and returns the nth Fibonacci number

e.g.

>>> fib(8)
21
Fibonacci

- The sequence:
  - 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, …

- Recursive definition:

  \[
  \begin{align*}
  \text{fib}(n) &= \begin{cases} 
  0 & \text{if } n = 0 \\
  1 & \text{if } n = 1 \\
  \text{fib}(n-1) + \text{fib}(n-2) & \text{if } n > 1 
  \end{cases}
  \end{align*}
  \]

  - base cases
  - recursive case
Spiral

Given the following code, what addition needs to be made for the turtle to finish at the beginning?

```python
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)
Spiral

How would we make this drawing?
Towers of Hanoi

Objective:
Move $n$ disks to a destination (DEST) pole

Rules:
- Can only move one disk at a time
- Must move the top disk from a stack
- Cannot place a larger disk on top of a smaller one
- Can use the extra pole for “temporary storage”
Towers of Hanoi

Algorithm:

- Move top n-1 disks from SRC to TEMP
- Move target disk from SRC to DEST
- Move n-1 remaining disks from TEMP to DEST
## 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></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>
Count Midd

Write a function count_midd(s) that takes in a string (assumed lowercase) and returns back the number of times “midd” is seen in the string

e.g.

```>>> count_midd(“middlebury is full of middkids taking midterms”)
2```