Lecture 15: Recursion (Sections 5.8-5.10)

CS 1110
Introduction to Computing Using Python

Recursion

• Recursive Function:
A function that calls itself (directly or indirectly)

Examples

• Blast-off
• Gift in gift
• Family Trees
• Towers of Hanoi
• Deblanking

Tower of Hanoi

• Three towers: left, middle, and right
• n disks of unique sizes on left
• Goal: move all disks from left to right
• Cannot put a larger disk on top of a smaller disk

Divide and Conquer

Goal: Solve really big problem P

Idea: Split into smaller problems, solve, combine

3 Steps:
1. Decide what to do for simple cases
2. Decide how to break up the task
3. Decide how to combine your work

Decide what to do for simple cases

Move from left to right
Decide how to break up the task

- Simpler than the original task, slowly becoming the “simple case”
- Simple case

Decide how to combine your work

Hanoi(4)

Hanoi(4)

(uncover the big one)

Hanoi(3)

move the big one

Hanoi(3)

(cover the big one)

Simple case

4 Discs: High-level Idea

1. Move top three disks from left to middle
2. Move largest disk from left to right
3. Move top three disks from middle to right

Recursion vs Iteration

- Recursion is provably equivalent to iteration
  * Iteration includes for-loop and while-loop (later)
  * Anything can do in one, can do in the other
- But some things are easier with recursion
  * And some things are easier with iteration
- Will not teach you when to choose recursion
- We just want you to understand the technique

Divide and Conquer

**Goal:** Solve problem P on a piece of data

**Idea:** Split data into two parts and solve problem

```
data 1  data 2
```

Solve Problem P  Solve Problem P

Combine Answer!

Putting it All Together

```
def deblank(s):
    """Returns: s w/o blanks""
    if s == 
        return 
    elif len(s) == 1:
        if s[0] == 
            return 
        else:
            return s
    left = deblank(s[0])
    right = deblank(s[1:])
    return left+right
```

Base Case

Recursive Case