Class 16

Recursion

Recursion

- Use a dictionary to look up an unknown word
- What if the definition in the dictionary contains a word we don't know?
- We use the same dictionary to look up this new word
- Continue looking up unknown words until we have learned the meaning of all the unknown words

Recursion

- In a similar manner, we might have a function that solves a problem by using itself to solve a smaller version of a problem
- Recursion means “when a thing is defined in terms of itself”
- In programming, recursion happens when a function calls itself *within its own definition*

Example 1

- Given integer *n*, write function to return left-most digit

Constructing a recursive function

Recursive functions have two parts:

1. A base case, in which the function can return the result immediately
1. A recursive case, in which the function must *call itself* to break the current problem down to a simpler level

Example 2

- Given integer *n*, write function to return sum of left-most two digits

Recursion

- Recursion is a programming technique
- Pro: Sometimes it is easier to write a recursive solution than an iterative solution
- Con: Sometimes the recursive solution requires too much memory to be workable

Example 3

- Factorial function

Benefits of Recursion

- While it takes a bit of practice to easily recognize how to decompose problems into recursive formulations, it can be one of the quickest ways to design an algorithm
- A recursive version of a function can sometimes be much simpler than an iterative version

Example 4

- write_vertical
  - Writes digits of a number vertically on a screen

Constructing a recursive function

- A recursive function contains a call to the function being defined
- The recursive call must accomplish a smaller version of the task (“Progress Condition”)
- The function must have one or more cases in which the task is accomplished without using a recursive call (“Base Cases” or “Stopping Conditions”)

Example 5

- Fibonacci sequence