EdX 6.00x Notes

Lecture 5:

- Iterative algorithms
  - Loop constructs (e.g. while or for loops) lead naturally to iterative algorithms.
  - Can conceptualize as capturing computation in a set of "state variables" which update on each iteration through the loop.
- Recursion
  - Recursive step: Reduce a problem to a simpler (or smaller) version of the same problem, plus some simple computations
  - Base case: Keep reducing until reach a simple case that can be solved directly.
  - Note: Examples of problem that can be solved by recursion are factorials. (e.g. n!) and Towers of Hanoi.
- Some observations (about recursion)
  - Each recursive call to a function creates its own environment, which local scoping of variables.
  - Bindings for variables in each frame are distinct, and binding are not changed by the recursive call.
  - Flow of control will pass back to earlier frame once function call returns value.
- Inductive Reasoning
  - How do we know our recursive code will work?
  - iterMul terminates because b is initially positive and decreases by 1 each time around loop; thus must eventually become less than 1
  - recurMul called with b = 1 has no recursive call and stops
  - recurMul called with b > 1 makes a recursive call with a smaller version of b; must eventually reach call with b = 1
- Mathematical induction
  - To prove a statement indexed on integers is true for all values of n:
    - Prove it is true when n is smallest value (e.g. n = 0 or n = 1)
    - Then prove that if it is true for an arbitrary value of n, one can show that it must be true for n+1
- Towers of Hanoi
  - The story:
    - 3 tall spikes
    - Stack of 64 different sized discs – start on one spike
    - Need to move stack to second spike (at which point universe ends)
    - Can only move one disk at a time, and a larger disc can never cover a small disc.
  - Solution:
    - Think recursively!

        - To move a stack of size n, move a stack of size n -1 onto the spare location
        - Then move the bottom disc where you are trying to go, then followed by the n-1 stack.
- Assert keyword
    - General Description: Checks to see if a statement is true and abort as if a fatal error had occured (or raise an exception) if the condition is false.
    - Useful link on how to use assert for projects: https://wiki.python.org/moin/UsingAssertionsEffectively
- Fibonacci
    - Example of a problem solved by recursion. Developed by Leonardo to simulate rabbit population growth.
    - Base Cases:
        - Females(0) = 1
        - Females(1) = 1
    - Recursive case:
        - Females(n) = Females(n-1) + Females(n-2)
- Palindrome Problem
    - Recursive solution:
        - First, convert the string to just characters, by stripping out punctuation, and converting upper case to lower case.
        - Then
            - Base case: a string of length 0 or 1 is a palindrome
            - Recursive case:
                - If first character matches last character, then is a palindrome if middle section is a palindrome
- Internal procedures:
    - When a procedure is defined inside of another procedure.
    - Belong only to the procedure they are defined in.
- Divide and Conquer Algorithm:
    - Solve a hard problem by breaking it into a set of sub-problems such that:
        - Sub-problems are easier to solve than the original
        - Solutions of the sub-problems can be combined to solve the original
- Global Variables:
    - Name is defined at the outermost scope of the program, rather than at the scope of the function where it appears.
    - Note: Can be dangerous due to scoping issues. Use it carefully because you are destroying the locality of the code.
    - Uses keyword **global**