Lecture 2
Fundamentals of Programming.

-- Write out the agenda for the lecture.
-- A challenge problem for those who are familiar with this.
-- If finished, then we should probably arrange to meet to discuss working on a programming project.
-- Arrange seating accordingly (front, working on lecture; back, working on problem).

1. RUNNING PYTHON
   i. E.g., print "Hello, world!"
   b. Interactive. It evaluates and prints.
      i. Terminal
      ii. IDLE
   c. Script. No automatic printing.
      i. Terminal
      ii. IDLE

2. LINEAR PROGRAMS
   i. E.g., \( m = 3 \) (#slope), \( b = 6 \) (#intercept), \( x = 1, y = mx + b \)

3. SIMPLE FUNCTIONS
   a. Everything inside the function. Very specific.
   b. Passing arguments. More abstract by leaving some variables as input.

4. BRANCHING PROGRAMS
   a. Cooking metaphor, if needed, add salt.
   b. Basic structure: if <some-test>: block of instructions  else: some other block of instructions.
      i. E.g., Least between two values, \( x \) and \( y \).
   c. Indentation.
   d. If does not need the else.
   e. Ifs can be nested: elif
      i. Is \( x \) smallest, medium, or largest, among \( x \), \( y \) and \( z \)?
   f. Boolean algebra can be used in the test.
   g. Test all possible branches.
      i. E.g., Even/Odd program. [#2.1]

5. ITERATION: WHILE LOOPS
   a. Calculate something that requires something to be repeated many times. E.g., average age of all students.
   b. Basic structure: while <some-test>: block of instructions.
   c. Examples.
      i. Show [#2.2]. What does it do? How do we figure it out? Run on the board.
      ii. Exercise [#2.3]: Program that asks for user to answer ‘yes’ or ‘no,’ and doesn’t quit until it gets the answer.
         1. Variation that quits after 3 tries.
   d. Basic considerations:
      i. Choose variable that will ‘count.’ Change through the iterations.
      ii. Initialize it outside the loop.
      iii. Set up the right end test. (Includes the variable that’s changing).
      iv. Construct the block of code (Variable must be changing).
      v. Decide what to do when done.

6. FLOW CHARTS
   a. (Factorial of X)
   b. Square root of a perfect square X.
      i. Implement. [#2.4]

7. PROGRAMMING STYLE
a. Defensive programming.
   i. What happens if the number is not a perfect square? What happens if the number is a negative number? [2.5]

b. Comments.
   i. Function description. Expected input/output.
   ii. Description of what piece of code is doing inside the function.

c. Variable names.
   i. Start with letter or underscore (not numbers). No spaces anywhere.
   ii. Some 28 keywords are blocked.
   iii. Meaningful.

d. Type discipline. Dynamic types.