We are introduced to **thinking like a computer scientist**.

1. Welcome!

   (a) Review the Syllabus.
   (b) Book: *Think Python 2e*, found on-line. Read before each class.
   (c) Ten problem sets/homeworks (10%).
   (d) Eight laboratory projects (30%).
   (e) One individual, multi-week project (20%).
   (f) Two exams (20% each). Mark this now: Midterm is March 6, no exceptions.
   (g) Six (!) teaching assistants: Geoffrey Lu, Javier Esparza, Julia Kawano, Louisa Nyhus, Mi Yu, and Tuan Tran. Available in lab (TCL312): Sundays 1-4 (*after 2/17*) and 7-10, Tuesdays 6-9, Wednesdays 7-10, Thursdays 2-5 (TPL207) and 7-10

2. Goals and responsibilities.

   (a) I want to teach you to think *algorithmically*—like a computer scientist.
   (b) We'll solve problems using implementations of algorithms as programs in Python.
   (c) We'll learn how to evaluate the effectiveness of our implementations.
   (d) We'll learn how to demonstrate correctness.
   (e) We'll learn approaches to isolating and removing program flaws or bugs.
   (f) We'll learn how to reduce effort through notions of abstraction and encapsulation.
   (g) We'll learn how fun it is to build useful tools.

3. **Dramatis personae**

   (a) **The Problem.** Something that we must conquer. In this class, the problems will often involve large amounts of data.
   (b) **The Algorithm.** A *theoretical* or *abstract* approach to solving the problem.
   (c) **The Hardware.** The computational vehicle that *executes instructions*, as precisely as possible. Dumb as sand.
   (d) **The Operating System, ‘OS’.** The always-running program that is an *abstraction* of the hardware. It allocates resources in careful ways. It blurs hardware differences so that software is portable and resilient.
4. The Plot, as it unfolds in several acts.

(a) Computers can solve many problems. There are some solutions that just take too long. There are problems computers cannot logically solve. The mindful programmer can spot the differences.

(b) Little problems are easy (and fun) to solve. Big problems are composed of little problems. The thoughtful programmer sees the way to reusing small solutions to conquer big problems.

(c) New problems are often old problems recast in subtle ways. The effective programmer builds powerful tools to solve classes of related problems.

(d) Our craft is artful
Favor beautiful programs
and shun the ugly.

5. A lick of Python.

(a) Let’s get a first, sacred program out of the way. What does this do?
   
   ```python
   print("Hello, world.")
   ```

(b) How many words are in Tom Sawyer (tomsawyer.txt)?

(c) How long is the average word in Tom Sawyer?

(d) How long is the average word in the dictionary (/usr/share/dict/words)?

(e) Thought problem: All of your fingers support the process of typing. If you were to type a text (say, Tom Sawyer), which finger would get the most use?