Computer Science 134C
Introduction to Computer Science, in Python
Lecture #1 (Hello, world!)
September 7, 2018

Keywords
Python, algorithms, programs, software, hardware, correctness, abstraction, encapsulation, problem solving, tool building

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) A dozen problem sets/homeworks (15%).
   (d) Ten laboratory projects (35%).
   (e) Two exams (25% each). Mark this now: Midterm is on the evening of October 16, no exceptions.
   (f) Teaching assistants: Will Burford, Elizabeth Button, Peter Christie, Javier Esparza, Abigail Fournier, Ariel Koltun-Fromm, David Lee, Grace Mazzarella, Dzung Pham, Alex Smith-Bove, Alex Taylor
   (g) TA hours: Sunday-Thursday 7pm-10pm, Tuesday 8am-11am, Wednesday 10-noon, Thursday 9-noon, TCL217a.

2. Goals and responsibilities.
   (a) We 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 personæ*
   (a) The Problem. Something that we must conquer. In this class, the problems have the potential to 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.

(e) The Software. The instructions that will actually solve the problem. Cast in a language that must be precisely understood by both the hardware+OS and the programmer.

(f) The Hero. The programmer...you.

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?

   ```
   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?