CSSC: Introduction to Programming

Curriculum Overview

Thomas Adriaan Hellinger

10/07/2017

Introduction - First Course

I'm not entirely sure how this is going to work, but I believe we should have multiple classes to cover the basics. I'd like to have one initial class that covers all the very basic, basics. I'm thinking that the first class will be a long one, and split into two parts, with a short break in between. The first session will cover the absolute minimal math requirements, a *very* quick history on the development of computing, a simplified review of the key components of modern computers (laptops and PCs, specifically), then an overview of programming languages, concluding with an install of the Python programming language. This should conclude the basic theory portion of the class.

After the break, and we have Python installed on everyone's laptops, we should start covering the basic tools to use. This should include the most basic command line stuff, possibly with Cygwin and/or IPython/Jupyter notebooks too. We should definitely introduce them to a developer text editor, I'd recommend Atom, as it isn't too much of a shock and it's also in the Free Software Directory[1] (although apparently there is Google tracking by default, but we can show folks how to disable it). We should finish the day with showing people the resources needed to learn more and extend their python install. Finally, we should work on a classic "Hello World" program, with a minor twist involving a library like random or os.

[1] https://directory.fsf.org/wiki/Atom

Introductory Courses - Suggested Curriculum

SO, this is a very basic suggestion that would need to be elaborated on. Also, the number of sub-sections don't necessarily correlate with the time spent on a particular section.

I **Initiation Course**

(a) Theory Portion

i. Minimal Math section

A. Simple Algebra - (Computers count from zero, exponents, Please Excuse My Dear Aunt Sally, etc)

B. Number systems - (Binary, Octal, Hexadecimal)

ii. History

A. Pre-mechanical computers

B. Babbage, his machine, and Lovelace, the first programmer

C. The analog computers of the war years, Alan Turing and the Turing machine

D. ENIAC and the Von Neumann Architecture

E. Semi-conductors and the first digital computers

F. The development of DARPANet, Bell Labs; Unix and C

G. Al Gore invents the Internetz out of a series of tubes

iii. So WTF is a computer, anyway?

A. The motherboard: Key components

B. RAM v. ROM: To the Death!

C. Video cards, the GPU, and monitors: The things that make you see things

D. Peripherals: Keyboards, mieces, and joysticks

iv. The Big Overview of How a Programming Language Becomes 0's and 1's (Trade offs for each level away from binary code)

A. Machine Code - 0's and 1's

B. Assembly Language

C. C

D. C++ and Java

E. Interpreted Languages (Python, Ruby, Perl, etc.)

F. "Formatting Languages" (HTML, XML, JSON, etc)

G. Brief mention of "specialized languages" (R, SQL, LaTeX, etc) ((Maybe))

(b) Practice Portion

i. OMG! A commandline appears! *WHAT DOES IT WANT!?*

A. When to use the commandline and why

B. Walking the Path (ls/dir, mv, cd, mkdir, rmdir)

C. Files/directories and data/executable files; file extensions

D. Checking your privilege, digitally speaking

E. Quick note on the python 2/3 gap

F. Charming the Python (and maybe the IPython? On Jupyter?), and pip for a GREAT JUSTICE

ii. Let's make us a program

A. This is a text editor. It edits text, and text is code.

B. Importing stuff, and the magic of using other people's code

C. Hello world, here is a random number for you (maybe mated with a dict to determine how you happen to be feeling).

D. Make that code dance!

iii. Places to go, things to read and where to find answers

A. learnpython.org (in fact we may want to do go through several of the beginner tuts here before making a program)

B. stackoverflow.com (because there is a 99.5% chance that someone has asked whatever you're about to ask. Don't re-ask the wheel)

C. docs.python.org (the Fucking Manual for Python to RTFM)

D. "Learn Python 3: the Hard Way" by Zed Shaw for those who want to do things the right way

So that's a very basic outline of the first class. We need to flesh this out, of course, and figure out what is important. Certain things might have to be deleted, or added. Right now, it seems that the theory part might be too heavy, and we might want to streamline that. Conversely, this might just be an indication that the practice part is too light and needs to be expanded a bit.

For now, I'm pretty sure that this course can be delivered in under the time required to watch a Lord of the Rings movie in theaters, even with the intermission that those movies desperately needed. I think, anyway (the first part, not the intermission part).