0.1 No physical copy

Also student tend to have computers in bad shape

0.2 Instructor Time Sink

0.2.1 Coordination With Canvas

0.3 Additional Login

Proposal for Adopting and Adapting an Open-Source Curriculum and Textbook for Python

Andrew Rosen
Assistant Professor of Instruction

May 3, 2019

1 Premise and Motication

CIS 1051 is one of two introductory programming courses offered by the Computer and Information Sciences Department. Students must take this course or CIS 1057 (An introductory programming course in C) before moving onto CIS 1068 (a more advanced programming course in Java). CIS 1051 is geared towards students who have no background in computer science or programming, with only the most rudimentary math prerequisites. This is the only programming course many students take.

I currently teach most of the sections for 1051, which amounts to about 150 students. The typical textbook costs between $60 to $150, depending on the source (the overall cost is left as an exercise to the reader). This is typical for a computer science textbook from a major publisher. This cost leads many students to just pirate the textbook (a behavior which many faculty turn a blind eye to), rely on the slides, or do without.

Previous semesters have utilized *Think Python*, by Allen Downey, who is quite passionate about free books and has written a number of them for computer science and programming topics. *Think Python* is a well known free textbook, licenced under the CC BY-NC 3.0 creative commons license. The hard copy MSRP is $45, but typically sells for $30. *Think Python* is well written but the textbook is short, sparse on examples and exercises, and perhaps just a bit *too* concise. I have found our instructors typically just used as a reference rather than a companion for learning and diverge from the material at some point.

This most recent semester used *Automate the Boring Stuff with Python*, by Al Sweigart, also under a creative commons license (CC BY-NC-SA 3.0). *Automate the Boring Stuff with Python* is an excellent book, with numerous examples, jokes, and perhaps a slightly unhealthy obsession with RoboCop. It is intended for an audience of learners who do not intend to go into a programming related career, but might find their daily work task improved by programming.

It includes extremely useful topics that are not normally included in most introductory textbooks, such as Regular Expressions (think powerful find/replace functionality) and using user made software libraries.

It is an excellent text for self learning, but as a textbook for students it is lacking in some areas. First, the order of material is not always the best. It is also sparse on the number of exercises and most of them are quite simple. Finally, it does not go over turtles, an extremely basic and elucidating graphics library.

2 Plan

I plan on using *How to Think Like a Computer Scientist: Interactive Edition* from Runestone Interactive. Runestone Interactive is an organization that provides open source, online textbooks. *How to Think Like a Computer Scientist* specifically has a number of features.

- Completely free.
- Open source and forkable. Instructors can modify the textbook to include additional exercises or material.
- Students can check off their progress as they read, allowing the instructor to keep track of students keeping up with the material.
- Interactive code blocks. Code can be run, executed, and modified in the browser. This isn't a completely unique feature, but it is executed very well with a low amount of overhead on the user's browser. This is important, as poor implementations of runnable code can make pages difficult to use on weak computers (something lower-income students tend to be saddled with).
- Integrated turtle implementation. Students have reported that turtle graphics really enhanced their understanding of control structures used in programming, as they get an easy to see visual feedback.
- A code lens feature: Users can step thru the code on line at a time and see how variables are affected.
- Self checking exercises. The book has a number of different types of autograding exercises, as well as support for creating your own exercises, such as:
  - Multiple choice.
  - Parson puzzles, where users drag and drop blocks of code in the correct order.
  - Fill in the blanks
  - Timed Exams

  - Pop-up questions in codelens.

- Student polls and feedback integrated into the pages, so students can report how difficult a certain topic is.

By contrast, the prior two books mentioned only have traditional self check exercises and executable code blocks online. My plan is to examine using the textbook as-is, supplemented by content and exercises from `python-sneks`. `python-sneks` is another open source curriculum for introductory, but is newer and not as established Runestone Interactive. It provides a number of resources for intructors, such as quizzes, assignment, projects, and slides. The other two textbooks will be listed as supplements for students who need additional reading.

3 Notes

I can foresee a few difficulties with this course:

- There will be a few inevitable errors, typos, and integration problems stemming from using the technology the first time around.

- No physical copy. Some students prefer a physical copy, but will be unable to use one as the text is completely electronic.

- The course necessitates access to a reliable computer. Some lower income students are saddled with personal computers with low-specs, cracked screens, and/or non-functional keyboards. These students can be accommodated on an individual basis.

- Teaching a student who is blind or visually impaired always present a great challenge in programming, perhaps moreso in Python, where indentation is key factor in making sure programs built correctly. Such students would need to work closely with the instructor to address any concerns. However, the html layout of the textbook will be slightly easier for a screenreader to read the text, as opposed to a pdf.

However, the benefits greatly outweigh these considerations. Students will be asked to report their satisfaction or suggested changes throughout the textbook, as well as on the exams. Effects on student performance can be evaluated by comparing exam and overall scores to those from Spring 2019. Furthermore, understanding can be evaluated using the integrated textbook exercises.