A Nine-week Afterschool School Curriculum for Introducing High School Students to the Principles of Computer Science

Nick Goel | CPSC 490 | Advisor: Jason Hirschhorn | Spring 2016

Motivation

As job opportunities in computer-related industries continue to grow and as the importance of computing to the economy at-large increases year after year, computer science education in American high schools continues to lag. Recent estimates suggest that fewer than 3 in 10 high school students in the US attend a school that offers any formal coursework in computer science (1). And in only 29 states do computer science classes even count toward high school graduation requirements (1). Furthermore, glaring underrepresentation of women and racial minorities studying computer science has improved only slightly in the last decade. Only 18% of students taking AP Computer Exam in 2013 were women, 3% were black, and 6% were Hispanic (2). Very similar demographics play out among students pursuing CS degrees in college and in the computer technology workforce (3, 4). Most research points to the importance of early exposure in students developing a lasting interest in the subject (1, 2, 5). Perceptions of CS as difficult, inaccessible, and boring among students not yet exposed perhaps pose the greatest barrier to getting more students involved who have the capacity to succeed in the subject (1, 5). Among underrepresented groups, such negative perceptions are particularly prevalent (5).

Project Overview

Here I propose a nine-week curriculum for introducing high school students with no prior experience to the most basic principles of programming and computer science. The curriculum is designed for weekly two hour-long classes in an after-school setting, with an emphasis on problem solving, project-based learning, and self-motivated exploration. The goal of the curriculum is above all to be fun for students—to dispel their misperceptions about CS and to motivate them to continue to study CS in a more formal setting. The design and content of each weekly module are founded on research in lesson design and teaching strategy, as well as the latest tools and research in computer science education. Student’s will be introduced to the basic
ideas of programming through Scratch, a “media rich” programming environment from MIT’s Media Lab designed for beginners. Students will then transition to programming in Python after two weeks.

**Learning Programming with Scratch**

Scratch is a programming environment developed by MIT’s Media Lab and released in 2007. In Scratch students program with the mouse, as lines of code are represented by colorful, Lego-like puzzle pieces that snap into place when syntactically appropriate. Code then influences the behavior of characters, or “sprites,” that visually enact the program a student has written by moving across the “stage.” A growing body of research supports the idea that Scratch can be a highly effective tool for drawing students into the study of computer science (6, 7, 8). Scratch links the most fundamental aspects of computer programming to the types of activities that are more relevant to the interests of a student—like telling a story or building a game as opposed to solving math problems or printing out simple text on the screen (ie “Hello World”) (6, 7). Moreover, while the syntax of almost any formal programming language can seem at first abstruse or intimidating to the unfamiliar student, in Scratch lines of code interact in a way which is immediately intuitive (6, 8). In this sense Scratch “lowers the floor” for students getting started in programming yet has a remarkably “high ceiling” for creativity and self-motivated exploration (6). I plan to leverage these qualities Scratch in the first two weeks of my curriculum, before transitioning students to programming in Python.

**Curriculum Outline**

Each weekly lesson will involve goals and standards for the week, instructional material and class exercises, homework problems, and benchmarks for evaluation.

**Week 1: Introduction to Programming in Scratch** – using Scratch to introduce students to key programming concepts like conditional statements, loops, Boolean logic, variables, and functions

**Week 2: Building a game in Scratch** – focused on assisting students in building a self-directed project in Scratch, reinforcing programming concepts introduced the previous week
Week 3: Introduction to Python, variables and primitive data – introduction to Python syntax and programming environment, covering data types

Week 4: Functions and Information-Hiding - Learning how to program in functions, code reuse and information hiding

Week 5: Conditional Statements and Boolean data – Introducing the concept of conditional statements and boolean logic, and their usefulness in modeling and solving problems

Week 6: Loops - Introduction to “for” loops and “while” loops in Python, emphasis on more complex problem solving

Week 7: Arrays, Lists, and Strings - Introduction to the idea of data structures, string indexing and manipulation

Week 8: Problem Solving Techniques - reinforcement of tools learned in previous weeks through a more careful focus on how to approach logical problems, thinking through programmatic solutions, debugging

Week 9: Final Project—building a web crawler - synthesizing ideas from throughout the course into a final project—a web crawler or search engine program that can traverse web pages in search of key words, utilizing Python libraries, interacting with the web

Teaching at the Co-op After School program

For the past two semesters, I have taught an afterschool course in computer programming at the Co-op Arts and Humanities High School in New Haven as part of the Yale student-run group Code New Haven. With permission from school administrators, aspects of this curriculum will be implemented in a weekly afterschool class at the Co-op High School for the spring 2016 semester. General comments on the efficacy of different components of the curriculum will be included in the final report.

Deliverables

1. Complete curriculum, all supporting materials, problems, code snippets, etc, includes annotations and references to the research and prior work on which the content and design of the curriculum is based

2. Written report explaining the ideas behind each aspect of the curriculum, as well as general assessment on the implementation of the curriculum at the Co-op High School, general student reaction, unforeseen obstacles, etc
References:


