How do we introduce non-majors (and budding majors) to CS?

- CS0 Introduction to Computer Science
  - Typically a crash-course in every computer science topic in one class.
  - Students are bombarded with one week of each major area of CS.
  - How much do they really absorb?

- CS1 Computer Programming with $x$
  (where $x$ = Java, C, Scheme, C#, etc.)
  - Students spend a great deal of time learning how to "speak" in this new language (syntax).
  - Different paradigms don't help matters.
  - Some CS1 courses use micro-worlds, multimedia, robotics, etc.
Principles of Computation

• Survey of the major contributions and issues associated with computer science, focusing on the study of the process of computation.
  • CS is not viewed through a specific programming language.
  • CS is not viewed through a specific application area (e.g. multimedia, robotics, etc.)
• This course focuses on what it means to perform computation and what issues arise as mankind automates this process using computers.
• Designed for students who will probably take only one CS course in their lives.

Course Topics

• History of Computation
  • What societal needs caused mankind to make great advances in understanding and automating computation?
Course Topics

• Expressing Computation using Algorithms
  • Why do computations need to be expressed precisely?
  • How do we trace an algorithm to see what it is computing?
  • What common building blocks are used to specify algorithms?

Course Topics

• Organizing Data
  • Are there better ways to structure data than just using a linear arrangement (vector)?
  • In what situations would we use these structures?
Course Topics

- Expressing Computations to a Computer
  - How do we use programming languages to express our algorithms to a computer so it can compute them?
  - Why are there so many programming languages?
  - Does the computer "understand" any of them?

- "Tricks of the Trade" (Algorithmic Techniques)
  - How do we express computations recursively? Is this more intuitive?
  - How is "divide and conquer" used in computation?
  - Does a greedy approach to solving a problem always give the optimal answer?
Course Topics

• Perfecting Computation
  • How do we know if a computation is correct? Why is this important?
  • How do we measure how efficient a computation is? What does that mean anyway?

Course Topics

• The Limits of Computation
  • What makes a computation intractable?
  • Are all computations theoretically solvable?
  • Can we describe a universal computer?
Course Topics

• Parallel and Distributed Computation
  • What problems occur if multiple computational processes need to use the same resources? Can deadlock be avoided?
  • If we run a computation in parallel, how much faster do we get an answer?

Course Topics

• Applications
  • What are the computational aspects of public-key encryption?
  • How intelligent are computers anyway?
• Future of Computing

http://www.jyi.org/volumes/volume8/issue2/images/srivastava1.jpg
Exploring Computation Without Syntax

- Our course uses a public-domain program called RAPTOR, a flowchart simulator.
- Students can build simple procedural programs without learning the syntax details of a language.
- Contains conditionals, loops, input and output, arrays, subroutines, graphics.

http://raptor.martincarlisle.com/

Programming without Syntax

Four In A Row

15 Puzzle

Lights Out
Writing in a CS Course: Examining Social Aspects

- MP3 file sharing systems and copyright infringement
- Privacy and security of electronic data, the rise of identity theft
- Electronic voting systems, verification and security
- The role of online social networks: the good, the bad and the ugly

Enrollment

- Enrollment

- Requirements
  - Business: Acts as an alternative to Introduction to Programming.
  - Humanities/Social Science: Satisfies a Gen. Ed. requirement for modeling (mathematics and experiments).
Distribution of Majors by School

• Through Fall 2006:

(* H&SS = Humanities & Social Science - includes Information Systems)

Distribution of Majors by Year

• Through Fall 2006:
Distribution of Grades

- Through Spring 2006:

  ![Distribution of Grades Chart]

Student Feedback

- Continue the use of guest speakers?
  - Yes 80%
  - No 15%
  - Unsure 5%

- Continue the use of Raptor?
  - Continue
  - Continue but use more graphics 30%
  - Use another language/application 20%
  - Stay off the computer! 5%

- Did you take another CS class before or during this class?
  - Yes 20%
  - No 80%

- Are you interested in taking another CS class if you could?
  - Yes 55%
  - No 35%
  - Unsure 10%

- Would you recommend this course to your friends?
  - Yes 85%
  - No 15%
Describe this course in one sentence..

- "An alternative/substitute to [introductory programming]."
- "You learn a lot more than just how to code."
- "It's a lot better than doing real programming."
- "Understanding the world of computing beyond programming, and why and how programs and computers actually work."
- "It's a way to stay away from programming while actually learning something."
- "The fundamentals of programming."

*There's still so much work to do...*

Future Work

- Offer this course to high school students during the summer for college credit.
  - Aim at those high schools that have no CS courses beyond computer literacy.
- Experiment with non-traditional programming environments (e.g. Alice, Subtext).
- Track students who take additional courses in CS to see if this course had any effect on their interest and performance.
For More Information

- **15-105 Principles of Computation**
  - [http://www.cs.cmu.edu/~tcortina/15-105fa06](http://www.cs.cmu.edu/~tcortina/15-105fa06)
- **Raptor**
  - [http://raptor.martincarlisle.com](http://raptor.martincarlisle.com)
- **More About Computational Thinking**
  - [http://www.cs.cmu.edu/ct](http://www.cs.cmu.edu/ct)
- **Questions after SIGCSE:**
  - tcortina@cs.cmu.edu