Adopting CS Principles in a Breadth-First Survey Course

Chris Mayfield

Department of Computer Science
James Madison University

CCSC-CP 2017, Lincoln, NE
Survey Courses

Problem: how to introduce computer science to new students
  ▶ In a way that appeals to a wide audience
  ▶ And doesn’t focus solely on programming

Late 1980s: ACM task force on the Core of Computer Science (COSC)
  ▶ Proposed a three-course sequence, 42 lectures, 35 labs
  ▶ “A rigorous, challenging survey of the whole discipline.”

Today: many departments still offer survey courses
  ▶ Breadth-first introductions to computer science
  ▶ Emphasis on computer fluency, general education
7 big ideas, 44 learning objectives, 312 essential knowledge statements
Main Idea of This Paper

<table>
<thead>
<tr>
<th></th>
<th>Creativity</th>
<th>Abstraction</th>
<th>Data</th>
<th>Algorithms</th>
<th>Programming</th>
<th>Internet</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online Discussions</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>U1: Introduction</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>U2: Data Storage</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>U3: Program Execution</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>U4: Operating Systems</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>U5: Computer Networking</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>U6: Information Security</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>PT1: Explore/Impact</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>U7: Algorithms and Python</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>U8: Programming Languages</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>U9: Software Engineering</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>U10: Data Structures</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>U11: Database Systems</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>U12: Artificial Intelligence</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>PT2: Create/Program</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Framework = JMU CS 101

U12: Artificial Intelligence
PT2: Create/Program

Mayfield 2017
Adopting CS Principles in a Breadth-First Survey Course
CCSC-CP 4 of 27
Currently using Brookshear and Brylow (2015), *Computer Science: An Overview*
Some Context

About my institution

- Teaching-oriented, public university
- 19K undergrad, 2K grad, over 500 CS majors
- 18 CS faculty (all full-time, tenure track)

About the course

- CS 101, Introduction to Computer Science
- Offered as an elective, will be required soon
- 2–3 sections per year, 25–30 students each
Course Goals

- Provide our majors with a common language and broad understanding of CS that will help them put the rest of their coursework into a larger context.
- Align with the proposed AP CS Principles course that seeks to broaden participation in computing in K-12 education.
- Give non-majors a unique opportunity to learn how to think like a computer scientist, without having to take a programming-intensive course.
Assessment

- **30% Participation**: labs and exercises, online discussion posts, group activities
- **50% Quizzes**: vocab matching, multiple choice, fill in the blank, short answer
- **10% Explore Task**: collaborative research paper and poster presentation
- **10% Create Task**: pair programming assignment and individual reflection
Curriculum and Labs
Unit 1: Introduction

*Algorithms, metacognition, history of computing, seven big ideas*

**Lab:** Lightbot Hour of Code

http://lightbot.com/hocflash.html

**CSP:** Creativity 1.2.2; Abstraction 2.2.1, 2.2.2, 2.3.1; Algorithms 4.1.1, 4.2.4; Programming 5.1.2; Impact 7.1.1
Unit 2: Data Storage

Logic gates, binary, hexadecimal, RAM, hard disk, ASCII, overflow

Lab: Logisim ripple carry adder
http://www.cburch.com/logisim/

CSP: Creativity 1.2.2, 1.2.3, 1.2.5; Abstraction 2.1.1, 2.1.2, 2.2.1, 2.2.3, 2.3.1; Data 3.3.1
Unit 3: Program Execution

*How CPUs work, instructions, machine cycle, code vs data, masking*

**Lab:** Machine language simulator

http://bmachine.sourceforge.net/

**CSP:** Creativity 1.2.5; Abstraction 2.1.1, 2.1.2, 2.2.2, 2.2.3, 2.3.1, 2.3.2; Algorithms 4.1.2; Programming 5.2.1, 5.4.1, 5.5.1
Unit 4: Operating Systems

Job scheduling, multitasking, components of OS, firmware, processes

Lab: Unix commands and files

http://www.ee.surrey.ac.uk/Teaching/Unix/

CSP: Creativity 1.1.1, 1.2.2, 1.2.3, 1.2.5; Abstraction 2.2.2, 2.2.3; Data 3.1.1, 3.1.2; Programming 5.2.1, 5.3.1; Impact 7.3.1
Unit 5: Computer Networking

Protocols, routers, client/server, Internet, IP, DNS, URLs, HTML

Lab: Wireshark, mtr, web tools

https://www.wireshark.org/

CSP: Creativity 1.1.1, 1.2.1, 1.2.5, 1.3.1; Abstraction 2.1.1, 2.2.3, 2.3.2; Data 3.1.3, 3.2.1; Internet 6.1.1, 6.2.1, 6.2.2; Impact 7.1.1, 7.1.2, 7.3.1
Unit 6: Information Security

Access control, privilege levels, malware, DoS, encryption, keys

Lab: Telnet vs ssh, encryption

http://extranet.cryptomathic.com/aescalc

CSP: Creativity 1.2.2, 1.2.4; Abstraction 2.1.1, 2.1.2; Data 3.1.1, 3.1.2, 3.3.1; Algorithms 4.2.1, 4.2.2; Internet 6.2.2, 6.3.1; Impact 7.3.1, 7.4.1
Unit 7: Algorithms and Python

*Primitives, pseudocode, problem solving, decisions, loop control*

**Lab:** Intro to Python and IDLE

http://codingbat.com/python

**CSP:** Creativity 1.2.2, 1.2.4, 1.2.5; Abstraction 2.2.1, 2.2.2; Algorithms 4.1.1, 4.1.2, 4.2.4; Programming 5.1.2, 5.1.3, 5.2.1, 5.3.1, 5.4.1, 5.5.1
Unit 8: Programming Languages

Paradigms, compiler vs interpreter, variables, functions, scope

Lab: Finch robot dance party

http://www.finchrobot.com/

CSP: Creativity 1.1.1, 1.2.1, 1.2.3, 1.2.4; Abstraction 2.1.1, 2.2.1, 2.2.2, 2.2.3; Algorithms 4.1.1, 4.1.2; Programming 5.1.1, 5.1.3, 5.2.1, 5.3.1
Unit 9: Software Engineering

Software life cycle, prototyping, coupling, cohesion, UML diagrams

Lab: Static analysis, debugging

http://www.aptana.com/

CSP: Creativity 1.2.3, 1.2.5; Abstraction 2.2.1, 2.2.3, 2.3.1; Programming 5.1.2, 5.3.1, 5.4.1; Impact 7.1.2, 7.3.1
Unit 10: Data Structures

Arrays, lists, stacks, queues, trees, pointers, contiguous vs linked

Lab: Visualizing binary trees

http://pythontutor.com/

CSP: Creativity 1.2.5; Abstraction 2.1.1, 2.1.2, 2.2.3, 2.3.1; Data 3.1.3, 3.2.2; Algorithms 4.1.2, 4.2.1; Programming 5.3.1, 5.4.1, 5.5.1
Unit 11: Database Systems

*File system vs DBMS, schemas, relational model, SQL, data mining*

**Lab:** Exploring your SQLite data

http://sqlite.org/

**CSP:** Creativity 1.1.1, 1.2.2, 1.2.4; Abstraction 2.2.2; Data 3.1.1, 3.1.2, 3.2.1, 3.2.2; Algorithms 4.1.2; Programming 5.1.1, 5.1.3, 5.5.1; Impact 7.1.1, 7.2.1, 7.3.1, 7.5.1
Unit 12: Artificial Intelligence

Turing test, semantics, production systems, state graph, heuristics

Lab: Finch robot obstacle course
http://www.finchrobot.com/

CSP: Creativity 1.1.1, 1.2.2, 1.2.3, 1.2.4, 1.2.5; Abstraction 2.2.2, 2.3.2; Algorithms 4.1.1, 4.1.2, 4.2.1, 4.2.3; Programming 5.1.2, 5.1.3, 5.4.1, 5.5.1
Evaluation and Reflection
Pre/Post Survey

1. How confident are you that you are able to do the following?
   ▶ 5-point scale: apprehensive to confident
   ▶ List of 10 representative learning objectives

2. Rate your interest in the following areas of Computer Science.
   ▶ 5-point scale: uninterested to interested
   ▶ List of 12 main chapters from the textbook

3. Additional open-ended questions on the post-survey:
   ▶ How has taking CS 101 benefited you?
   ▶ What advice would you give a student taking CS 101?
Quantitative Results

Data collected from first three years of the course

- About 150 students, 18% female, 48% non-major

Confidence

- Slight increase overall (female went up, male went down)
- Not surprising given the grades/experience of students

Interest

- Slight decrease overall (about the same for male/female)
- More interested in CS as a whole, less in specific sub-areas
Qualitative Results

Retention

▶ “I’m more comfortable working with not only computers, but the software we’ve used like Wireshark, IDLE, and SQLite.” (CS major, female)
▶ “It gave me a more in-depth knowledge in the field . . . by teaching material that can actually help you in other CS classes.” (CS major, male)
▶ “I have learned that CS is not for me, but I find it very fascinating.” (Undeclared, male)

Recruitment

▶ “Taking this class has taught me a ton of valuable information. I took this class to see if I wanted to switch majors to CS, and now I do.” (Nursing major, male)
▶ “101 has increased my interest in the computer science field. I have enjoyed each chapter and definitely want to continue studying it.” (Undeclared, female)
▶ “I didn’t think I would actually enjoy CS, but this course made me realize that I actually do, especially programming.” (Philosophy major, female)
Thank You!

https://w3.cs.jmu.edu/cs101

(Google: JMU CS 101)