Microteaching: Semantics, Definition of a Computer, Running Times, Fractal Trees, Classes as Encapsulation, and P vs NP

Colleen M. Lewis
Computer Science Department
Univ. of Illinois at Urbana-Champaign
Urbana, IL, USA
ColleenL@illinois.edu

Kathi Fisler
Computer Science Department
Brown University
Providence, RI, USA
kfisler@cs.brown.edu

Jenny Hinz
Lane Technical High School
Chicago Public Schools
Chicago, IL, USA
jhinz@cps.edu

David J. Malan
Harvard University
Cambridge, MA, USA
malan@harvard.edu

Joshua E. Paley
Mathematics Department
Henry M. Gunn High School
Palo Alto, CA, USA
josh.paley@gmail.com

Manuel A. Pérez-Quiñones
Dept. of Software & Info. Systems
Univ. of North Carolina at Charlotte
Charlotte, NC, USA
Perez.Quinones@uncc.edu

Shikha Singh
Computer Science Dept.
Williams College
Williamstown, MA, USA
shikha@cs.williams.edu

ABSTRACT

SIGCSE is packed with teaching insights and inspiration. However, we get these insights and inspiration from hearing our colleagues talk about their teaching. Why not just watch them teach? This session does exactly that. Six exceptional educators will present their favorite piece of innovative lecture content just as they would to their students. The moderator, Colleen Lewis, will describe the central pedagogical move within the innovation and how this connects to education research. The goal of the session is to inspire SIGCSE attendees by highlighting innovative instruction by exceptional educators. The specific content of the innovative instruction may be applicable for some attendees, and the discussion of the underlying pedagogical move within each innovation can be applied across the attendees' teaching.

CCS CONCEPTS

• **Social and professional topics → Computing education**.

KEYWORDS

pedagogy; pedagogical content knowledge; innovating teaching

1 INTRODUCTION

The session will highlight innovative explanations and pedagogical moves (like "Nifty Assignments" but for instruction). Each of six exceptional educators will teach the audience something (7 minutes each). After this, the moderator will draw the attention of the audience to particular pedagogical moves that the instruction included (2 minutes). Resources from each of the presenters will be shared on CSTeachingTips.org. The presenters cover a range of topics, grades, and pedagogical methods. Like the past two instantiations of the session [1, 2], we expect attendees will be able to take ideas from this session directly back to their teaching.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE '21, March 13–20, 2021, Virtual Event, USA
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8062-1/21/03.
https://doi.org/10.1145/3408877.3432582

2 EXCEPTIONAL EDUCATORS

2.1 Semantics - Kathi Fisler

We present a new approach to teaching programming language (PL) courses. Its essence is to view PL learning as a natural science activity, where students probe instructor-provided languages experimentally to understand both the normal and extreme behaviors of their features. This has natural parallels to the "security mindset" of computer security, with languages taking the place of servers and other systems. The approach is modular (with minimal dependencies), incremental (it can be introduced slowly into existing classes), interoperable (it does not need to push out other, existing methods), and complementary (since it introduces a new mode of thinking). It also emphasizes the idea that languages can associate significantly different semantic behaviors with the same syntactic notations.

Kathi Fisler is a Professor (Research) at Brown University and a co-director of Bootstrap, a national K-12 outreach program that integrates introductory CS into math, physics, and social science classes. She spent many years doing software and security verification research before deciding that people were harder (and more interesting) to model than systems. Her current research area is computing education, where she studies the impact of different programming languages on how students approach problems.

2.2 Definition of a Computer - Jenny Hinz

CS is so much more than just programming, but often when a student walks into their first CS classroom without previous experience their assumption is limited to CS as just programming. The Exploring Computer Science curriculum helps expand the student's view of what CS is by breaking narrow stereotypes of the field and broadening the perspective for the students. Why is it so important to explore this topic with students and address any misconceptions? This topic in the ECS Curriculum creates a foundation for different avenues of CS, opening the access to a broader range of interests and confidence levels, creating equitable opportunities to broaden the playing field for success. To begin the examination of what CS encompasses, the ECS course has the students investigate the following questions: what is a computer and what is computation? I will be your guide through that journey ourselves as we discuss and debate those very questions.

Jenny Hinz is a CS Teacher in the Chicago Public School district. She helped implement research on block based coding in the classroom as part of a three year study with Northwestern University. Currently, she is working on developing curriculum for an iOS Application Development Course for highschoolers in partnership with Northwestern University.

2.3 Running Times - David J. Malan

Searching an array of a values is a bit like checking what's behind door number 1 (or 0) followed by door number i, which sounds a bit like Let's Make a Deal. So why not approach it as such? We present an introduction to arrays, linear search, and binary search that typically involves inviting one or two students to the front of the class to search a pair of arrays, unordered and ordered. The array itself can be implemented with pieces of paper taped to the board, behind which are numbers in chalk, or as HTML divs on a touchscreen instead, or even as actual doors. Sometimes the demo goes well, with one search taking n steps and the other $\log n$. Sometimes the students get lucky, and both take just 1! With a bit of planning or choreography, though, the intended points can be made. And quite often do students' classmates start rooting for them along the way.

David J. Malan is Gordon McKay Professor of the Practice of Computer Science at Harvard University, where he teaches CS50.

2.4 Fractal Trees - Josh Paley

Fractal trees can be utilized on Day 1 of class to teach recursion–without using the word "recursion" or the term "base case" in a manner that is accessible to high schoolers (and probably much younger kids). Also covered are exponential growth, the idea that a computer cannot solve every problem, infinite loops, creation of a procedure, and more.

Josh Paley has been a lecturer and instructor in industry as well as at the high school, community college, and university levels. He has taught AP CS AB and A, created three CS classes, and he worked on the Beauty and Joy of Computing during his nearly two decades at Gunn HS. He is proud to have received an Aspirations Award for Instructors from NCWIT. He thinks that there are aspects of recursion that are completely accessible to students ages 8 and up and aims to make that case in his presentation.

2.5 Classes as Encapsulation - Manuel A. Pérez-Quiñones

We often claim that classes encapsulate behavior. However, students find this confusing because they are both the creator and the user of the class; hiding or encapsulating details from themselves seems odd to them. Instead of presenting classes as a way to wrap a name around other variables (e.g., a circle is a class that has a point and radius), I present classes as a way to define an existing entity in the world, one that has properties, parts and actions. For example, a student's apartment on campus has color (e.g. wall paint), some parts that are required (e.g., a door, a bed) and other parts that are optional (e.g., posters on the wall, laundry on the floor), and actions (e.g., close the door, pick up the laundry).

I typically engage them in a brainstorm session to define the parts and attributes of an entity. Then we design a class classifying some of the information as attributes, parts, and actions. This directly leads to the functionality that must be at the interface of a class. For parts, there must be a way to add and remove them. For attributes, there is a set/get, etc. In my portion of the session, I will cover class constructors, fields, accessors, and mutators.

Manuel A. Pérez-Quiñones is a Professor of Software and Information Systems at the University of North Carolina at Charlotte. He is an ACM Distinguished Member and for his efforts to diversify computing has been recognized with the 2017 Richard A. Tapia award, and the 2018 CRA Nico A. Habberman award.

2.6 P vs NP - Shikha Singh

The problem P versus NP is a major unsolved problem in CS, often publicized for the million dollar prize offered for its solution. It is widely believed that P, the class of problems that can be solved quickly, is not equal to NP, the class of problems whose solution can be verified quickly. The concept of NP hardness is used to give evidence that a problem is unlikely to admit a fast solution. If a problem is NP hard, it means don't try to find a fast solution for it, you'll be wasting your time! In this lesson, I will explain the P versus NP problem and NP hardness in a way that is accessible to advanced highschoolers and CS undergraduates.

Shikha Singh is an assistant professor of CS at Williams College. Before joining Williams, Shikha spent a year at Wellesley College as an assistant professor.

ACKNOWLEDGMENTS

CSTeachingTips.org is funded by the NSF (1339404 and 1821136).

REFERENCES

[1] Colleen M Lewis, Leslie Aaronson, Eric Allatta, Zachary Dodds, Jeffrey Forbes, Kyla McMullen, and Mehran Sahami. 2018. Five Slides About: Abstraction, Arrays, Uncomputability, Networks, Digital Portfolios, and the CS Principles Explore Performance Task. In *SIGCSE Proceedings*. ACM, 269–270.

[2] Colleen M Lewis, Daniel D Garcia, Helen H Hu, Saber Khan, Nigamanth Sridhar, Bryan Twarek, and Chinma Uche. 2019. Microteaching: Recursion, Coding Style, Creative Coding, Inheritance and Polymorphism, Loops, and the Internet. In *SIGCSE Proceedings*. ACM, 962–963.