Workshop #1: Teaching Computing Foundations to Non-Majors

Catherine Bareiss, Olivet Nazarene University
Larry Vail, Olivet Nazarene University

This workshop shows participants about a new way to teach non-computing majors about computing fundamentals. This involves teaching computing concepts (such as sources of error, algorithm analysis, data storage, and simulations) that students encounter when they use computers to do work in their own disciplines. The workshop focuses on how scientists use computers in their work but will also introduce ways to expand this to other disciplines. The course that will be presented presents different modules of computer science interspersed with different science modules that apply the computing knowledge. More details can be found at http://cf4s.olivet.edu .

Workshop #2: Teaching Parallel & Distributed Computing with MPI

Joel Adams, Calvin College
Richard Brown, St Olaf College
Elizabeth Shoop, Macalester College

CS 2013 brings parallelism into the CS curricular mainstream. The Message Passing Interface (MPI) is a platform independent, industry-standard library for parallel and distributed computing (PDC). The MPI standard includes support for C, C++, and Fortran; third parties have created implementations for Python and Java. This hands-on workshop introduces MPI basics using parallel patterns, including the single program multiple data (SPMD) execution, send-receive message passing, master-worker, parallel loop, and the broadcast, reduction, scatter, gather, and barrier patterns. Participants will explore 12 short programs designed to help students understand MPI basics, plus longer programs that use MPI to solve significant problems. The intended audience is CS educators who want to learn about how message passing can be used to teach PDC. No prior experience with PDC or MPI is required; familiarity with a C-family language and the command-line are helpful but not required. The workshop includes: (i) self-paced hands-on experimentation with the working MPI programs, and (ii) a discussion of how these may be used to achieve the goals of CS2013. Participants will work on a remote Beowulf cluster accessed via SSH, and will need a laptop or a tablet with an SSH client (e.g., BitVise, iSSH), or a laptop with both a recent C/C++ compiler and MPI (e.g., OpenMPI or MPICH) installed. See http://csinparallel.org.

Workshop #3: Teaching Computer Science Soft Skills

Orit Hazzan, Technion-Israel Institute of Technology
Gadi Har-Shai, Technion-Israel Institute of Technology

This workshop addresses the teaching of computer science soft skills, such as teamwork, diversity, presentation and communication skills, and ethical behaviors. It is based on the assumption that the people involved in software development processes deserve more attention, and therefore, their soft skills should also be addressed, beyond their scientific and engineering skills. In the workshop, we will highlight such soft skills using activities that aim to analyze them from a social and cognitive perspective. We will also present and discuss a course outline dealing with soft skills of computer science, including suggestions for specific activities and tasks.

Workshop #4: SEED Labs: Using Hands-on Lab Exercises for Computer SEcurity EDucation

Wenliang Du, Syracuse University

Security courses have been integrated into many mainstream undergraduate and graduate computer science curricula. To achieve effective education, learning security principles must be grounded in experience. Over the last 12 years, we have developed 30 hands-on lab exercises for computer security education. These labs, called SEED labs (SEcurity EDucation), cover a variety of security topics, including vulnerabilities, attacks, software security, system security, network security, web security, access control, authentication, cryptography, etc. These labs are built upon a Linux virtual machine, the image of which can be downloaded from our web site. Students just need to use a single computer (can be their own laptop) to work on these labs, so there is no need for a dedicated physical laboratory. All softwares used in the labs are open-source and free.

The SEED project has been funded by 3 NSF grants with a total budget of 1.3 million dollars. So far, SEED labs have been used by over 250 institutes in 30 countries.

To help more people use these labs, we would like to hold a workshop at SIGCSE. We will select some of the most popular SEED labs, demonstrate how they work. We will then select three labs, and guide the participants to work on these labs in the workshop. Participants who want to gain experiences with more SEED labs can apply to attend our free 3-day workshop in June (all cost will be covered by our NSF grant).

Workshop #5: Teaching Introductory Computer Science For A Diverse Student Body: Girls Who Code Style

Jeff Stern, University of Michigan
Ashley Gavin, Girls Who Code
Kari Bancroft, Girls Who Code

At a time when less than 20% of AP Computer Science test-takers are female, Girls Who Code [GWC] has developed a unique teaching philosophy and effective curriculum to engage young women in CS early in their careers. This curriculum is built around cultivating exposure, interest, community and support, and technical skills. More than 500 young women have participated in GWC's Summer Immersion Programs, a seven-week, full-time computer science program for high school students. 95% of participants said they are definitely or more likely to consider a major / minor in computer science after participation.

This workshop showcases GWC's best practices so computer science educators can participate in building the pipeline of diverse talent by effectively teaching others. At the end of this workshop, participants will have concrete examples of how to effectively teach students who are underrepresented in computer science. Participants will also have the opportunity to practice teaching in the style and spirit of the GWC Summer Immersion Program, with the goal of engaging young women, and expanding this knowledge to reach minorities and other diverse populations.

Workshop #6: Making Music with Computers: Creative Programming in Python

Bill Manaris, College of Charleston

This workshop is an introduction to creative software development and music making in Python. This material is intended for CS0/CS1 courses and for courses at the intersection of computing and the arts. The workshop will introduce music making activities for teaching traditional CS1 topics, GUIs, event-driven programming, and connecting to external devices (e.g., smartphones, digital pianos) via MIDI and OSC (Open Sound Control). Participants will be introduced to Jython Music (http://jythonMusic.org), a library of Python modules for creative programming and music making, and will be making their own music artifacts a few minutes later. Software, course materials, and student activities will be provided. Laptop Required.

Workshop #7: Intellectual Property Law Basics for Computer Science Instructors

David G. Kay, University of California, Irvine

Increasingly the practice of computing involves legal issues. Patenting algorithms, domain name poaching, downloading music, and "re-using" HTML and graphics from web sites all raise questions of intellectual property (IP) law (which includes patents, copyrights, trade secrets, and trademarks). In the classroom, computer science educators often confront questions that have legal ramifications. The presenter, who is both a computer scientist and a lawyer, will introduce the basics of intellectual property law to give instructors a framework for recognizing the issues, answering students' questions, debunking the most egregious misconceptions about IP, and understanding generally how the law and computing interact. All CS educators are welcome; no computer is required.

Workshop #8: A Swift Introduction to Swift App Development

Michael Rogers, Northwest Missouri State University
Bill Siever, Western Illinois University

Swift is a new programming language recently introduced by Apple as a replacement for Objective-C. Considering that Objective-C ranks third on the Tiobe Index and is the progenitor of virtually all the 1.2 million apps in the App Store, Swift is likely to become the dominant language for creating both iOS and OS X apps.

While Swift is aimed at application developers, the language and the tools that accompany it are also a boon for CS educators. Many of the features in Swift that aid commercial development, such as type-safety, clean syntax, closures, and named parameters, are also beneficial when learning programming. In addition to the language itself, Apple has introduced a remarkable feature called Playgrounds. As the name suggests, Playgrounds allow students to “play”, that is, to interactively experiment, with code. They also provide a convenient visualization tool to graphically depict the impact of iteration, providing crucial insights for novice programmers. It is also possible to use Playgrounds to write tutorials or even entire textbooks with embedded, live code for students to experiment with.

The workshop will introduce Swift through the use of Playgrounds. Participants will work through a variety of hands-on, active learning exercises to learn Swift’s syntax and semantics. More importantly, they will experience a sample of active learning exercises that can be used in to introduce students to programming with Swift.

Workshop #9: Conducting Educational Research in the Computer Science Classroom: Choosing the Appropriate Research Design to Address Your Research Questions

Aman Yadav, Michigan State University

This workshop will provide CS educators with tools to conduct educational research. Primary objectives of this workshop are: (1) learn basic principles of research design; (2) learn about various types of research designs: qualitative vs. quantitative; experimental vs. quasi-experimental; case studies, survey; and (3) to practice designing research. This workshop will help participants make informed decisions when faced with limitations of educational research and collect empirical evidence about what works in the classroom. In addition, we will also discuss how to develop robust student outcome measures, such as surveys and tests. The workshop will be beneficial to participants who have not yet done all of these activities as well as those who have some background in educational research.

Workshop #10: Pencil Code: Bridging the Gap Between Visual and Text-based CS Education

David Bau, Google
Matthew Dawson, Google
Anthony Bau, Phillips Exeter Academy

This workshop provides an in-depth look at the learning gap between visual programming and text-based coding. We will work with the open source coding environment Pencil Code, a tool with a dual-mode editor that switches between visual block programming and text programming seamlessly. The tool allows students to work with programs with any level of complexity using either blocks or CoffeeScript text code.

Attendees will come away with an understanding of best practices when teaching students transitioning to text code for the first time, and they will gain practical experience with lessons involving functions and data. We will discuss our experiences teaching using the tool, and we will share curriculum material appropriate for middle-school and high-school classrooms. The tool is open-source and free to use online at http://pencilcode.net/.

Workshop #11: Teaching Cryptography and Access Control Hands-on

Steve Carr, Western Michigan University
Melissa Keranen, Michigan Technological University
Jean Mayo, Michigan Technological University

Cryptography and access control are perhaps the two most fundamental mechanisms for data protection. This workshop presents hands-on methods for teaching cryptography and access control that leverage software tools developed with funding from the NSF. The workshop will proceed in two sessions. The first session will address teaching well-known ciphers including the Vigenere, DES, AES, RSA, SHA ciphers and elliptic-curve cryptography using tools from the cryptoVisual software suite (VIGvisual, DESvisual, AESvisual, RSAvisual, SHAvisual, and ECvisual). The second session will address access control and the Multilevel Security, Role Based Access Control, and Domain Type Enforcement models using tools from the acVisual software suite that support graphical policy development and analysis. The presenters have used this material in undergraduate courses in Cryptography and Computer Security and the presented methods are aimed at undergraduate classes. Participants will install and use the software on their own laptops running Linux, Windows, or MacOS. The cryptoVisual

software suite is available at http://www.cs.mtu.edu/~shene/NSF-4/. The acVisual software suite is available at http://acv.cs.mtu.edu.

Workshop #12: Bridging the Divide: Developing Culturally-Responsive Curriculum for K-12 Computer Science Education

Alicia Washington, Howard University

In order to increase the computer science pipeline, emphasis must be placed on not only who is taught, but also how they are taught. Traditional computer science pedagogy has been unsuccessful in attracting, engaging, instructing, and retaining underrepresented students. Culturally-responsive pedagogy must be leveraged to successfully instruct a diverse range of computer science students.

With the onset of technologies such as Facebook, iPods, Xbox, smart phones, mobile applications, and more, underrepresented students are already actively engaged in utilizing computer technology. However, in order to transition them from consumers to creators of this technology, culturally-responsive curriculum must teach fundamental concepts such as algorithms, problem solving, and abstraction in the context of issues that affect their daily lives.

This workshop is designed to help K-16 computer science educators, professionals, and others understand and identify activities and assignments that infuse culturally-responsive content throughout.

Participants will first engage in a discussion on culturally-responsive pedagogy, why it is necessary in computer science, and how computer science is a part of their individual daily lives. Participants will then work in groups to design and present a culturally-responsive interactive lesson, extended project, and homework assignment. Each group will be provided different school and class demographics for a diverse range of scenarios and content.

By the end of the workshop, participants will have a clear understanding of culturally-responsive pedagogy, its importance in computer science, and how to begin infusing more of it into lessons and activities. Laptops are recommended.

Workshop #13: On Beyond Sudoku: Pencil Puzzles for Introductory Computer Science

Zack Butler, Rochester Institute of Technology
Ivona Bezakova, Rochester Institute of Technology

Problem solving is a powerful teaching methodology for computer science - giving students a real problem to solve instead of simply discussing abstract concepts can motivate them and give them a path to better understanding. However, it is challenging to create novel example problems that are meaningful and engaging yet can be easily understood by all students. In this workshop, we will introduce participants to the vibrant world of pencil puzzles and show how many different types of puzzles can be used throughout the introductory CS curriculum. Pencil puzzles are those designed to be solved by hand with pencil and paper (such as Sudoku, but including of dozens of new types!) which have clear rules and are made to be solved deductively. As such, they are explicitly designed to be easy to understand and intriguing and naturally inspire algorithmic thought in the solver. We will explore a variety of on-line resources, including our own curated repository, to see how assignments throughout the introductory CS curriculum can be easily kept fresh. Participants will also experience a sample problem-solving session and collaboratively develop a new assignment. This workshop is intended for all teachers (late secondary and post-secondary) of introductory programming courses. Laptops are recommended.

Workshop #14: How to Plan and Run Summer Computing Camps - Logistics

Marguerite Doman, Winthrop University
Barbara Ericson , Georgia Institute of Technology
Kristine Nagel, Georgia Gwinnett College
Nannette Napier, Georgia Gwinnett College
KrishnenduRoy, Valdosta State University

This workshop will provide details on how to plan and run non-residential computing summer camps for 4th-12th grade students. Georgia Tech has been offering computing summer camps since 2004. These camps are financially self-sustaining and effective. Items used in the camps include: CS Unplugged, LightBot, Scratch, Alice, LEGO robots (WeDo, NXT, EV3, and Tetrix), EarSketch, and App Inventor. Georgia Tech helped start other computing camps at eleven other colleges and universities in Georgia from 2007 to 2010 as part of Georgia Computes!. This last year as part of the Expanding Computing Education Pathways (ECEP) NSF grant we have also helped institutions start or expand summer computing camps in South Carolina, Massachusetts, and California.

The workshop will include forms, a timeline, sample agendas, sample flyers, budget plans, a planning checklist, suggested projects, surveys, pre and post-tests, evaluation results, lessons learned, and more.

Workshop #15: Small or Liberal Arts Colleges Adapting to CS2013: Making It Work

Andrea Danyluk, Williams College
Michael Jipping, Hope College
Rhys Price Jones, Wellesley College
David Reed, Creighton University
Brad Richards, University of Puget Sound
Richard Wicentowski, Swarthmore College

Roughly once per decade, the ACM and IEEE-Computer Society form a joint task force to produce curricular guidelines for undergraduate computer science programs. The latest guidelines document, Computer Science Curricula 2013 (CS2013), was released in December 2013. CS faculty at many institutions are interested in understanding CS2013, evaluating their curricula against it, and adopting some or all of the recommendations. This task is non-trivial at any institution, but it can be particularly challenging at small or liberal arts institutions. For instance, small schools with few faculty are limited in the number of courses they can offer, and many liberal arts colleges place limits on the number of courses that can be required for a major. How might a department with limited resources trade off covering the core while still providing electives to students? A meta-issue involves mapping CS2013 at all: Given the demands on a small department, are there ways to improve and update a curriculum without having to go through a complete CS2013 mapping? In this workshop we will: (1) give a brief overview of CS2013, (2) describe experiences mapping individual courses or an entire curriculum to CS2013, (3) split participants into working groups for course or curricular mapping based on starting points and goals, (4) re-group to share lessons learned from the mapping experience. Laptop Strongly Recommended.

Workshop #16: Steal This Courseware: FOSS, Github, Flask, and OpenShift

Stephen Jacobs, RIT Center for Media, Arts, Games, Interaction and Creativity (MAGIC)
Remy DeCausemaker, RIT Center for Media, Arts, Games, Interaction and Creativity (MAGIC)

This workshop introduces participants to the pedagogy and practice of using Free/Open Source Software development practices into their curriculum, and then guides them through deployment of a turnkey courseware framework to be used for their own courses. The framework supports automatic blog checking, automatically generated student profile pages, Gravatar integration for profile pictures, Travis-CI continuous Integration tests, and repository changes reported via Github webhooks to IRC. Participants will learn how to use Github in the Classroom, the basics of Flask, a python web framework, and how to deploy their courseware to Red Hat's OpenShift Cloud, a free Platform-as-a-Service to host courseware and/or other web sites.

Workshop #17: The Internet, Creativity and Global Impact: Curriculum Modules for the new AP Computer Science Principles Course

Lien Diaz, College Board, Advanced Placement Program
Richard Kick, Newbury Park High School
Andrew Kuemmel, Madison West High School

This workshop focuses on content in two curriculum modules for AP Computer Science Principles (CSP) developed by the College Board. They highlight instruction for concepts about 1) the Internet and 2) the interplay between creative aspects of computing and impact of computing on society.

Workshop #18: Games, Phones, and Programming in the Classroom

Veronica Catete, NC State University
Barry Peddycord III, NC State University
Tiffany Barnes, NC State University

The new CS Principles curriculum, a pilot Advanced Placement course, offers novice students an exciting opportunity to learn computing in a hands-on, fun way. High school and college teachers of introductory computer science course are invited to this workshop to learn basic game and mobile phone development. Participants will learn GameMaker, AppInventor, and TouchDevelop. These tools allow students to create and have fun with computing while teaching object-oriented and event-driven programming and game architectures. Participants should bring their own laptops. We will provide links to curricular modules for the CS Principles: Beauty and Joy of Computing course.

Workshop #19: Infusing Cooperative Learning into Early Computer Science Courses to Support Improved Engagement

Jeff Gray, University of Alabama
Fran Trees, Rutgers University
Owen Astrachan, Duke University

Many new curricula and tools have been developed recently to promote the exciting opportunities available in computer science. However, curriculum and supporting tools alone do not drive engagement - the most interesting and innovative curriculum can still be taught in a disengaged manner, leading to lost opportunities for broadening the appeal and interest in computing across a diverse student population. The learning science literature on Cooperative Learning (CL) has been shown to increase class participation and student learning, while also promoting diversity in a manner that supports the differentiated instruction needed to engage students who have mixed abilities. This workshop will demonstrate how the best practices of CL can be applied in early CS courses (e.g., CS

Principles AP CS A, or CS1). Workshop participants will first be introduced to the general CL structures that have been used in many different disciplines across multiple age/grade levels. These structures will then be used to demonstrate specific application toward computer science concepts. The workshop itself will be taught in a cooperative learning style so that participants can understand the dynamics and structure of a CL classroom.

Workshop #20: Computer Science Principles with EarSketch

Jason Freeman, Georgia Institute of Technology
Brian Magerko, Georgia Institute of Technology

EarSketch (http://earsketch.gatech.edu) is an integrated curriculum, software toolset, audio loop library, and social sharing site that teaches computing principles through digital music composition and remixing. Attendees will learn to code in Python and/or JavaScript to place audio clips, create rhythms, and add and control effects within a multi-track digital audio workstation (DAW) environment while learning computing concepts such as variables, iteration, conditionals, strings, lists, functions, and recursion. Participants write code to make music, with a focus on popular genres such as hip hop. The agenda outlines the pedagogy of connecting musical expression to computation to broaden participation and engagement in computing; the underlying concept of thickly authentic STEAM that drives this approach; the alignment of the curriculum and learning environment with CS Principles; and basic musical concepts underlying EarSketch. The intended audience for this workshop is secondary and early post secondary CS educators. The course is of particular relevance to CS Principles teachers but also applicable to any introductory programming or computing course. No prior musical knowledge or experience is expected and no prior programming experience with Python or JavaScript is required.

Workshop #21: Teaching Computing with Processing, the Bridge between High School and College

Aaron Cadle, Arlington ISD
Ira Greenberg, Southern Methodist University
Deepak Kumar, Bryn Mawr College
Darby Thompson, Sidwell Friends School
Ursula Wolz, RiverSound Solutions
Dianna Xu, Bryn Mawr College

This workshop showcases an engaging way to attract students who typically avoid a traditional introductory Computer Science course (CS1), with fully developed, classroom-tested course materials. This workshop has been successful at SIGCSE and other venues in the past. This year we highlight our successful approach in pre-AP courses, as well as continued refinement of curriculum for college-level CS1. Our courses focus on essential CS1 principles, but show applications of these principles with contemporary, diverse examples of computing in a modern context, including advanced areas typically not accessible in CS1 such as: physics-based simulations, fractals and L-systems, image processing, emergent systems, cellular automata and data visualization. Students produce dynamic visual work using the programming language Processing, which is fully compatible with Java. We aim to inspire the Computer Science community to use innovative and creative approaches to attract a broader audience to their classes.

Participants will be introduced to the Processing language as well as its lightweight IDE through a series of on-the-fly coding examples. Additionally, course materials and handouts detailing the software, curricula and teaching resources will be given to the participants. Instructors of all levels are welcome; high school computer science teachers are particularly encouraged to attend. All participants will need to bring their own laptops.

Workshop #22: Supporting new Adopters to Peer Instruction in Computing

Daniel Zingaro, University of Toronto Mississauga
Leo Porter, University of California, San Diego
Quintin Cutts, University of Glasgow
John Glick, University of San Diego
Joe Hummel, University of Illinois at Chicago
Cynthia Lee, Stanford University
Jaime Spacco, Knox College

Recent work in computing has converged on a collection of complementary findings suggesting the value of the Peer Instruction (PI) pedagogy. Compared to lecture, PI has been shown to decrease fail rates, increase final exam grades, and increase engagement and enjoyment. In PI, students work together to exchange perspectives and use clickers to answer challenging conceptual questions in the presence of a knowledgeable instructor.

In our efforts to mentor potential PI adopters, we note difficulties bootstrapping PI uptake at new institutions and new departments. In this workshop, our main goal is to support potential adopters in the process of shifting from lectures to PI. In recent months, we have contributed materials for many freely-available PI courses, and led a successful 3-day NSF-funded PI workshop. We will work with participants and their existing lecture-based resources to begin considering the ways that PI might be effective in new contexts.

Instructors interested in increasing engagement in any CS course may attend. Participants are encouraged to bring current lecture materials. Laptop optional.

Workshop #23: Reviewing NSF Proposals: Learn about Effective Proposal Writing via the Review Process

Paul Tymann, National Science Foundation
Michael Erlinger, National Science Foundation

This workshop focuses on the NSF proposal review process. Via close examination of the review process, participants gain an understanding of how to write good reviews and how to improve their own proposal writing. The workshop covers the following topic areas: the proposal review process from submission of a proposal to award or decline; elements of a good review; NSF merit criteria (intellectual merit and broader impacts); elements of a good proposal; and how to volunteer to review proposals.

The workshop uses a structured guided-interactive methodology to lead participants through each topic by introducing related issues, engaging participants in group exercises designed to explore and share their understanding of the issues, and then providing some “expert” opinion on these issues. Good and bad examples and a Top Ten List of Do’s and Don’ts will be provided.

Workshop #24: Creating Stimulating, Relevant, and Manageable Introductory Computer Science Projects that Utilize Real-Time, Large, Web-Based Datasets

Eli Tilevich, Dept. of Computer Science, Virginia Tech
Clifford Shaffer, Dept. of Computer Science, Virginia Tech
Austin Cory Bart, Dept. of Computer Science, Virginia Tech

This workshop introduces participants to CORGIS, a technology developed under the auspices of an NSF-funded project at Virginia Tech. The CORGIS Datasets Project comprises a software architecture framework and carefully engineered client libraries through which students can access either large datasets or those generated by real-time web services from domains, including weather reports, stocks,

earthquakes, and news updates. The CORGIS technical scaffolding gradually introduces students to some of the most vexing complexities of distributed computing. To support the diverse needs of computing educators when teaching introductory CS classes, each CORGIS dataset is available in Python, Java, and Racket, with compatibility on key platforms. The dataset libraries are available through an online curated gallery, designed to be easily adapted to instructors' specific academic needs, including the ability to rapidly prototype new CORGIS libraries. With CORGIS, computing educators can introduce important big data or real-time distributed computing concepts without overwhelming students with the low-level details that working with such data typically requires.

This workshop introduces CORGIS via a hands-on approach, familiarizing participants with the core functionality of our architectural framework and client libraries. We will proceed in three parts: (1) present CORGIS by working through a case study of creating a programming project in a typical CS 2 course; (2) demonstrate how the framework can be used to rapidly prototype a new library of the participants' choice; and (3) critically discuss the technology in small and large groups. This presentation improves on our offering from SIGCSE 2014.

Further information is at: http://think.cs.vt.edu/corgis.

Workshop #25: Building Code Magnet Labs for Tablets and Other Devices

Barry Kurtz, Appalachian State University
Rahman Tashakkori, Appalachian State University
Ahmad Esmaili, Stony Brook University

Code magnet labs are 5-10 minute activities where students are asked to construct a method, function or rule to complete a specific task. The programming languages Java, C/C++, Python and Prolog are supported. In a code magnet lab the student is presented with a sequence of possible code magnets that can be arranged using drag-and-drop to create the desired method. Magnets for control statements can nest other magnets, including nested control statements. The completed method is compiled and subject to unit testing. Test results guide students towards a correct solution; multiple submissions are allowed. Since there is no keyboard entry when using code magnet labs, the labs can be completed using laptops, desktops, tablets and other mobile devices. This workshop is a shorter version of the half-day workshop presented with support from three NSF grants. Participants will experience using existing labs and learn to build their own code magnet lab for Java, Python, C or Prolog (participant's choice). Significant stipends averaging about $1000 will be available for after-workshop activities where the participant develops and tests multiple code magnet labs for his/her own courses in the 2014-2015 academic year.

Workshop #26: Introducing Secure Coding in CS0, CS1, and CS2

Siddharth Kaza, Towson University
Blair Taylor, Towson University
Elizabeth Hawthorne, Towson University

The CS 2013 curriculum includes Information Assurance and Security as a pervasive knowledge area. However, introducing security in lower level courses is challenging because of lack of appropriate teaching resources and training. This workshop will provide a well-tested strategy for introducing secure coding concepts in CS0, CS1, and CS2. We will introduce attendees to secure coding through hands-on exercises, and provide self-contained, lab-based modules designed to be injected into CS0-CS2 with minimal impact on the course (www.towson.edu/securityinjections). Participants will be encouraged to bring in their own syllabus and labs to modify to include secure coding concepts. The first 15 participants will be reimbursed for the workshop cost on attendance. Laptop recommended.

Workshop #27: Puzzle-Based Learning: Introducing Creative Thinking and Problem Solving for Computer Science and Engineering

Raja Sooriamurthi, Carnegie Mellon University
Nickolas Falkner, University of Adelaide
Ed Meyer, Baldwin Wallace University

Puzzle-based learning (PBL) is an emerging model of teaching critical thinking and problem solving used in Universities and schools. Today's market place needs skilled graduates capable of solving real problems of innovation in a changing environment. A learning goal of PBL is to distill domain independent transferable heuristics for tackling problems. While solving puzzles is innately fun, companies also use puzzles to assess the creative problem solving skills of potential employees. In this interactive workshop we will examine a range of puzzles and games. What general problem solving strategies can we learn from the way we solve these examples? Participants will emerge with the needed pedagogical foundation to offer a full course on PBL or to include it as part of another course.

Workshop #28: KELP CS and LaPlaya: A Computational Thinking Curriculum and Development Environment for 4th-6th Grade

Diana Franklin, UC Santa Barbara
Charlotte Hill, UC Santa Barbara
Hilary Dwyer, UC Santa Barbara

This workshop introduces our elementary school programming curriculum, KELP-CS, and the corresponding programming environment LaPlaya. KELP-CS (Kids Engaged in Learning Programming) is an innovative, modular computational thinking curriculum for 4th-6th grade students. Off-computer activities connect computer science concepts to students' every day experiences. On-computer activities in LaPlaya develop students' computational thinking and programming skills. Finally, an engineering design project allows students to apply these new skills through an open-ended, creative project (e.g. digital storytelling in Module 1, and virtual game in Module 2). LaPlaya is a modified Scratch programming environment tailored to the developmental needs of 4th-6th grade students. In addition, LaPlaya provides multiple data collection options for researchers such as logging, automatic analysis, and assessment questions.

In this workshop, we begin by introducing the KELP-CS curriculum; LaPlaya, the development environment used in the on-computer activities; and the resources for teachers embedded in both. Workshop participants will do sample on- and off- computer activities from Module 1 (4th grade) and discuss tips and strategies for teaching computational thinking with this age group.

For the second part of the workshop, we overview additional background for participants interested in a behind-the-scenes view of the LaPlaya development and structure. We demonstrate LaPlaya's logging system, optional Javascript analysis for projects, and platform for assessment questions.

Workshop #29: Teaching Privacy: What Every Student Needs to Know

Gerald Friedland, International Computer Science Institute
Serge Egelman, International Computer Science Institute
Daniel Garcia, UC Berkeley

Although frequent stories in the popular media have raised awareness about online privacy, most young people do not have a very good handle on what the specific issues are, nor the practical steps they can take to manage them. Teachers recognize that all their students -- from future engineers to those totally bored by science -- need a realistic understanding of how online privacy works, so they can protect themselves online. In fact, the latest CS curricular recommendations include privacy but there is

no comprehensive set of field-tested teaching materials. To address this, we are developing TROPE (Teacher's Resources for Online Privacy Education), a set of classroom-ready teaching materials (teachingprivacy.org). TROPE will provide educators with lesson modules, interactive demonstrations, and a teachers' guide, so they can readily integrate privacy into high school and college classes. Our goal for this workshop is twofold. First, we will introduce educators to TROPE and provide guidance on how they can cover privacy-related topics in their classrooms without being subject-matter experts. Second, we will solicit feedback and on-the-ground stories; by gaining a better understanding of specific problems faced by educators and students, we can increase TROPE's utility to teachers. We will provide teachers with up-to-date technical information about online privacy, including relevant highlights from our research; hands-on activities illustrating principles of online privacy; and an overview of the materials we are creating for TROPE. This will be an interactive workshop, driven by participants' questions, experiences, and interests. For CS educators at all levels.

Workshop #30: Decoding CS Principles - A Curriculum from Code.org

Baker Franke, Code.org
Brook Osborne, Code.org

Code.org is developing a rich set of instructional resources designed for high school teachers to meet the objectives of the (AP) Computer Science Principles course framework. This workshop will provide a sneak-peek to teachers of the head-to-tail curriculum package, which includes daily lesson plans and instructional guides for an entire school year, along with new software and tools for students and teachers. The session will also give participants a taste of the professional development program designed to support instruction.

Code.org's CS Principles curriculum is an inquiry-based course designed to be rigorous but accessible for both high school students and instructors, allowing adequate time to explore and learn the principles of computing through a series of engaging activities, plugged and unplugged, employing a variety of instructional strategies. The course places particular emphasis on the Internet and innovation and student preparation for the AP Performance Tasks.

Participants will experience the curriculum's lessons, both as students and teachers, in the fashion of the professional development program which helps teachers prepare engaging lessons from the curriculum as they would in a realistic context (i.e. small amount of prep time, probably the day the lesson needs to be given). Participants will also have the opportunity to hear from high school teachers who have taught the lessons in their classrooms during the 2014-15 school year. Code.org's partner schools will pilot the course in the 2015-16 school year, but the course and its materials will be open and free for all to use.