ABSTRACT
Programming is one of the most fundamental and central topics in the information technology curriculum. Because of its importance it is crucial to understand how to effectively teach development students. In this panel we share best practices for teaching programming to a variety of populations, including freshman, non-majors, and community college students. Various pedagogical approaches including pair programming, studio-based instruction, peer instruction, active learning, cooperative learning, project-based pedagogy, high-impact education practices, and CS Unplugged type activities are included.

Categories and Subject Descriptors
K.3.2 [Computers and Education]: Computer and Information Science Education

General Terms
Human Factors, Languages.

Keywords
Best practices, programming, development, CS1, high-impact practices, project-based pedagogy, CS Unplugged.

1. OVERVIEW
One of the most fundamental topics in the computing curriculum is programming, and it is often the case that the first experience students have with computing is a programming course. The information technology discipline holds programming as fundamental, making it one of the pillars of the model curriculum [6]. An article discussing the development of the curriculum notes that “as with any computing professional, the IT graduate must develop the skill to program” [4, p. 354].

Because of its importance to the information technology curriculum, understanding the best ways to teach programming is crucial for faculty. In this panel we discuss best practices for teaching students to program, including techniques that are most effective in a first programming course, practical ways to implement high-impact educational practices in the context of programming classes, teaching non-majors cutting edge technology, and non-traditional activities for community college students. The discussion will include relevant pedagogies such as pair programming, studio-based instruction, peer instruction, active learning, cooperative learning, project-based pedagogy, high-impact education practices, and CS Unplugged type activities.

2. AMBER SETTLE: TEACHING CS1
A programming course is often the first experience that students have with computing, and as such, it is one of the most widely studied courses in the curriculum. In the past decade a variety of novel approaches to the CS1 classroom have been developed and studied, including pair programming, peer instruction, student-generated content, studio-based instruction, active learning, and cooperative learning. Each provides a way to approach the CS1 course in a novel way, by restructuring student interaction and reframing the interaction between faculty and students.

While they all differ in their details and the ways in which they have shown to be effective, these approaches share some commonalities that illustrate best practices for the CS1 classroom, including: make the classroom dynamic, increase student-to-student interaction, encourage reflection, reduce isolation, and encourage a cooperative classroom environment. The connections between various novel CS1 pedagogies will be discussed to provide instructors with best practices that can be used in conjunction with or independent of other teaching techniques.

3. DEBORAH LABELLE: HIGH-IMPACT PRACTICES FOR PROGRAMMING
High-impact educational practices (HIPs) applied to a liberal education have shown to increase student success in all disciplines [7]. The IT student may take part in high-impact practices such as capstone projects and internships during junior
and senior year [6]. Therefore, these practices are more likely experienced as summative rather than formative activities. Research suggests that students engage in multiple HIPs in lower-level courses for optimal success [1].

Opportunities for programming students to engage in HIPs such as service learning, community-based learning and collaborative learning projects are hidden in plain view on a typical college campus. As I walk around my small campus I look for development projects. There’s a green house that needs a plants database with a web interface, there’s a student taking volleyball stats on paper, and there’s the sociology students using a 21-page paper survey to gather information about homelessness. How does the geology teacher keep track of his baby-jar collection of sand?

I work out the project expectations with my colleagues in these other disciplines, to try to ensure that the projects will have as much benefit for the recipients as for the students working on them [3]. The projects are designed to start in the classroom, and run well beyond the semester. My students volunteer to continue working on the projects on their own time. They become committed to the user and along the way they develop better programming skills and gain an understanding of how to use their skills to help others.

4. HAZEM SAID: PROJECT-BASED PEDAGOGY FOR NON-MAJORS
Cutting-edge courses such as those teaching iPhone development are particularly good attractors for non-majors. Teaching those populations development techniques can be challenging, but using project-based pedagogy can be helpful. Project-based methodology keeps the students engaged and focused on problem solving and enable students to relate development to their field of study.

In this context students are asked to develop four applications at an institution on the quarter system. Students spend two to three weeks on each application, and learning is done within the class time. Weekly journals are used to offer opportunities for the students to reflect on the challenges they are facing and offer the instructor an opportunity to provide context to the activities of the previous week and to adjust the pace, contents and challenges as the class proceeds. A group project is used for the final assignment. Students work in groups where they select an idea and work on developing the application for that idea. Class time is used for group work with the instructor consulting as necessary. As an important piece of the course, students are offered opportunities for extra credit or resubmitting the projects so they can focus on understanding the technology without a negative impact on their grade.

5. SHEILA S. SICILIA: CS UNPLUGGED IN THE COMMUNITY COLLEGE
Community colleges have a very diverse student body, in terms of age, past experience, and college preparedness. Many of our students have not had good academic experiences in the past and are potentially “at-risk” for failure in college. Various studies have suggested that a factor for many of these students is a mismatch between the students’ learning styles and traditional teaching methods [8]. Many of our students are “concrete-active” “tactual” or “kinesthetic” learners [5]. The study of computer programming, however, involves understanding logical processes that are not easy to see or touch.

Community college students respond positively to activities that model the processes in a way that is more tactual or kinesthetic. For instance, an activity called “The Living CPU” can be used to explain how the CPU processes machine code. We walk through running a short program at the binary opcode level, by acting out the parts of a simplified CPU, with a simplified instruction set. Students are assigned roles, such as ALU, Control Unit, Program Counter, Memory manager, etc. I draw grids for the memory on the whiteboard, as well as areas for the program counter and registers. The “program” that we execute is about 24 bytes long. These activities can be seen as a type of CS Unplugged activity for a community-college population [2].

6. REFERENCES