Teaching Statement

Marc Eaddy
155 E 49th Street #6B
New York, NY 10017

+1 (212) 593-3583
me133@columbia.edu
www.columbia.edu/~me133

Teaching Philosophy

For me, the goals of teaching and research are complimentary: *to further knowledge*. I find it extremely rewarding when a student finally comprehends a difficult concept. My teaching philosophy is organized around the following principles:

Learn By Doing. Students are often surprised at how difficult it is to *apply* concepts after seeming to understand them in class. I believe *passive learning* techniques (e.g., reading, lectures) are ineffective at helping students acquire knowledge. To understand complex concepts, students should engage in *active learning* (e.g., interaction, homework, programming exercises) and solve concrete problems (e.g., "Why doesn't my program compile?").

Challenge Students. When the subject matter is too simple, students quickly become bored and mentally disengage. I believe students enjoy being challenged. When a student receives a perfect score on an exam or homework, they have not learned the limits of their knowledge, which is one of the goals of higher learning. Most important, the increased mental activity helps them learn and retain the material. I prefer to challenge students by increasing the *quality* (e.g., covering advanced concepts) of the material, not the *quantity* (e.g., doubling the workload). I continually poll the class to ensure that the material is not too difficult or too easy.

Presentation Is Important. I am often complimented on my ability to patiently distill complex concepts. My approach is to use terms and examples that the class can relate to, and to be enthusiastic about the subject matter. I employ many techniques to keep the class interested including personal anecdotes, pictures, algorithm visualizations, quotes, and jokes. I also frequently ask the class questions, which is a great way to get their attention, encourage them to think, and gauge their comprehension of the material.

Teaching Experience

I taught "Programming Languages (C)" at Columbia for three semesters—a privilege not often extended to PhD students at Columbia. It was a great experience! I consider myself a dedicated teacher. I developed the entire curriculum including syllabus, lectures, slides, assignments, exams, online bulletin board, and class website (http://www.columbia.edu/~me133/coms3101-01.htm). I wrote software to automatically test and grade the assignments, and email the results. However, I always manually reviewed the submissions and included personalized messages to explain the results. I regularly extended my office hours to provide additional help to students.

I believe teachers must continually self assess and hone their craft. I setup an anonymous feedback form and used the students' comments to adjust my presentations and assignments. I also requested a teaching consultant to attend one of my classes. To hone my teaching skills, I attended a day-long teaching workshop and took a course on Computer Science Education. My efforts paid off: My course evaluations were high and improved with each semester.

I was a teaching assistant for the graduate-level Network Storage and User Interfaces classes. For the latter, I was a guest lecturer and also designed a class project assignment that was very interesting (design an MP3 player) without being too difficult. I have also guest lectured for Software Verification Tools. In addition, I have greatly enjoyed mentoring three graduate students and three undergraduates on research projects spanning multiple semesters. One of the undergraduates was recently selected as a CRA Undergraduate Award finalist, among only sixteen students in the country, based in part on the work he did for me.

Teaching Expertise

I am comfortable teaching courses on programming in Java, C++, C, and C#, software engineering, data structures, advanced programming, programming languages, compilers, and the foundations of computer science. With a little preparation, I would be comfortable teaching operating systems and arbitrary introductory computer science courses.

My industrial and research experience provide me with the expertise to teach advanced and rarely offered courses on aspect-, component-, and feature-oriented programming languages, program analysis, and program transformation.