

# Teaching Statement

Gary Wassermann  
wassermg@cs.ucdavis.edu

I love to teach because I love to learn. When I understand something I had not understood before, the first thing I am eager to do is to explain it to someone else or discuss it with a fellow student. Those of my teachers who have taught with particular clarity, rigor, creativity, and interest have inspired me to communicate material to my students with enthusiasm that inspires interest of their own.

I aim to teach in such a way that each person is glad to have come. In preparing for each class, I try to put myself in the students' place in order to truly communicate and avoid just going through the motions. First, I strive for clarity. I find that people generally do not learn conclusions without learning the steps that got there, and they generally do not learn the steps without a clear set-up and a high-level intuition about what's going on. To help provide an intuition, I use analogies and address common points of confusion explicitly. While I desire ultimately to teach students new ways of thinking, my primary vehicle for accomplishing this is teaching the material at hand clearly. When I guest lectured on logic programming, I taught Prolog, brought in my laptop, and had the students "help" me write a few programs.

Second, I strive to bring energy. Students will rarely be more lively than the teacher is or expects them to be. I move around the room. I learn the students' names. I call on them, and I find that when I use their names, they gladly make their best effort. I particularly found this to be the case when I TA'd for *Intro to Programming and Problem Solving*, our introductory CS course. Several students who had not declared a CS major, and particularly some of the women, began the course expecting not to get it. When I learned their names and proved approachable in office hours, they gained a confidence toward the material, and one student described me as "infinitely patient."

Many teach because they find it gratifying to see students grasp new concepts. I enjoy that too, but the best part of teaching for me is understanding the material better myself. I consider that I understand something when it seems simple to me, and only then am I ready to teach it and make it seem simple to others. I had this experience when TAing for both *Computer Security* and *Software Engineering*, and I counted it a success when I heard one student helping another by using the same approach I used to explain the Chinese Wall policy.

TAing both of these classes helped in my research and my research helped in teaching. Not only did I enjoy the classes more because of this, but it helped in two of my teaching goals. First, I want to convince students that it is useful to understand foundational principles of Computer Science, including more the theoretical ones. Second, I want to impart and interrelate skills at multiple levels, such as formal logic, algorithms, system implementation, and empirical studies; I envision pursuing this goal especially in the context of advising, but I bring it to the classroom as well. I perked the students' interest by relating the 30-year old material we were covering to my current work on web application security.

Because I enjoy teaching, I have asked for and been granted the opportunity to teach the Spring 2008 offering of *Programming Languages*. This is a large, required, upper division course that introduces the fundamentals of modern programming languages and presents the strengths of various languages and language families. I will have the opportunity to implement my teaching goals as I lecture, design the assignments, write the exams, and manage the teaching assistants.

In the future, I would like to teach such courses as *Programming Languages*, *Compilers*, *Software Engineering*, and *Computer Security*. In all of these courses, I will both teach the development of the foundational ideas and have the students learn by doing through implementation and in-class exercises. I would also be interested in *Introduction to Programming*, *Data Structures*, and *Theory of Computation*.