

TEACHING STATEMENT — Ross Tate

Due to a reading disability, learning from books and notes is a slow and painful process for me. Thus, I owe the multitude of concepts I learned in college to the extensive effort made by the educators in my past to make their lectures engaging and informative and their assignments efficient and effective. Without their efforts I would not be where I am today. They inspired me to seek out opportunities to teach and advise students to help them reach their own goals.

TEACHING — CSE-130: Undergraduate Programming Languages

Teaching is an opportunity that comes with many responsibilities. Not only does one need to teach the core principles at hand, but one also needs to enforce broader skills while furthermore keeping students actively engaged in the material so that they challenge themselves and retain knowledge better. One way I work to address these goals simultaneously is by focusing on drawing connections. Not only is making connections an important skill that enables knowledge to be applied more broadly than its original context, but it also makes the material interesting even to students not necessarily pursuing the primary focus of the class. Furthermore, relating unfamiliar concepts to a familiar setting enables students to draw from that familiar setting to get a better grasp of the core principles and their importance. So while a teacher has many responsibilities, with careful design one can make these responsibilities cooperate towards a well-rounded and effective class.

When I taught our undergraduate programming-languages class, in addition to covering the core principles I wanted to show how they relate to other programming experiences. I frequently connected concepts from the languages I taught to Java, a language my students already knew. For example, relating ML closures to anonymous inner classes, Prolog unification to updateable references, and CLIPS triggers to SWING events gave students a concrete intuition for what each concept is and how it works. I showed how algorithms from these languages could help them improve their Java programs, such as how Hindley-Milner type inference can be approximately adapted to Java programs in order to make methods and classes more generic. In fact, my first project for the class was in Java so that, as they implemented that same project in each language, they could experience the strengths and weaknesses of each language. Furthermore, they could recognize how knowing these languages could have helped them write that first program even though it was in Java, learning a larger lesson that exposure to new ideas can have unexpected benefits.

By focusing on connections, I hoped my students would be able to apply the class material in ways neither I nor they could predict. Indeed, I was always surprised to find unexpected applications of class material in both my industry and research experience, which always drove me to learn more interesting concepts. Less than a month after class had ended, one of my students contacted me to thank me for their own such experience. He had started a research internship in artificial intelligence, and to his surprise agent programming was just like the event-based programming language I had added to the class. Because he saw that the research built off the concepts I had already taught him, he could start researching immediately. He was so excited, and it was rewarding to hear that I enabled that.

Each class is a learning opportunity for the teacher as well. For my class, I collected feedback frequently to see how I could improve the class even as it was progressing. For example, through this feedback I found out before the deadline that my opening Java assignment was too difficult for many of my students because they had a different set of programming skills than I had expected. In particular, many of my students were not able to break down problems, exhibited concretely by attempts to implement the entire assignment using only one function. This posed a problem for me. I wanted my students to learn this fundamental skill, but I did not want their lack of practice to hinder them from the bigger purposes of the assignments. I adapted to the situation with two phases. First, I temporarily compromised by breaking down my assignments into smaller parts but meanwhile still having them figure out how to put those pieces together, personally training them when additional help was necessary. Second, I progressively made the pieces larger over time, and when working directly with students I would emphasize first investigating ways to break the problem at hand into parts, guiding them through the steps so that they could personally experience how this is done. By the end of the quarter, I was very proud of my students, not just for learning the material, but also because I could tell they had developed this problem-solving skill and become better programmers for it.

ADVISING — Mentoring, Educating, and Critiquing

I was one of the founding students in my research group. This means that already in my second year I was a senior student of our group. Right away I was mentoring incoming students, passing on the lessons I had needed to learn on my own and save them from the mistakes I had made. Because of my mathematics education, programming internships, and constant search for new connections, students would look to me for a broader understanding of the problem at hand, often to find solutions to their problems or applications of their solutions. I even gave a seminar for professors and graduate students on practical applications of category theory to computer science, as I had found myself giving the same one-on-one lessons over and over. I enjoyed each opportunity to share knowledge as each was also an opportunity to get to know my comrades and to learn new perspectives they had to offer.

A peculiar quirk of my uncommon reading disability is it makes me a very careful reader, and so I act as editor when possible. After many editing sessions, I have learned how to offer constructive criticisms that provide clear directions for improvement while leaving enough freedom to adapt my suggestions to the authors' own style and intentions. Each occasion furthermore gives me the chance to learn about their research in detail and learn from them new ways to communicate ideas. Thus, I have found that I myself can gain from each piece of advice I give.

Over the years I have had the opportunity to mentor, educate, and critique, many of the roles I am grateful to my own advisor for. I look forward to advising my own graduate students so that we may learn from each other and work together to improve the state of the art in programming languages.