Teaching Philosophy

As teachers of computer science, we help shape the values of our field as it is practiced beyond academia. I believe our most important educational mission in computer science is simple: we must teach that programmers should understand their programs. That includes both encouraging clear thinking and careful inquiry, but also teaching a culture of accountability, where programmers should exercise agency and take responsibility for the increasingly automated and complex systems we create.

A Focus on Why

My experience as an engineer at Google has given me a vivid appreciation of the pivotal role that our programs have in affecting the perception of facts in the world. The power of software is not dissimilar to the power of the written word.

Because of this power, it is important that we teach our subject in a way that develops our students' readiness to take responsibility for the software they create. While computer science has traditionally been taught with an engineer's focus on *how* to solve a given problem, I think it is even more crucial to emphasize the importance of asking *why*. Why particular problems need to be solved, why some solutions should be chosen over others, and why our systems have the effects that they do. We must especially honor and celebrate the analysis of unintended consequences.

In teaching students to articulate 'why,' a key educational role is played by open-ended project assignments. As a teaching assistant for Robert Miller's HCI class, I stepped students through the process of understanding problems through user interviews; then coached as they honed design ideas in front of their peers; and then checked as they altered implementations in the face of user tests. The prescribed design process required students to be explicit about the choices they were making, and to exercise their analytical skills. I taught students that surfacing piles of bugs is a wonderful thing, even when not everything can be fixed. Troubles arise when problems go unseen.

This mission is becoming more important as our field evolves. The current revolution in machine learning is creating s a crisis in transparency in computer science. This fundamental transparency challenge drives my dissertation research as well as my teaching philosophy. Especially when teaching machine learning, we must teach students to understand the importance of spending the effort needed to understand the operation of our software, beyond superficial performance measurements.

A Focus on Inclusiveness

I also believe in the importance of welcoming everybody into our field. Even for nontechnical students, I think that a strong understanding of automation has become an essential part of critical thinking. But computer science suffers from a terrible diversity problem, and fully addressing the issue requires us to look at the way students are introduced to computer science, including before college.

Before enrolling at MIT, I spent three years devoted to promoting precollege computer science education, with the goal of welcoming a more diverse set of students into the field. I began by teaching weekly programming classes at majority subsidized-lunch schools and vocational schools, and based on this experience, I created a programming system, *Pencil Code* (https://pencilcode.net/) that is designed to be as welcoming to students as possible, while building a bridge to professional-level mastery.

Pencil Code was designed to overcome real-world educational barriers that get in the way of conceptual learning, such as beginners' difficulty in typing punctuation or remembering programming keywords. Its design was forged in the classroom, and refined based on the daily experience and challenges of teaching an ordinary class. It was designed for use by public-school teachers who teach a diversity of students. Because of this work, and because of adoption by a set of brilliant and enthusiastic teachers, Pencil Code is currently used by 2,000 students globally every school day.

My experience with beginners has taught me the pivotal importance of creating a welcoming environment to encourage people to participate in our field. Computers are unforgiving tools, and it is easy for students to see themselves as an imposter as soon as something does not work. In my university research work, I am conscious of the importance of creating a supportive, inclusive environment. As I have started to run research meetings, I watch the clock to make sure the floor is shared by all who have something to present, and when it is time to brainstorm, I especially work to make sure that women and minorities have access to plum research projects. I think it is important not to unconsciously disadvantage students because of how much they talk or how well they fit into a social group.

University Teaching

I will be enthusiastic to teach introductory courses in Software Engineering, Neural Networks, or advanced courses in Computer Vision, HCI, or Explainable Deep Learning. At MIT, I designed and taught an IAP seminar on The Structure and Interpretation of Deep Networks. I have also previously taught a section of an HCI class; reviews of my teaching have been positive (6.5/7 stimulated interest; 6.6/7 knowledge of subject; 6.6/7 helped me to learn).

I also enjoy research-related teaching. I have organized three research workshops while at MIT. A conference workshop on interpretable AI; a MIT symposium on robust and interpretable deep learning; and a conference workshop about creating systems to facilitate novice programming.

Mentorship

I have enjoyed advising both undergraduate and junior graduate students in my time at MIT. As an advisor, I think it is my role to telegraph enthusiasm for research directions that the community will be interested in, while teaching students that research is fundamentally a rigorous, collaborative learning and teaching exercise. At every research meeting, I ask students to try to teach us a little something new. We find the interesting parts, and we dissect both failures and successes with equal relish.

Each student I have mentored has been different, bringing a unique set of perspectives, skills, and goals, and I have worked with each student individually to find research problems that capture their interest, and where they have the skills and resources to explore productively.

I have found it is important to focus students on asking the 'why' questions in research. In machine learning it is very tempting to chase statistical performance numbers without looking at the data; but when when we anchor research on cases that can be directly visualized, it helps build intuition that leads to better work. I believe this perspective has helped put several junior students I have worked with on a solid path.

I have had the pleasure of working with several star students who I am confident have a bright research future after they graduate. I am proud of one student who graduated last year, Bill Peebles, who become a CS Ph.D. student at Berkeley, where he is continuing to publish research around the puzzles that we obsessed over together during his time as an undergraduate.