I have enjoyed multiple teaching opportunities at three universities, each proving a distinctive experience. In particular, I have been an instructor three times, a TA eleven times, head TA five, and worked on course development for six distinct courses. While I have learned much through all of these experiences, I will draw my teaching examples primarily from the largest course for which I served as sole instructor of record: CS2043.

Cornell's CS2043 is a technical "hidden curriculum" course with 130 students designed to teach everything from basic Unix shell tools to git and virtualization. Anonymous student course evaluations[1] were overwhelmingly positive, specifically praising the lecture and organization, and noting the course was "intuitive" and "engaging."

Teaching Philosophy

As a teacher, I keep in mind three key tenets: building classes that demystify their subjects, driving engagement with course material, and providing students multiple avenues to show mastery. These three tenets are key ingredients to a strong education experience. As instructor, I found opportunities to put these tenets in play—both at small (12-30 students) and medium (130 students) scale.

Demystifying The Subject I am excited to lead classes that spark student curiosity and inspire student exploration. I encourage students to look behind every curtain, teaching students that they can learn the basics behind even the most magical aspects of computer science. In my classes, this extends even as far as curriculum design and class infrastructure; at the end of my classes, students should know not just the material, but also feel capable of peeking inside even course submission and grading tools. In CS2043, I accomplished this by entwining assignment design and course infrastructure: all course assignments were prepared and submitted on the same server, and all class infrastructure was built using only the tooling taught in the first half of the class. I urged students to poke around, find easter eggs, use course infrastructure to communicate, or even "look inside" the autograding, quiz, and assignment submission systems.

Driving Engagement Active student engagement is essential for successful learning outcomes and directly tied to subject ownership, leading to mastery. In CS2043, I drove engagement through interactive demos and short quizzes. At the start of each class period, students would log into a shared Linux server and complete a short quiz in which they were asked to define any unix command of their choice—provided they had not defined it before. During the class period, I'd use this server for demos and have the students follow along, allowing students to independently explore the live effects of the commands on the server environment.

CS2043's quizzes were graded for completion—not correctness—allowing me to keep close track of student understanding without giving students too much extra stress. Guided by quiz responses, I was able to use review time built into each lecture to proactively correct misunderstandings or reinforce confusing material—before the corresponding assignments were due. Student reviews consistently mentioned high engagement, heralding the class as "so intuitive" and "fast-paced, but reasonable," indicating to me that this strategy was effective.

Allowing Students to Show Mastery In all my courses, I provide students multiple pathways to demonstrate mastery. In CS2043, our mix of project-based assignments, written assignments, and quizzes allowed students many ways to demonstrate their knowledge. When students missed project targets or began to consistently miss commands in quizzes, we reached out and invited them explicitly to meet with me (or a TA) for an interactive session, giving the students the opportunity to show their understanding and correct their past submissions. Ultimately, no student failed this course; every student was able to catch up and learn the material, no matter how far behind they had fallen in the semester.

As an aspiring professor, I am also keenly aware of the need to balance research and teaching. While the combined efforts I outline here may seem time-consuming, with the right infrastructure and organizational approach I have found the load to be quite manageable. For example, during the semester in which I taught CS2043, I was able to handle all course activities while also managing two distinct undergraduate research projects and submitting two first-author papers—a productive load for a graduate student even without considering teaching.

[1] available at `http://www.languagesforsyste.ms/files/milano-CS2043-evals.pdf`

Teaching Interests

I am especially excited to teach existing undergraduate-level courses in programming languages (including functional languages and compilers) or systems (including distributed systems, operating systems, and networking). I am also keen to introduce new courses, with ideas for a new undergraduate and graduate course.

As an upper-division undergraduate course, I would enjoy teaching a class in programming languages via language design. We would explore the various choices available to language designers and see how these choices manifest in existing languages—including their effect on tooling, community, and the perception of what the language is "good" for. At the graduate level, I would enjoy developing a class on application-focused system design. This class would combine paper-reading with class projects, aiming to motivate students to rethink traditional API assumptions and dive into what real-world applications actually need from their underlying systems in order to operate effectively.

Research Mentorship and Advising Philosophy

I have worked with many gifted undergraduate and graduate students during my time as a PhD student and postdoc. At UC Berkeley, I had the opportunity to prioritize advising and mentoring, including serving as an additional advisor for two junior PhD students and sole advisor for six undergraduates. A majority of my advisees chose to continue to graduate school and now study at Cornell, MIT, and Stanford. To protect students' privacy, I'll avoid referring to them by name or project here.

In mentoring I am guided by the needs and professional development of my students, rather than the immediate needs of a line of research. This means balancing students' research interests and agency, while ensuring students are never left adrift—at every stage of their work with me, from when they first join until project completion. A clear example of this philosophy in action comes when a student first joins my projects. I treat finding a research project for a new student as an exercise in curricular design. Aligning with my teaching approach, I draw up a series of "assignments" that nourish the student's foundational understanding and clarify potential research direction. For example, when an undergrad joined a project that required formal proof work, we began with close reading of similar work, collaborative discussion, and assignments that included concrete, approachable, week-by-week tasks, starting with basic proofs on toy type systems and leading to large-scale proofs on active research. This addressed the student's needs as a researcher, rather than just the narrow needs of a project, and ultimately led to a published paper.

But research is not a class; it is a collaboration. I strive to be a co-conspirator in research; I meet with students in neutral, student-centered spaces, like breakout rooms or hallway whiteboards. I join the students hands-on for projects, hold design sessions and even pair-program, as needed. I try to be available to students beyond the strictures of the hour-long weekly meeting and occasional emails, responding to students via DM instead of email, hopping on discord calls, or grabbing 15 minutes to check-in. I meet students where they are and stretch their development.

A critical component of my philosophy is giving my students a sense of ownership over research, by giving them a say in research direction. If their ideas or solutions come out of left field, they become an exercise in research communication! Through this process, students become stronger researchers, either by learning the skills needed to vet their ideas or by discovering an early—and meaningful—contribution to an ongoing project. For example, when an undergraduate's natural curiosity drew them to set aside tasks relevant to our paper submission and focus instead on extensions to our core ideas, I joined them on a journey to see what insights their curiosity would turn up. While this exploration didn't wind up making it into the paper, it did crystallize exciting future directions—and led the student to find a bug in the original system!

In both formal classroom settings and research mentoring, my student-centered approach has garnered positive feedback from former mentees and former students alike. I look forward to new opportunities to learn from my students as a professor.