I was fortunate to learn from great teachers and mentors during my studies. After experiencing teaching and mentoring through TA-ship, giving guest lectures, and mentoring both graduate and undergraduate students, I feel ready to teach and mentor the next generation of computer scientists and practitioners. My philosophy is based on a combination of what I have perceived as a student and my past experience of teaching and mentoring.

## Teaching Experience

I have experienced teaching and mentoring through various forms in my areas of expertise: software engineering (SE) and natural language processing (NLP). *I was a TA* for Programming Paradigms (an SE course), where I helped design the syllabus, assignments, and exams. The class had two sections for university students and professionals (pursuing the professional Master's degree), and I was in charge of the latter section (for answering questions; course materials are shared across sections). To help students comprehend the class materials, especially in the remote learning setting during the pandemic, I organized weekly office hours and maintained an online Q&A forum. Additionally, I have given two guest lectures in Machine Programming (an SE+NLP course) and one in Engineering Programming Analysis (an SE course). *All three lectures were highly rated by the instructors.* I have also enhanced my lecturing skills by giving *12 conference talks* (seven online and five in person, and including two talks I helped present as a non-author). Moreover, my experience in *organizing the NLP+Programming seminar at UT* (for five years) and the *cross-institutional UT-Cornell SE seminar* enriched my skills in presenting and organizing group discussions.

## Teaching Approach

The primary theme of my teaching philosophy is to engage and motivate students in learning through activities during lectures and extended support after lectures. Because I changed my primary field of study from physics (during undergraduate) to computer science (during Ph.D.), I have experienced different styles and methodologies of teaching. There are several methodologies that I found consistently useful in both fields or even more important for teaching in computer science: connecting lectures with research and practice, making teaching interactive, and encouraging students' self-learning. Based on these objectives, I finetune my teaching style depending on the type of course being taught. For senior-level and graduate-level courses, I give students more opportunities to engage in software projects and explore research topics, in the hope of motivating them to pursue career paths in the computer science field. For introductory-level courses, I put more weight on making the course accessible to all students from different backgrounds and with different skills.

**Connecting research and practice**. When designing courses, I always include relevant research topics and industrial practices. Specifically, I expose students to research prototypes and industry-strength tools in the form of demonstrations, assignments, or course projects. For example, two of the guest lectures I gave were derived from my own research projects, where I explained the connection with the courses and how students can use the knowledge they learned to do a similar research project. I have also helped with preparing an assignment where students used our research prototype to better understand software testing.

**Interactive teaching**. I like conducting live coding demonstrations in lectures and office hours, which I found very helpful for explaining new concepts to students. For example, in one guest lecture that I gave right after a lecture on the theory of seq2seq models, I demonstrated using

seq2seq models to solve SE tasks. Debugging is also an important part of demonstrations when code does not run as expected, as students can participate in solving the issues using the knowledge they just learned. I also welcome questions in class, which is a signal that students are paying attention, and I will pause at appropriate intervals to encourage questions. Moreover, I collect feedback from students to adapt my teaching style to their learning style.

**Encouraging self-learning**. Self-learning is an important ability for students to become successful independent researchers or software developers. Specifically, students should learn to organize their own way towards completing the given task—including designing algorithms, selecting tools and libraries, etc. To give students the opportunity to practice self-learning, I leave a part of my lectures and assignments open-ended. For example, in the assignments I helped design for Programming Paradigms, we encouraged students to add documentation and tests in their coding assignments, and many students did so.

## Mentoring Approach

I mentored three undergraduate students and two junior Ph.D. students, which has resulted in six publications, with two more under submission and two more in preparation. One Ph.D. student that I mentored is from an underrepresented group. Besides that, I have been the go-to person for my labmates to ask for advice and informal mentorship starting from my second year (despite being the youngest in absolute age). My job as a mentor was initially limited to technical discussions (e.g., doing code reviews and pair programming), and later, as I became senior, expanded to things like organizing projects on a high level, giving advice on career paths, discussing ideas, etc. My motivation comes from enjoying the mutual learning that occurs during the collaborations with mentees and the excitement when my advice contributes to the mentees' success.

From my experience of transitioning from studying physics in my undergraduate to researching computer science in my Ph.D., I know very well how important mentorship can be for graduate students in their first year. For myself, the hands-on experiences from my advisor were extremely helpful. When mentoring new students, I usually spend enormous effort in the first project to bootstrap their skills. For example, I would frequently review progress and plans, brainstorm ideas, and do pair programming. I found that once the mentees are equipped with adequate technical and thinking skills, they are self-motivated to continue their research momentum without my input.

As a mentor, it is important and challenging to deliver the knowledge of learning to learn. Rather than simply giving the mentee a list of tasks to do, I will make sure the high-level motivation for the tasks is clear. I like to discuss why-questions, and I encourage the mentees to ask such questions to help the mentees to understand the connection between ideas and implementations. I also share my own stories of learning new skills and solving challenges, so that the mentees can relate and to demystify some issues they initially think as hard or unique to them.

## Teaching Interests

Based on my teaching experience and research area, I feel qualified for teaching courses related to SE, NLP, software testing, machine learning, programming languages, compilers, and data structures. I also plan to develop a course on SE+NLP, in which I will cover topics on the applications of NLP-originated machine learning models on SE tasks (e.g., language models for code generation, seq2seq models for code summarization), the framework for implementing these models, and techniques for improving these models.