Joseph Near

Teaching Statement

I first became excited to pursue research due to my experience as an undergraduate instructor for the programming languages course at Indiana University, and I remain convinced that teaching is one of the most important aspects of academia. Teaching provides the opportunity not only to transmit knowledge to the next generation of computer scientists, but also to inspire curiosity and creativity.

My experience—both in research and in teaching—has shown the value of exploring a concept using executable artifacts to better understand that concept. As computer scientists, we are in the fortunate position of being able to build such executable artifacts on which we can perform experiments. In my research, I always test my own understanding of new concepts by implementing and playing with them; in my teaching, I encourage students to understand concepts by designing their own experiments and drawing conclusions from the results. I therefore favor hands-on, experiment-based learning, and use programming-based demonstrations and assignments in my teaching, both to promote understanding and to evaluate students’ progress.

Teaching Experience

I had my first teaching position at 13 years old. My first summer job was as assistant teacher in a QBASIC programming course at an elementary school—the same course that introduced me to programming. I taught this course for ten summers, eventually taking over as head teacher and modernizing the course to use Python. This position provided my introduction to teaching with executable artifacts: kids as young as 9 years old took the example games I provided, and made explosions bigger and characters move faster. These examples inspired students to explore programs and discover how programming works.

For six semesters, I was an Undergraduate Instructor for C311, the programming languages course at Indiana University. I began with grading, answering student questions, and leading recitations, but my duties eventually expanded to giving lectures, holding office hours, and writing course materials, assignments, and exams. In 2008, I was selected Undergraduate Instructor of the Year by the Computer Science Department.

In C311, we taught the semantic implications of language design decisions by assigning students the task of building a series of interpreters. This strategy has the advantage of communicating the formal semantics of a language in a medium that students already understand—code—instead of requiring them to learn a new formal representation like operational semantics. Moreover, the resulting interpreters are executable, meaning that students can experiment with them to gain an intuition about the results of a change to the language semantics.

The interpreter-based approach was also effective for evaluating students and helping those who were falling behind. Building an interpreter to implement a particular semantics is difficult without a deep understanding of those semantics, so a student without sufficient understanding of the course material was usually unable to complete the week’s assignment, and the parts of the interpreter the student found difficult corresponded to the concepts he or she failed to grasp. I found this to be an extremely useful tool in gauging student progress.

As a Teaching Assistant for 6.005, the undergraduate Software Engineering course at MIT, I taught software engineering techniques in recitation sessions by connecting the concepts used in lecture to concrete programming projects. For example, after a lecture on the concept of encapsulation, I led a recitation on the use of interfaces, and assigned the students a programming project involving the use of interfaces to implement a data structure for safe concurrency. The mixed approach of conceptual lectures and large programming projects gave students hands-on experience putting the concepts into practice and highlighted problems in their understanding.

At UC Berkeley, I was a co-instructor in the graduate course 294-116: Secure and Intelligent Programming. This course was designed to give graduate students an overview of recent research in security and privacy topics in programming languages and machine learning. The course was organized around guest speakers, who presented on and discussed their important works in these areas. I was responsible for organizing speakers, leading discussions during class, and grading student projects.
Teaching Interests

I am eager to teach courses in data privacy, software security, programming languages, and software engineering, at both the graduate and undergraduate levels. I would be particularly interested in teaching courses on differential privacy, programming language semantics, type systems, and static analysis and verification.

My teaching and research experience touches a number of different disciplines, including theory, security, privacy, databases, distributed systems, programming languages, formal methods, and software engineering. I would be qualified and ready to effectively teach undergraduate courses in these areas.

At the graduate level, I am excited to teach seminars on differential privacy, practical static analysis techniques, and security type systems. Such classes have helped me to direct my own research within the larger context of our field, and I believe they are an important part of a graduate curriculum.

Advising Approach

I am always eager for opportunities to mentor students. At UC Berkeley, I have had the opportunity to mentor four undergraduate researchers. This experience has helped me to explore the important challenge of effective communication to ensure the student’s success, since each one has been most productive under slightly different patterns of communication. In one case, working closely with the student, including meetings almost every day, resulted in increased enthusiasm and progress by the student. In another case, pairing two students together helped to improve the quality of code both of them produced. For all four students, I found it effective to make myself available and communicate excitement about their ideas; all four plan to pursue PhD degrees due to their enthusiasm for research.