Teaching and Mentoring Statement

Amy Babay
December 2018

While I’ve recently completed my PhD in Computer Science, with research in distributed systems and networks, I began my undergraduate career as a Classics major and graduated with a BA in Cognitive Science. I didn’t start taking courses in the Computer Science department until my junior year, when I decided to learn programming to better understand areas of Cognitive Science that touched on computational linguistics and natural language processing. My path changed my senior year, when I took Intermediate Programming. The professor, Yair Amir, convinced me that computer science was interesting and a field I could excel in – I ended up staying for a Masters in computer science, then ultimately coming back to complete my PhD with Yair as my advisor.

My own dramatic change in career path has shown me the impact that good teaching and mentoring can have on students and motivates me to invest my effort, energy, and skills into helping students grow and succeed, both in the classroom and through mentoring in research. As a PhD student, I had the opportunity to co-teach the same Intermediate Programming course that changed my plans for four semesters. Discovering that I enjoyed teaching led me to volunteer to help with the Distributed Systems course for multiple semesters, to provide informal mentoring for students in the Advanced Distributed Systems and Networks course, to more formally mentor Masters students in their research, and finally to work with Yair to co-design and co-teach a new project-based course called Software for Resilient Communities.

Based on my teaching and mentoring experiences so far, my overall teaching philosophy is:

- **Expect a lot, but invest a lot:** My courses extend students’ capabilities and require them to complete meaningful projects, but I believe in investing in support systems that enable all students, who come from a wide variety of backgrounds, to succeed. Plus, I’ve seen that an instructor’s energy can be infectious, and showing students how much effort we put into our courses inspires them to do the same.

- **Focus on practical project-based skill building:** Just as I think it is important to take ideas all the way to deployment in my research, I like to have students implement ideas for themselves, as reducing abstract concepts to real systems typically exposes any gaps in understanding.

- **Connect learning to the real world:** I believe that connecting the skills they are developing to real problems keeps students engaged and positions them to begin applying their skills to create useful solutions to those problems, even while they are still students.

I look forward to expanding and revising this approach in the coming years, as I gain experience and learn from the other educators.

As a co-instructor for Intermediate Programming, I work hard to make the class hands-on and to interact with students one-on-one. Students are required to complete five substantial programming projects throughout the semester, a big change from the relatively small weekly assignments of the Introduction to Programming course that can initially seem overwhelming for some students. However, we compensate for this by devoting significant class time to working with students on their projects or other in-class programming exercises, and we are constantly available for additional help. The ability to actually talk with my instructor in this class was part of what got me into the computer science field, and I’ve seen the difference this approach can make in helping students...
with diverse backgrounds succeed. Some students in the course have grown up programming, aced AP Computer Science, and take the course first semester freshman year; others only realized their interest in computer science in college and show up with one semester of programming experience. Getting to know each student lets us encourage and help those who are struggling or lack confidence and allows us to engage more advanced students on a deeper level.

Providing this level of interactive experience and support in a period of exploding CS course enrollments requires a large investment of time and effort, and in 2015 I received the Computer Science Department Special Service award in part for my role in enabling the course to be taught in this way. We break the course into multiple 30-student sections, where each section runs independently as its own course. We create a team that includes the instructors, a graduate teaching assistant, and several undergraduate course assistants to allow us to meet with every student during in-class tutorial sessions and to provide fast responses and meet with students that need help outside of class. Part of enabling this approach to scale is educating the instructional team as well, and I work with the course assistants to train them on grading fairly, providing useful feedback, and helping students effectively. While the course assistants are excellent students themselves, they need to learn how to guide other students on strategies for debugging their programs or thinking through their designs rather than just solving their problems for them. This model of breaking up large classes to facilitate one-on-one interaction and hands-on learning has been expanded to the Introduction to Programming class at JHU and proven to be successful in courses with total enrollments up to about 100 students. For larger classes, even if the same level of in-class interaction is not feasible, I believe that building a strong instructional team to support students can make a big difference.

I also believe that students should be exposed to how they can use the skills they acquire to make a difference in the world. Connecting course topics to research can be an effective way to do this: In Intermediate programming, we include a class devoted to discussing research to get students excited about the kinds of things they can apply their new programming skills to. In Distributed Systems, I gave a guest lecture each semester on multicast and group communication protocols that included my own Masters research on a new totally ordered multicast protocol and discussed how it improved coordination performance in modern data centers (a few students even ended up implementing that protocol in their course projects). More advanced course projects can even lead to successful research. In Spring 2015, two students in the Advanced Distributed Systems and Networks course conducted a project inspired by the initial stages of my dissertation research on timely, reliable Internet transport. Emily Wagner and Amit Mehta developed an algorithm for time-constrained flooding, which provides optimally reliable communication while reducing costs compared with naive overlay flooding. Emily went on to complete a Masters that I co-advised, developing an overlay network simulation tool that was an important part of our IEEE ICDCS 2017 paper, which received the best paper award.

In Spring 2018, I took this connection to real-world effects a step further by co-designing and co-teaching Software for Resilient Communities, a 15-student undergraduate course in which students design and implement practical open-source software systems targeted at making our communities stronger. Students ranged from freshmen to juniors and were split into four groups, with each group focusing on a project aimed at a different level of community (from the Johns Hopkins community, to the Reservoir Hill neighborhood community in West Baltimore, to the Baltimore City community, to the US community). The goal of the course was for students to apply their computer science knowledge to solve real problems, gaining valuable experience in working with the “customers” of their work, communicating their ideas to technical and non-technical audiences, working with existing open-source projects, and collaborating to achieve ambitious goals.

The pilot offering of the course was a success: each group managed to create a useful product
or at least proof-of-concept. For example, the Hopkins team improved the course registration experience for Hopkins students by connecting the Semester.ly course-search and schedule-creation interface (originally created by a previous generation of Hopkins students and widely used among undergraduates) to the official registration system (SIS) through an “add-to-cart” button that would automatically export schedules created in Semester.ly to SIS. That button was used hundreds of times during the first registration period after its introduction. They also added the ability for students to connect with their advisors on Semester.ly, allowing advisors to view their schedules and leave comments. The Reservoir Hill team created a complete scheduling portal that our partners at the St. Francis community center are excited to use, and they also generated interest in using the same software framework that they built on (Lutece, originally developed by the city of Paris) as a platform for Baltimore City’s web services. The US team worked on a project closely related to my own research on intrusion-tolerant SCADA systems, successfully adding intrusion detection capabilities for a limited class of attacks to our Spire SCADA system. The combination of intrusion tolerance and intrusion detection is a research question I am interested in exploring further; a useful step by two freshmen, one sophomore, and one junior is impressive.

In May 2018, I received the Johns Hopkins Computer Science Department’s Professor Joel Dean Excellence in Teaching award and was a finalist for the Whiting School of Engineering Graduate Teaching Assistant award based on my role in creating the Software for Resilient Communities course and my overall record of teaching during my PhD. As a teacher and mentor, I care deeply about seeing my students succeed and am willing to invest my time and energy to give them the best possible opportunity to do so.

As a faculty member, I would especially like to bring this teaching approach to the following courses:

- **Basic undergraduate courses:** Introduction to Programming, Intermediate Programming, Data Structures.

- **Advanced undergraduate or core graduate courses:** Distributed Systems, Computer Networks.

- **New courses drawing on my research expertise:** Programmable Networks.

My experience with Intermediate Programming at JHU has given me a particular enthusiasm for getting to know students in basic courses and helping them gain both the strong technical foundation and the confidence to succeed in their future computer-science-related endeavors. In core courses in my general research area, I would bring my hands-on project-focused teaching approach to help students internalize the theory of distributed systems or networking by putting it into practice. In my specific area of expertise, I am especially interested in creating an advanced networking course that would cover the evolution of network programmability research, including active networking, peer-to-peer overlay networks, structured overlay networks, and Software Defined Networking. Such a class is closely tied to my own research vision of achieving new levels of network programmability by combining structured overlays and SDN and offers the opportunity to both get students involved in cutting-edge research and gain new insights into the subject by considering it from this broader perspective.