Teaching Statement

Denae Ford

When I think about teaching, I look forward to igniting a curiosity in other students. The opportunity to inspire an awareness in students of how interdisciplinary and vital computing is to our world is what drives me. Teaching students to take hold of the reigns that rule their world and direct them in the way they want it to go is important to me. In order to do this effectively, I subscribe to 3 different strategies in the classroom and mentorship setting: 1) active learning, 2) active listening, and 3) providing practical examples.

My Teaching Approach

Active Learning: When introducing new material in class I find it especially important to accommodate different ways of learning among students. To do so, I present and assess the same information I teach in multiple ways. An example of this would be: orally presenting a concept, visually showing an example of how it can be used in industry, and then providing students with an in class example to work through independently, compare answers in small groups, and finally go over the content together as a collective group. I found that the medium for which students grasp information can be transient and can change depending on each concept. I want my classroom to be able to accommodate that.

Active Listening: As a course instructor and mentor I have been able to develop by listening to my students and then tailoring my classroom to their needs. In short, listening and adjusting accordingly. My students will have different backgrounds and levels of familiarity with the material I present. To best support these varied experiences, being accessible to having one-on-one with students about material before or after class can be just enough allow students to communicate their concerns about grasping the material. This also allows me to be able to figure out what was confusing about my delivery of the material so I can revise how I assess their knowledge. My philosophy as an instructor is to put the information where the students can get it. As a faculty member, I feel it is my responsibility to help all students understand the concepts the best they can—a task that requires active listening. I have found that active listening particularly important in introductory programming courses where people are coming from a variety of majors or backgrounds and are familiar with different types of evaluations. For example, students backgrounds in computational biology may not be interested in programming for the same reasons a linguist and thus may have different expectations for assessments; my courses will consider their different motivations.

Providing Practical Examples: For computer science instructors, teaching involves many concrete examples, hands-on coding experiences, and collaborative work that reflects the skills students need to be successful after their postgraduate careers. To make learning most effective it is important to be able to offer concrete examples. One approach to do this in a culturally relevant way is through offer practical examples of how students will use the knowledge obtained in industry. In my previous lectures I have given, I have invited software testing engineers to my class to share the beauty in breaking things to build them back up stronger than ever. This further grounds the material presented to students such as diabolical testing strategies.

In summary, the process of teaching and learning is like teaching a person to fish. Affording others the opportunity to learn for themselves—and then teach others—is truly the greatest gift.

Experience

Lecturing: Throughout my Ph.D. studies, I have found that learning programming can happen at different stages and forms. I prepared lectures to undergraduates and graduates computer science students about being evaluated on software engineering skills in interviews, taught a middle school girls courses on web programming, and taught non-traditional students an introductory course on data science. All of these experiences were especially important in my development as an instructor. I learned the importance of timing when and how students will need the taught material. These experiences have encouraged me to pay close attention to what my students want to leave

my classroom empowered with and present relevant examples when explaining new topics. One example of how I support the needs of my students is by offering students open access to the lecture materials so that they can continue to tinker and learn outside of the structure lecture time.

Service: My passion for sharing the spark of programming has expanded beyond the typical college course and has poured into helping young people learn to code. I have served as a one-on-one tutor for high school students with an interest in creating games. I have also served as a course instructor for middle school coding camps for girls since 2013. Having hosted these camps at NC State University, I was able to introduce student's to their first time visiting a college campus. Being able to be the face and personality of computing for young African-American and Latinx boys and girls has allowed me translate my passion of service into an invitation to join the INTech Camp for Girls Board of Directors.

Mentoring: Serving as a mentor has allowed me to lead by example and guide new set of researchers to study interdisciplinary aspects of computer science. I have advised undergraduate students on their research projects on summer REU projects and semester long projects. All of these projects have resulted in an IEEE or ACM publication. For all of my undergraduates, who also all identify with a underrepresented group, working with me was their first research experience. As a fellow woman of color, to be able to introduce these women to what research can be, all while demonstrating how transferable the investigative skills of research are, has been especially rewarding.

Example Courses

Below are a list of courses that may complement the current curriculum along with existing classes that I am excited to teach:

- *Human Computer Interaction* — An upper level undergraduate or graduate level course describing the interdisciplinary basics of human-computer interaction and user-centered design.
- *Software Engineering* — An upper level undergraduate course that reinforces the software engineering development process, tools used to make the process efficient, and prepares students for how software systems are built in industry.
- *Introduction to Computer Science Course* — A lower level undergraduate course that introduces the basics of programming to CS and non-CS majors.
- *Evaluating Participation Mechanisms in Online Communities* — A graduate level seminar that reviews and analyzes community mechanisms used to foster inclusion in online communities. In this course students will learn about interdisciplinary mechanisms used to attract and maintain users in an online platform.
- *Supporting Platforms to Learn Programming* — A graduate level course that outlines how to support the methods people around the globe use to learn programming. In this course students will be able to learn about niche platforms, services they offer to programmers, and ultimately determine how to build one that better supports how people code.