Teaching Statement
Austin Clyde

A teacher is more than an educator for some students. In my teaching, I aim to be a role model, a mentor, and a person who provides that last step of courage for students to foster their voices. As a first-generation college student, my college professors opened the world as a place that can be explored, analyzed, and thought about. If I can show just a few students that the life of the mind is a possibility for them, that they have the capacity and power to think about the big questions, I feel I have shared the same gift I was given.

I strongly believe in developing three complementary skills in my students, whether this is their first course in Python or a seminar in AI ethics. I aim to nurture students' curiosity, share the tools to pursue that curiosity, and build their confidence and dignity in their scholarship to share it with their communities—academic or not.

A fundamental requirement to building these skills is creating an inclusive classroom environment where students feel like they belong and even that they have a stake in the overall project. A central goal is to help students feel comfortable in the context of learning computer science. For example, while teaching a computational thinking summer bridge course for first-generation low-income students, I found that sharing my background in the same program provided the students with a sense of comfort. It helped students feel comfortable with their background, wherever it is from. Students' lived experience is not something they should leave outside my classroom door. In my courses, I illustrate the diversity of the field by highlighting many of computer science's marginalized voices ranging from Alan Turing to Ada Lovelace. FGLI students are already more prone to alienation from academia than other students. The structure of a college course, the academic buildings, and the formal syllabi can create power dynamics that prevent students from feeling comfortable speaking up. This means, in my teaching, encouraging students to understand their difficulties in the classroom and reflecting on my implicit biases that may emerge in the classroom or interactions with students. Through fostering comradery in the classroom, students can be prepared for academia's community-first approach.

Curiosity. My teaching style is heavily influenced by inquiry-based learning, aiming to put students in the driver's seat of their learning journey. In the classroom, this often means encouraging students to see me as their partner in learning rather than the source of exactly what to know. While not all courses are amenable to full adoption of this approach, components of it can be included in nearly any lesson. For instance, in my introduction to computer science course, which had strict technical requirements for their upper-level coursework, I encouraged students to think historically about computer science and wonder why the topics are the way they are. During a lesson on binary and hexadecimal numbers, I pressed students to think creatively about the storage of numbers which led to a curiosity-driven conversation on alternative worlds of computer design. These constructivist conversations are well-tuned for computer science, where engineering and human choice sits on every topic.

Analysis. My courses often focus on a series of analytic tools. Rather than treating the material as a series of facts to be learned, I strive to show students the methods and questions that encouraged other scholars before them to arrive at the conclusions they did. Research is a discussion; it happens to take place over hundreds of years and in text. These tools come as theoretical ideas about building systems or programs in computer science coursework. For example, recursion is notoriously one of the most challenging concepts to teach in computer science. So, in my introduction to computer science courses, I bring recursion to the level of daily life and show students that recursion is not a foreign computer science concept, but a fundamental technique humans use to solve all sorts of problems. I do so with an in-class exercise where I have students attempt to sort blocks in groups, requiring them to take very discrete actions with a single arm (like how instructions in single-process algorithms are written). Students learned through this exercise to abstract away the technical difficulties of, say, getting a function signature correct from a deeper understanding of how a system should operate. In technical course work, I believe that when students grasp that the technicalities are historically contingent formalisms, they feel much more comfortable engaging at the abstract level, which is where more learning success happens.

Dignity. In preparing students for possibly entertaining an academic or research-focused career, they must instill a sense of ownership, confidence, and humility in their scholarship. By centering student assignments on a problem they are curious about, students often feel a sense of ownership because they are deeply curious about what answer they might arrive at. While some students may feel overwhelmed to the point of imposter syndrome, encouraging students to take what they perceive as a risk allows them to build up the muscle of confidence in their work. For example, the final project in one of my computer science courses centered on students finding and analyzing a public dataset of their interest. While many students felt comfortable stating a problem to which they wanted to know the answer, nearly all these students believed that they could not truly develop a solution, let alone dare make progress. For example, one student wanted to tackle a question regarding water quality in her neighborhood but feared the problem was too big. After numerous office hours and discussions, the student completed a phenomenal report of publication quality. At this moment, a teacher can affirm students' fears as valid, that in answering a question comes responsibility, while reassuring students that they possess the skills and the tools from this course to make a dent others would want to hear about.

Inevitably, my ideas about course design often incorporate different elements from various fields to show students the transferability of skills. I find a diversity of applications leads to each student finding a niche that sparks curiosity. It also helps students exercise higher-level abstract thinking and problem framing. Problem framing teaches students how to approach an unknown area of scholarship by translating the problem to fit their tools. For example, in a version of a standard introduction to computer science course taught in C, every lab assignment was drawn from a unique application area to answer questions a typical student might have. One lab drew on scientific computing to develop a gene-alignment algorithm for COVID-19 sequences, allowing students to understand the relationship between variants and their sequences while simultaneously responding to current events. Another lab drew on high-frequency trading, pushing students to develop their own algorithm for processing an order feed.

I take a holistic approach when assessing students, including long and short-form exams, participation, and projects. My computer science courses included frequent low-stakes assessments—one of my favorite tools. I would share a two-question quiz at the end of each class, accounting for a small fraction of student grades. These quizzes would be directly pulled from the lecture material to test basic understanding. While students were initially turned off by this, towards the end of the course, most students came to enjoy the quizzes to indicate whether they needed to come to see the TA or me for extra help.