I tend to be suspicious of educators who claim to be successful teachers, which I have seldom observed to be true in practice. Instead, I have often found that teachers whose peers and students put forth as examples of success tend to be the shining stars of their profession. As such, I make no claims of being a successful teacher in this statement—indeed, I often view my own efforts falling short of my own goals. Instead, I will try to present my past teaching efforts and attempt to characterize my teaching philosophy with respect to teaching style, student learning, assessment, and inclusive environments.

TEACHING EXPERIENCE
I have been teaching, in some official capacity, for the past eight years. Broadly, I have:

- Taught Computer Science at several levels, including: introductory summer camps; intermediate, major-required courses; and upper-level electives.
- Facilitated learning in small (8-15 students), medium (30-50 students), and large (70-100+ students) classrooms, each for at least two semesters.
- Designed courses from scratch and modified existing courses, including developing lecture material, assignments, projects, and assessments. These efforts include refining the material over multiple offerings of the course as well as incorporating research ideas.
- Managed staffs of graduate and undergraduate teaching assistants (ranging from 1 to 27).
- Participated in professional development programs to improve my teaching (Tomorrow’s Professor Today at UVA and the Graduate Teaching Certificate program at U-M) and to help other graduate students improve their teaching (serving as an Engineering Teaching Consultant at U-M).

TEACHING PHILOSOPHY
My approach to teaching is exemplified by the following anonymous quote from a student evaluation:

This was a great class. To be honest, I was a bit skeptical about taking the class hearing how rigorous it was. I decided to attend the first class on a whim and knew by the first meeting that this was going to be a great class. [...] I noticed after a bit that I was applying techniques learned in the functional programming section in my everyday programs. [...] This class was challenging but also a very welcoming place to learn. Kevin was a really great instructor and really made an effort to be accessible to us and engage the class. Thanks again for this phenomenal course!

No document I write will fully capture my teaching philosophy, but I might simplify and summarize my approach to teaching with four interrelated tenets: reaching students who feel left out, engaging and actively including students in the learning process, centering course design around goals rather than a list of content items, and teaching for student success. What follows are some brief ruminations on each of these tenets.

REACHING STUDENTS WHO FEEL LEFT OUT. Computer Science is a spectacularly interesting subject full of beauty and elegance, and I would like for people from all walks of life to appreciate the subject as I do. There are significant challenges with extant stigmas and misconceptions around the subject that cause some students self-select out. I use “students who feel left out” as an umbrella term for a diversity of individuals. These learners can be those who come from different fields of study, non-traditional students, and students from different socio-economic or cultural backgrounds, to name a few. The most rewarding aspect of teaching for me is when I’m able to reach these students—the challenge of doing so gives me continued motivation to improve and adapt my teaching.

I make no claims of discovering the “secret sauce” to make CS interesting and accessible to all students. Indeed, this is an ongoing effort, and I continue to tweak my content and delivery to reach more students.
My goal is not necessarily to “win over” students, but instead to teach engaging classes that make learning enjoyable and allow students to appreciate the elegance and utility of CS, even if it is outside their standard field of study. My approach to inclusive teaching can be characterized by two practices: recognizing that learning is diverse and varying teaching to suit, and using approachable language in class. I find that these techniques, while designed to support students who feel left out, are to the benefit of all students.

From my experiences teaching in both the humanities (German and an undergraduate TA) and the sciences, I find that students from one area often struggle with traditional instruction in the other. I believe this is due, in part, to students in the humanities and the sciences having varied training on the process of learning. I use an oversimplified model of this that breaks down learning strategies into two basic classes: memorization and rule abstraction. Loosely, I have found that some students prefer to—and perform better when they—memorize complicated facts and algorithms while others find more success in regenerating complex ideas and processes from a small set of ground truths and rules. As such, I design my courses to support both approaches to learning and to make this explicit for students. In general, my students have been quite self-aware of which approach works for them, and this verbal labeling helps students focus their learning.

I also try to make my language approachable to all students. Recognizing that all students might not have the same interests and background, I make an effort to vary my examples and explain background for students who might be unfamiliar with a particular idea. Language can be quite powerful, so I make a conscious effort to thoughtfully wield my words. I try to normalize struggle with material and avoid characterizing material as “hard” or “easy.” My efforts have not gone unnoticed; a former TA wrote the following to me the semester after I first taught an intermediate Data Structures and Algorithms course:

You also just brought a really kind, thoughtful and inclusive focus to staff that made me feel more comfortable bringing that sort of focus to the table myself. Essentially, you just made [the class] really human and approachable.

**Engagement and Active Participation of Students.** Students learn best when they are actively involved in the learning process, and this observation informs my daily interactions with students. In a typical class, I attempt to foster an environment in which students feel comfortable contributing answers and asking questions. I prefer presenting material using whiteboards (or digital whiteboards) rather than slides as this medium allows me to adapt course material on the fly in response to student understanding and questions. Active learning is a central tenet of my teaching philosophy. In larger classrooms, I often rely of think-pair-share activities (primarily for pragmatic reasons), but I tend to prefer techniques such as large group discussions, hands-on technology (coding in the classroom), interactive lectures, and activity-based review sessions. I also include frequent assessment techniques in my courses. For example, I have developed a practice of beginning each of my class meetings with a review of material from the previous classes. I ask students to verbally summarize and paraphrase the concepts they have learned and do not help the students unless they become particularly stuck on a topic. This assessment helps me to gauge student understanding and retention and also incorporates review for exams directly into each class.

**Goal-oriented over Content-oriented Learning.** I strongly believe a teacher’s primary role is to be a facilitator of learning. Instead of fixating on a set of topics to cover, I try to design my course material to focus on problem solving, computational thinking, process, and communication. Much like novels in a literature class might be chosen to allow for discussion of particular themes and literary devices, I view the topics and facts of a Computer Science course primarily as a vehicle for developing these high-level skills. In my own teaching experience, I have found that students find it easier to actively engage with the course material when it is framed around the development of problem solving skills: students can understand and appreciate the importance of the skill, even if the immediate utility of the content at hand might be unclear.

My views on goal-oriented education have largely been shaped by my own undergraduate education in the Liberal Arts. I received my degree from a department that rarely employed traditional lectures. Instead, “lectures” were often driven forward by student participation in response to questions posed by the instructor, and problem solving was often the focus. I have implemented aspects of this approach to learning in larger courses at research universities that have largely been content-oriented. First, I design my syllabi around a set

---

1 It turns out that a similar model has been presented in the psychology community to describe concept learning.
of learning objectives, which are articulated early and emphasised. Second, I have developed practices for helping instructors and teaching assistants (who are more comfortable with traditional lecture) transition to goal-oriented instruction. While teaching Data Structures and Algorithms at U-M, I developed an activity for teaching assistants to draft and critique learning objectives as part of the exam writing process. The TAs then used these as a framework for writing and evaluating candidate exam questions. This process improved the overall quality of the exams for the class and also resulted in fewer clarifying questions from students taking the exams. These learning objectives also continue to be used in the exam-writing process for the course.

**Teaching for success, not failure.** Bluntly, I do not see the utility of making a class hard for the sake of it; I do not view myself as a gatekeeper of knowledge who arbitrarily chooses which students are allowed to pursue said knowledge. I also do not attempt to make my classes “easy.” Ultimately, I want my students to successfully develop skills, learn material, and grow as intellectuals by the end of the term. Therefore, I attempt to strike a careful balance in my curricula between challenging and stretching my students and setting achievable expectations. Ultimately, this tenet of my teaching philosophy is the combined sum of my various approaches: accessible and inclusive teaching, varying instruction techniques, encouraging active participation from students, and designing courses around a set of goals rather than a list of content items.

**Teaching Interests**
I enjoy—and am interested in—teaching computing courses at all levels. My previous experience, training, and research experience position me well to teach courses in:

- Core introductory and intermediate courses, including programming, data structures, and algorithms.
- My areas of expertise: programming languages, computer architecture, and software engineering.
- Theoretical courses, such as theory of computation, discrete math, and formal proof writing.
- Upper-level electives (with preparation), including compilers, web development and embedded systems.

Additionally, I am interested in developing courses that extend beyond the nominal boundaries of computer science. Topics I am interested in include history of computers, computing and computation through the ages, philosophy of technology, ethics in computing, and music and technology. I believe that it is important for students to both understand where our field came from also also the cultural and societal impact of tools and algorithms they design. Such topics can augment existing interdisciplinary initiatives aimed at stronger cohesion between the humanities and computing disciplines. These topics also provide fruitful opportunities for team-teaching and interdisciplinary collaboration.