Teaching Statement

Suraj Rampure
rampure@ucsd.edu • rampure.org

An instructor's role in a large-scale computing course is akin to that of an orchestra conductor, responsible for coordinating several moving parts to achieve a common goal. In my view, that goal is to ensure that their students – who have different interests, motivations, and circumstances – achieve the course's learning objectives while feeling supported and inspired. In this statement, I will chronicle my attempt at achieving this goal by discussing key components of the courses I've taught as an instructor and TA over the past seven years, while touching on ways in which I plan to grow.

Well-structured and engaging lectures

In traditionally-structured courses, a student's intellectual journey with concepts often begins in lecture. I once read that the best teachers view teaching as a performance art and I have tried my best to embody that philosophy. In order to keep my students engaged, I try to make lectures feel like a well-crafted story rather than a sequence of technical jargon.

A specific pattern I like to employ when designing lessons is **motivate**, **elaborate**, and **apply**. An instance of this pattern I'm proud of comes from the **lecture** that introduces the `pandas`[1] DataFrame `merge` method in the UCSD freshman-level course **DSC 10**, *Principles of Data Science*. To start, I **motivate** the `merge` method by introducing a problem that we can't answer using our current `pandas` toolkit – specifically, I have one DataFrame with the prices of my phones and another with the stock of each phone and want to determine the total value of my inventory. By starting with this motivation, it makes it clear to students what the big-picture idea of the exploration is, in the event that they get lost in the technical details that follow. Then, I briefly introduce the syntax of the `merge` method, to prove to students that it can actually help us solve the problem at hand. Next, I **elaborate** by "peeking under the hood" and stepping through an interactive diagram (Figure 1) that compares the rows of the two DataFrames one-by-one and stitches them together each time there's a match, iteratively constructing the output DataFrame. Finally, I **apply** our new tool to another example; specifically, one that involves determining the number of rows that result

`phones`

	Model	Price	Screen
0	iPhone 13	799	6.1
1	iPhone 13 Pro Max	1099	6.7
2	Samsung Galaxy Z Flip	999	6.7
3	Pixel 5a	449	6.3

`inventory`

	Handset	Units	Store
0	iPhone 13 Pro Max	50	Westfield UTC
1	iPhone 13	40	Westfield UTC
2	Pixel 5a	10	Fashion Valley
3	iPhone 13	100	Downtown

	Model	Price	Screen	Handset	Units	Store
0	iPhone 13	799	6.1	iPhone 13	40	Westfield UTC
1	iPhone 13	799	6.1	iPhone 13	100	Downtown

Figure 1.
An interactive version can be found **here**.

[1] In DSC 10, we actually use a subset of `pandas` named `babypandas`, designed specifically for the course.

from merging two other particular DataFrames. I've noticed that this structure keeps students intrigued from start to finish.

I find it valuable to embed interactive diagrams in my lessons. Not only do I think diagrams help make lessons engaging and visually stimulating, but in following with the Universal Design for Learning framework[2], they provide students with self-standing visual explanations that they can refer to later on if my verbal and text-based explanations in lecture didn't click. I often find myself saying things like "if you're ever confused about how histograms work, come back to this diagram." To make these diagrams more prominent in DSC 10, I created the page **dsc10.com/diagrams** which embeds all such diagrams I created for the course. So far, I've mostly used Google Slides to create my diagrams, though I'd like to develop my Javascript skills to be able to create more expressive, web-native media (like **this**, for example).

All that said, no matter how well a lesson is delivered, the literature[3] tells us it's hard for students to retain information if they aren't frequently being asked to use what they're being taught. To keep my DSC 10 lectures active, I use a simple Google form, linked at **cc.dsc10.com**, which asks students to enter the answer to a multiple-choice "concept check" question that is asked in lecture. After I ask a question, I give students time to discuss the answer choices among themselves, before showing the distribution of answers among the class and asking for volunteers to explain their reasoning. (A simple tip I've picked up to encourage discussion when students aren't talking is to walk among them and say, "I should hear more talking!") In the future, I want to experiment with the peer instruction[4] method, which would allow students to answer concept checks twice – first individually, and then after discussing with peers. To make time for such in-class activities, I'll try presenting the more cursory material in each lecture as a pre-lecture reading, though I know this is easier said than done.

Polished, relevant, and fulfilling assignments

I believe that students' computing skills aren't formed by attending lecture, but from discussing and completing assignments. When developing an assignment, I start by identifying pertinent topics that students will have been exposed to when they start working on it, and carefully select examples that reinforce and extend those skills in meaningful directions. I prioritize releasing assignments that are polished and bug-free to minimize student frustration and confusion; I like to tell my instructional assistants that "everything we release should look like Apple designed it," meaning that every wording choice and formatting detail should be thought out carefully.

---

[2] CAST (2018). Universal Design for Learning Guidelines version 2.2. http://udlguidelines.cast.org

[3] Freeman, S. et al (2014). Active learning increases student performance in science, engineering, and mathematics.

[4] Mazur, E. Peer Instruction. https://mazur.harvard.edu/research-areas/peer-instruction

I've found that students are the most eager to learn when they're given the opportunity to apply course concepts to something that's somewhat personal to them. Given the variety of interests among students in large courses, I try to use a swath of examples over the course of the term, and frequently update assignments to switch out examples that are no longer relevant. In DSC 10, I've developed several new projects on topics that students found engaging; one that I am particularly proud of involves **UCSD admissions data**, in which students can quite literally see themselves in the data that they're analyzing. A more recent project I helped my co-instructor develop revolved around **Taylor Swift's discography**.

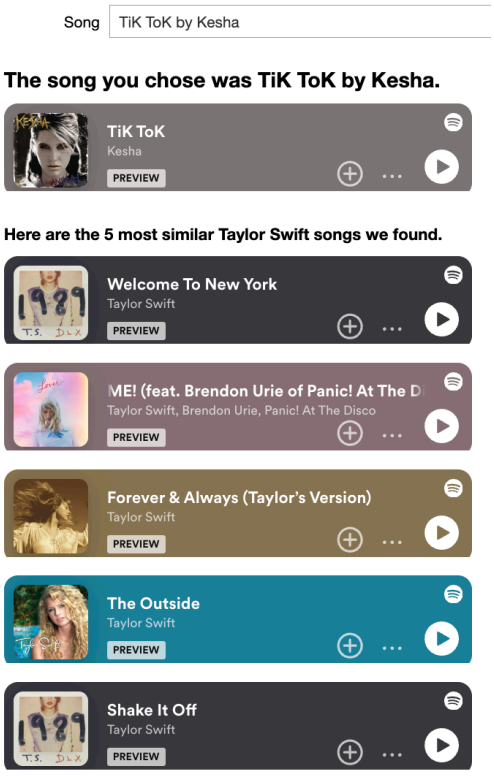

Figure 2.

I want students to feel a sense of accomplishment upon completing tasks in an assignment. I often do this by designing problems in which students create tangible artifacts through a series of challenging, but scaffolded steps. In classes like DSC 10, I often include interactive widgets that students can play with after they've implemented the necessary features in their code. One example comes from the aforementioned Taylor Swift project, in which students define a Python function that takes in the name of a popular song and returns the 5 most similar-sounding Taylor Swift songs based on features provided by the Spotify API and the Euclidean distance metric. Once they've implemented the function correctly, they get to interact with a widget (Figure 2) that allows them to pick an input song from a drop-down menu and immediately listen to the most similar songs directly in the Jupyter Notebook. Moments like these make otherwise mundane calculations seem material and fun.

Students appreciate the time invested in developing engaging assignments; in the End-of-Quarter Survey for DSC 10 in Fall 2023, one student said:

> *"I just want to say that all of the assignments were so incredibly well made ! i majored in data science because of the job prospects and financial security but the way this class was structured and taught me fall in love with the subject… the homeworks, labs, and projects were so well written with fun, relevant, and interesting questions i genuinely had the best time learning data science !! :D"*

In more theoretical courses, a type of problem I have come to enjoy creating is one in which students replicate a library implementation of an algorithm from scratch. For example, when I taught DSC 40A: *Theoretical Foundations of Data Science I*, the final **homework assignment**

revolved around the Naïve Bayes classifier. The assignment required students to compute, by hand, the posterior probability that a customer tips more than 18% at a restaurant, given various features. After finding these probabilities on their own, they then used `sklearn`'s `CategoricalNB` class to produce those same probabilities using just a couple lines of code. This flavor of problem demystifies the black-box nature of existing utilities and instills a deeper understanding of the underlying theory.

Challenging but fair exams

In programming-heavy courses, I view the role of exams as being to assess students' theoretical knowledge and computational thinking skills. I want my students to learn not only how to use the tools of the trade, but also how they work under the hood, because decades from now, the tools will change while the theoretical underpinnings will remain.

Especially given the prevalence of Generative AI-based code writing tools – which we **explicitly allowed** students to use, with caution, in the most recent offering of DSC 10 – there is no guarantee that the work students submit on take-home assignments reflects their own ability. As such, I believe in-person, on-paper exams are necessary to paint a complete picture of students' learning. In many of my classes, this is not the medium through which students work on most of their assignments; in such cases, I give students deliberate practice with in-person, on-paper problems in weekly discussion sections. With that said, I'm interested in exploring tools like PrairieLearn[5], which allow for digital, randomized exams for each student, along with a computer-based testing facility that ensures academic integrity.

To prevent students from having to context-switch between questions on an exam, I like to base my exams around central themes. In courses that revolve around tabular data manipulation, this often means having a single DataFrame that students refer to throughout the exam (**example**). I try to include questions of various difficulties; some questions are straightforward and allow students of all skill levels to demonstrate their confidence, while other questions are quite challenging and are meant for top students to differentiate themselves, and most are somewhere in between. To calibrate the length and difficulty of an exam, after my first draft, I have a handful of instructional assistants "beta-test" the exam, noting down the amount of time they spend on each subproblem along with any feedback. I then take this feedback to iterate on the exam and repeat this process a few more times until I'm satisfied that the assessment is fair and polished.

To allow students to demonstrate their growth throughout the term, I allow them to replace their scores on earlier exams with their scores on related questions from later exams. On the whole, my exams are typically quite rigorous, though I happily adjust students' grades at the end of the quarter to account for this whenever necessary.

---

[5] West, M., Herman, G. L., & Zilles, C. (2015). PrairieLearn: Mastery-based online problem solving with adaptive scoring and recommendations driven by machine learning.

Feedback mechanisms

It's important to give students regular mechanisms to share their experiences and provide feedback on the course. In courses with comparatively lighter workloads, I've administered surveys as often as weekly; in DSC 10, where homework and lab assignments are due weekly along with either a quiz, exam, or project in most weeks, weekly surveys are infeasible, so I provide Welcome, Mid-Quarter, and End-of Quarter Surveys. In the former, I try to learn more about my students and their motivations for taking the course, and in the subsequent two, I gather feedback on their experiences.

I've made significant changes to the structure of my courses in response to student feedback. For example, in DSC 10 offerings in AY 2021-22, students felt that they weren't getting enough practice for the course's on-paper midterm and final exams, given that all of their assignments were in Jupyter Notebooks. To address this, I built **practice.dsc10.com**[6] from scratch, a site that allows us to display old exam problems in two views – as old exams and as discussion worksheets. In AY 2022-23, we changed our discussion sections from being lecture reviews to being group time for working on these worksheets, giving students dedicated practice with exam-style problems each week. While the site received overwhelmingly positive feedback, students remarked that they still wanted more practice with exam-style problems. As a result, in AY 2023-24, we've instituted bi-weekly quizzes in discussion section, which require students to solve problems on-paper in a low-stakes environment similar to that of an exam. It's important to be transparent about these changes so that students know *why* a course is structured the way it is; in the End-of-Quarter Survey in Fall 2023, we told students that quizzes were a new component of the course and what their intended purpose was, and we're using the feedback we received to plan future offerings.

Invested and confident course staff

I'd find it impossible to run a computing course at scale without the support of instructional assistants (IAs). IAs in my courses collectively hold dozens of hours of office hours a week, which provide students with regular access to 1-on-1 help. No matter how hard I try, I will never have the lived experience of having actually taken one of my courses; not only have the majority of my IAs actually taken the course they're helping teach, but they come with a breadth of backgrounds and perspectives, giving my students a diverse pool of mentors to interact with. The **staff page** of a course website gives students a glimpse into this pool.

Along with the students in my classes, it's important to me that my IAs feel empowered and supported. They help behind the scenes with mission-critical tasks, like designing new assignment problems and writing autograders, maintaining students' grades and handling exceptions, identifying students in need of additional support, and answering questions on message boards, just to name a few. I try to provide IAs working on these tasks both guidance

---

[6] A longer-term goal is to polish the infrastructure used to create **practice.dsc10.com** so that other instructors – both in and outside of UCSD – can adopt it more easily. I've already created **practice.dsc80.com**, for DSC 80: *Practice and Application of Data Science*.

and autonomy, while making sure they know that there's space to make mistakes and learn from them. I hold weekly staff meetings and take my team out to dinner multiple times a quarter; such venues allow us to discuss ways in which to improve the course and also allow me to provide more casual mentorship to my IAs while building a sense of camaraderie. Not only do I believe that an empowered, tight-knit staff produces great work, but I think IAs who are satisfied with their job are more likely to indirectly encourage students to apply to serve as IAs by sharing their experiences candidly. Ultimately, the students I teach the most closely each term are my IAs; in larger classes, I'll have upwards of 20 IAs, meaning that overseeing my staff is like running another class. I expect a lot out of my IAs – for instance, I like to challenge them to return quiz and exam grades to students within 24 hours of the exam – but I show my investment by scanning and grading alongside them.

Moving forward, I'd like to invest more energy in more directly training my IAs on effective pedagogy, including in how to prepare to help students with specific problems in office hours and how to develop new content (e.g. homework problems). Outside of the term in which I taught the first-time IA **training seminar**, my guidance has been rather ad-hoc, and I know there's a wealth of knowledge in the broader computing education community on this topic that I should look to.

Conclusion

When teaching at scale, I've found myself wearing multiple hats. One of those hats is that of a content creator, who ensures lectures are written and delivered to be coherent and engaging, assignments are developed to be polished, stimulating, and relevant, and exams are written to be comprehensive and fair. Indeed, this is where the majority of my time is spent. Another one of those hats is to be a leader, in which I try and maintain the productivity and spirit of both my students and IAs. I have fallen in love with the act of coordinating each of these components just as much as I love the end result, which – when everything is executed correctly – is that students feel a sense of achievement.

I recognize that I'm early in my career and still have plenty to learn about how to teach computing. In the statement above, I recognized vectors for future growth, but more generally, I want to surround myself with students and instructors who will push me to challenge my assumptions about effective teaching. As a member of the Teaching Lab in the CSE Division at the University of Michigan, I know I'll be in such an environment.