Philip Guo **Teaching Statement**

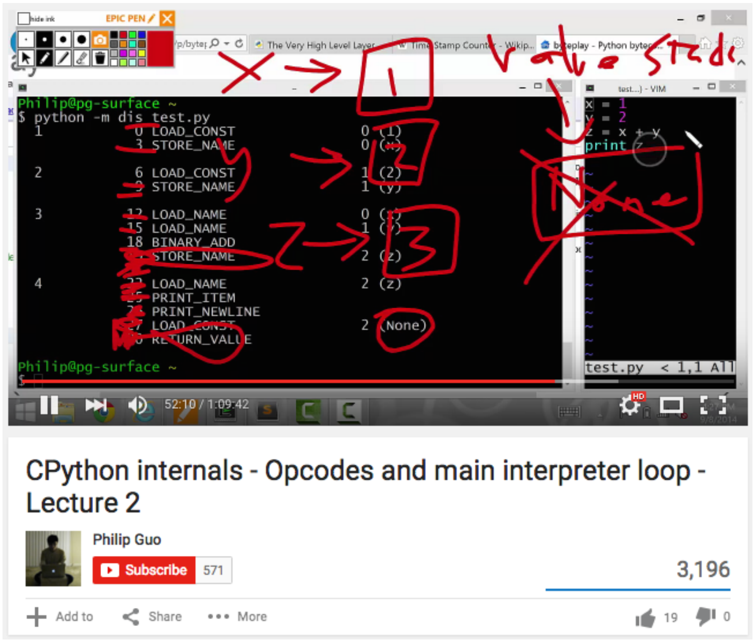
My diverse teaching experience, coupled with studying research-based best practices for pedagogy [1,3,4], has helped me develop my core teaching philosophy: ***be rigorous, engaging, and inclusive***.

I have taught courses in multiple roles: as a lab assistant, teaching assistant, and lead instructor. As an undergraduate at MIT, I was a lab assistant for the large introductory CS1 course, a digital electronic systems lab course, and a software engineering lab course. As a master's and Ph.D. student at MIT and Stanford, respectively, I was a teaching assistant for software engineering, programming languages, and two different graduate-level compilers courses. As a postdoc at MIT, I was one of four co-instructors of the 300-student HCI and user interface design course. Finally, as an assistant professor at the University of Rochester, I created and led two new courses: principles of web application development (advanced undergraduate level) and dynamic programming languages (graduate level).

My Teaching Philosophy: Be Rigorous, Engaging, and Inclusive

Rigorous: I strongly believe in teaching rigorous fundamentals that transcend the ad-hoc technological fads of the moment. For instance, in my web application development course, I teach foundational principles that underlie the modern Web, such as HTTP, REST, DOM, and Ajax. I emphasize to students that the specifics of which programming languages, libraries, frameworks, and deployment services are in vogue at the moment change at a rapid pace, and that these new technologies are easy to pick up on-the-job once they have developed strong foundations. In my dynamic programming languages course, I take students on a deep dive of the Python interpreter source code to show them how a modern dynamic language is implemented from the ground up in C. Again, my goal is not to train students to become specialized Python developers, but rather to illustrate general principles that hold for any dynamic language such as JavaScript, Ruby, Scheme, and future languages that have not even been invented yet. I have noticed that students take a course more seriously when the instructor is firmly committed to rigor and high standards.

Engaging: To bring technical subjects to life, I teach using a touchscreen laptop where I write code live, run the code, and then draw digital ink sketches on the screen. The figure on the right shows a video capture from one of my lectures on dynamic programming languages. Here I have just run a Python bytecode disassembler and used a red digital marker to draw diagrams over the terminal output. From both my own firsthand experiences and my 2013 research measuring the efficacy of different MOOC video formats at edX [2], I have found my live coding and digital sketching technique to be more engaging than using only PowerPoint slides. To help students review for exams, I record, edit, and upload most of my lecture videos, as well as extra video tutorials on topics such as command-line basics, version control with GitHub, and setting up web application development environments. Another benefit of putting my videos online for free is that learners from



```
Philip@pg-surface ~
$ python -m dis test.py
  1           0 LOAD_CONST               0 (1)
              3 STORE_NAME               0 (x)

  2           6 LOAD_CONST               1 (2)
              9 STORE_NAME               1 (y)

  3          12 LOAD_NAME                0 (x)
             15 LOAD_NAME                1 (y)
             18 BINARY_ADD
             19 STORE_NAME               2 (z)

  4          22 LOAD_NAME                2 (z)
             25 PRINT_ITEM
             26 PRINT_NEWLINE
             27 LOAD_CONST               2 (None)
             30 RETURN_VALUE

Philip@pg-surface ~
$
```

```
x = 1
y = 2
z = x + y
print z
~
test.py   < 1,1 All
```

CPython internals - Opcodes and main interpreter loop - Lecture 2

Philip Guo

around the world can watch them. Several of my videos now have thousands of views. I am also considering using these videos as the basis for trying flipped classroom methods in the future.

Inclusive: Students learn best when they are in a safe and comfortable environment. That is why I apply validated pedagogical research about how to avoid implicit bias in the classroom [3,4]. Most visibly, I try to be continually aware of who is raising their hands in class and then call on a diverse variety of students to speak, not just the few outspoken students who tend to dominate in-class discussions. I also use gender-neutral and value-neutral language throughout my lectures, emphasize collaboration and personal growth rather than intense head-to-head competition, and avoid confrontational language that may disproportionately discourage members of underrepresented groups.

Mentorship

Mentoring students on research has been one of the most rewarding aspects of my job as a professor. In particular, helping students fulfill their potential to become producers of new knowledge is an indescribably wonderful feeling. I first learned to mentor as a postdoc at MIT, where I worked closely with two undergraduates and three Ph.D. students on their HCI and online learning research projects. My work with all three Ph.D. students led to top-tier HCI publications (at CHI, UIST, and TOCHI, respectively) that formed the basis for their dissertation projects.

In my first 1.5 years as an assistant professor so far, I have mentored 18 students on research: 3 Ph.D., 5 masters, and 10 undergraduates. Five of these students joined my lab directly as a result of excelling in one of my classes and expressing enthusiasm for working with me. I am especially proud of mentoring undergraduates on publishable research projects: My undergraduate students Joyce, Mitchell, and Jeremy ***each published their own first-author conference paper*** as a sophomore, junior, and senior, respectively.

Example Courses

I look forward to teaching courses related to human-computer interaction, data science, software engineering, and online learning. I am comfortable continuing to teach courses that I have taught in the past, such as introductory HCI, user interface design, web application development, software engineering lab, programming languages, and compilers. For these existing courses, I want to experiment with innovative teaching methods such as flipped classroom and peer instruction if the opportunity arises.

I would also like to create new courses related to my research focus, such as:

- Experimental design for HCI
- Introductory data science and data-centric programming
- Advanced data science techniques for computational researchers
- Rapid prototyping of systems for online learning at scale
- Analysis of large-scale online learning data
- Interdisciplinary research seminar on learning science + computer science

References

1. Susan A. Ambrose, Michael W. Bridges, Michele DiPietro, Marsha C. Lovett, Marie K. Norman. How Learning Works: Seven Research-Based Principles for Smart Teaching. Wiley, May 2010.
2. Philip J. Guo, Juho Kim, Rob Rubin. How Video Production Affects Student Engagement: An Empirical Study of MOOC Videos. L@S: Conference on Learning at Scale, 2014.
3. Jane Margolis and Allan Fisher. Unlocking the Clubhouse: Women in Computing. MIT Press, Nov 2001.
4. Jane Margolis. Stuck in the Shallow End: Education, Race, and Computing. MIT Press, Aug 2008.