Teaching Statement

Showing students the work behind every answer, then asking for theirs

Geoffrey Kam

November 13, 2026

In my first semester as a teaching assistant at Washington University, a student wrote on my evaluation that asking me questions was "like talking to a chatbot." The student granted that I knew the material, but nobody could follow me: I had been giving answers and keeping the work behind them in my head. Since then I have taught by showing my work and by giving students hands-on practice. Students see how I write code and approach a problem, do the same work themselves, and then show me theirs. A semester after that first evaluation, a student in Macroeconomics II called me "the best AI [assistant instructor] in my first year at WashU!" My aim is that students leave my courses able to carry out the tasks we covered on their own and to show how they reached each answer.

The chatbot comment came from first-year Ph.D. Macroeconomics I in Fall 2022, where 7 of the 16 students returned the evaluation and rated their agreement that I presented material clearly at 2.67 out of 5, and at an appropriate pace at 2.33. I began typing full step-by-step solutions to each problem set and working through homework with students in office hours. In Macroeconomics II the next semester, 8 of 15 and 7 of 15 students in the two sections responded, and clarity and pace rose to 4.38 each in one section and 4.50 each in the other. I then carried the same approach into code as a teaching assistant for two Ph.D. econometrics courses, Advanced Microeconometrics and Structural Microeconometrics. I led the sections in Python, wrote the solution code for the problem sets, and walked the class through the estimation code for dynamic discrete choice models one step at a time.

Students code along instead of watching

In Spring 2025 I designed and taught my own courses: Econ 4850, Economic Analysis with Python, for upper-level undergraduates, and Econ 6850, A First Python Course for Economists, for master's students. Both were half-semester introductions for students with little or no programming experience. The undergraduate course moved from Python syntax to the tools of quantitative economics: computing OLS coefficients with matrix algebra, solving the Solow model with Newton's method, minimizing a loss function by gradient descent, cleaning and plotting data pulled from FRED, estimating parameters by simulated method of moments, and solving a dynamic programming problem by value function iteration. I ran both courses as flipped classrooms. Students worked through the lecture notebook before class, and in class they coded along on their own laptops and solved short exercises on the code just introduced, which we then went over together. Five weekly assignments asked them to replicate and extend the class exercises and to write each one up as a readable report. I designed them so that an attempt paid off even when it fell short: part of each grade rewarded completion, and each problem came in small steps that a student could work through alone. The lecture notebooks and problem sets for Econ 4850 are posted on my website, geoffkam.github.io/teaching.html.

Course evaluations say little about these courses: none of the 10 students in Econ 4850 completed one, and in Econ 6850 two of the nine enrolled students did. Both rated every item on learning, organization, enthusiasm and rapport at 5 out of 5. One called the course "not for beginners," which tells me that students without programming background needed more support at the start. Next time, all students will complete a short diagnostic exercise before the first class, those without programming background will work through a setup module first, and I will move value function iteration to a later course to give the earlier topics more time.

AI comes in once students can read its code

An AI assistant hands students finished code that they neither wrote nor worked through, so they

cannot tell when it is wrong. I want students to meet the limits of AI in my course, where I can show them how the way they prompt and check it changes what it produces. In the first weeks I asked students to turn off AI autocompletion, so that they made their own syntax errors and learned from them quickly. Once we moved on to quantitative exercises, they turned it back on to save time for concepts and methods. In the final weeks I asked them to use AI as much as they could, showed them how to prompt it for better code, and had them check its code against results they had already derived. The aim is that they leave knowing how far AI can carry them on a similar task and where they must still check its work.

Students show their work and defend it

Each course ended with a short group research project. I met with each group of two or three to settle its question and method. Before presenting, the group sent me its slides and Jupyter notebook, and I read both to prepare questions about the group's own code and output. One group of undergraduates compared a hedonic pricing model of more than 44,000 Canadian housing listings with a neural network, to ask what the network gained in predictive accuracy and lost in interpretability. Before fitting the network, the students computed the OLS coefficients with matrix algebra, checked them against statsmodels, and estimated them again by gradient descent, the method that also trains the network. I asked them which method is faster and when. They worked out that the matrix formula wins with a handful of regressors but that its cost grows much faster with the number of variables. They could explain where the regression and the network share a method and where they part. The network predicted prices more accurately, but only the regression gave each feature's price effect directly.

I can teach the core courses and a Python-lab elective

I am prepared to teach principles and intermediate macroeconomics, undergraduate econometrics and computational economics. I am also happy to teach an introductory undergraduate course in any field of economics. I supported principles of macroeconomics twice as a teaching assistant at the Chinese University of Hong Kong, and econometrics has been the subject of all my teaching-assistant work since 2024. I could also contribute to the first-year Ph.D. macroeconomics sequence, for which I was a teaching assistant in both semesters.

I would also like to teach a new undergraduate elective, Inequality, Entrepreneurship and the Macroeconomy, built around the question of my job market paper: when successful founders go on to invest in startups, does public money create more venture-backed startups by funding more investors or by putting more money into each deal? The course would require intermediate macroeconomics, econometrics and a Python course such as Econ 4850, and each of its four units would come with a Python lab. Students would first measure top wealth shares in the Survey of Consumer Finances, estimate the Pareto tail of the wealth distribution and count the households above the accredited-investor wealth threshold, the pool from which startup investors are drawn. They would then scrape Form D filings from the SEC's EDGAR system to describe the number of investors and the size of each deal. In the third lab they would simulate a simple model in which founders who sell their companies become investors, compare funding more investors with funding bigger deals at the same public cost, and solve the model by value function iteration and then with a neural network, to see when deep learning pays off. The last lab would apply a difference-in-differences event study to the 2010 expansion of the capital-gains exclusion for qualified small business stock. Readings would combine research articles with chapters from three online texts: the QuantEcon lectures of Sargent and Stachurski for the wealth and model labs, Scheidegger's *Deep Learning for Solving and Estimating Dynamic Models in Economics and Finance* for the neural-network solution, and Chernozhukov and coauthors' *Applied Causal Inference Powered by ML and AI* for difference-in-differences. Students would hand in each lab as a notebook and finish with a group research project that they present and defend, as in my Python courses.

Teaching Experience

Term	Course	Level
Instructor (designed both courses), Washington University in St. Louis		
Spring 2025	Econ 4850 Economic Analysis with Python	Undergraduate
Spring 2025	Econ 6850 A First Python Course for Economists	Master's
Teaching assistant, Washington University in St. Louis		
Spring, Fall 2024; Fall 2026	Econ 8135 (formerly 5141) Advanced Microeconometrics	Ph.D.
Spring 2024	Econ 5175 Structural Microeconometrics	Ph.D.
Spring 2023	Econ 502 Macroeconomics II (two sections)	Ph.D.
Fall 2022	Econ 501 Macroeconomics I	Ph.D.
Teaching assistant, Chinese University of Hong Kong		
2019–2021	Econ 2021 Basic Macroeconomics (twice)	Undergraduate
	Econ 3430 Public Finance	Undergraduate
	Econ 3160 Game Theory	Undergraduate