# ES 2-04: Introduction to Computing in Engineering

Spring 2025
Steven Bell

## Welcome to ES 2!

We are drowning in data. Computers and satellites and cameras and sensors are churning out incredible quantities of data every day. Or spend a few minutes with practically any form of media, where you'll see percentages and statistics and charts strewn all over the place. Data feels solid, and claims with numbers attached sound more definite, but numbers can be manipulated in many, many ways, often without any intent to deceive. How do we make sense of all this?

In one sense, ES 2 is about learning to program: we'll spend a substantial fraction of the semester studying details of the Python programming language and writing code. But in another sense, the programming is just a means to an end. The real aim of this course is to develop programming skills as a tool for critical thinking in our data-driven world. In this course, we'll write code to simulate systems, process datasets, and create visualizations. Along the way, we'll discuss many of the challenges, pitfalls, and outright lies that come up when working with data.

After successfully completing this course you will be able to:

- Use the Python programming language as a tool for numerical computation
- Model physical scenarios with code, and analyze how assumptions affect the predictive power of the model
- Explain basic tools for quantitative analysis of data (smoothing/filtering, histograms, mean/median, curve fitting)
- Load complex data from various formats and make gorgeous plots which accurately portray that data
- Identify assumptions and errors in claims made based on quantitative reasoning
- Discuss the social implications of technical decisions, and explain ways that "neutral" systems can exhibit bias

## How will this course help you succeed?

The skills you build in this course will be helpful in many of your future courses and central to your career as an engineer. Programming is no longer the exclusive domain of "computer geeks"; it's how everyone from economists to biologists to astrophysicists get work done when they've got large quantities of data or complex calculations to perform. With a little bit of code, you'll be able to automate tedious tasks, sort through millions of data points, perform enormous computations, and create plots that make Excel look amateurish.

More than that, though, we hope that the critical thinking skills you develop through the discussions and exercises during this course will help you to be a critical and thoughtful consumer of data, in and out of engineering. The world desperately needs people who can rigorously analyze quantitative claims, and who can present their own findings with clarity and integrity.

## Where should you look for information?

Wherever you want! There are countless resources for learning Python, and you are welcome to use whatever resources you find helpful. Professional programmers look stuff up all the time, and I expect you to do the same.

That said, there are a few places where you should look for course-specific information:

**The course website**: the course home page (https://www.ece.tufts.edu/es/2/) will have links to the course schedule, readings, videos, slides, assignments and other materials.

**Codio** is where you'll find the Python course content and complete in-class exercises and some homework.

**Piazza** will be our communication hub for announcements, questions, and live chat. If you have a question about the course content or logistics, please post on Piazza rather than emailing the teaching staff. You'll usually get a faster response, and everyone benefits from the answer.

You will submit the weekly sociotechnical reflections on **Canvas**. Other than this, we will not be using Canvas at all.

## How will you succeed in this course?

**Stay engaged.** If you already have some programming experience, you may find the first few weeks of the course a bit slow. Don't check out! From past experience, students who think they know everything already tend to slack off and then fall behind and do poorly in the latter part of the course.

On the flip side, if you have no programming experience at all, you may find the first few weeks to be fast-paced and challenging. We will give you many opportunities to revise and resubmit your work, but don't let that become an excuse to let things slide. If you find yourself

**Communicate.** The course will be difficult at times, and will require you to develop new skills and new ways of thinking. Don't try to do this alone!

Connect with your classmates and work together. We encourage you to collaborate on every aspect of the course. Reach out to the teaching staff. We're available before and after class, in weekly office hours, and via Zoom throughout the week. Piazza is open 24/7 for questions and discussion.

A good rule of thumb is that you should work on something yourself for at least 30 minutes, but not more than two hours. If you've spent 2 hours on a problem and have not made any progress, take a break, reach out for some help, and come back to the problem later (and maybe get some sleep?). You do not get hero points for spending 26 hours on a problem or staying up all night; those are a poor use of your time.

**Monitor your own learning.** You may be used to evaluating your progress entirely based on external feedback, i.e., the grades a teacher gives you on assignments and exams. Although we will be providing feedback in several ways, we expect to you to take charge of monitoring your own learning. A major goal of this course is for you to think critically about the results of any analysis, and that includes your own! When you complete a problem, you should be able to convince yourself (and others) that it is correct. If you can't, try to pinpoint exactly what it is that makes you uncertain. Finally, as you work through programming problems, ask yourself whether you're understanding or just guessing until things work. A little bit of guess-and-check is normal, but when you detect that you're randomly trying things hoping to stumble on the correct answer, stop and get some help.

## How will you and we measure your learning?

**Textbook exercises**
Before many class sessions, you'll read a few pages of the Codio textbook and complete the embedded exercises. These include multiple-choice and fill-in-the-blank questions and short bits of code that you experiment with.

**In-class exercises**
We'll be doing lots of coding in class. This is the chance for you to try new techniques, make mistakes, learn with others, and ask questions as you develop your skills. While the in-class assignments are not graded, I expect that . This is how you will build your skills.

**Weekly coding projects**
There will be some sort of coding assignment every week. Some weeks this will consist of

Other weeks will feature a programming project which requires you to build a simulation, analyze some data, or make some plots. We'll be using real data as often as possible, and we will often discuss the results of these projects during class.

**Sociotechnical reflections**
Each Friday we'll have a sociotechnical discussion about the impact of computing and data analysis on real human problems, and you'll have a reading and reflection from each of these.

**Final project**
You'll work with a small team to analyze, visualize and present some data of your own choosing. You'll have an opportunity to showcase your work for the class and other ES 2 sections.

**Coding interview**
Writing code on paper is a little silly, but live-coding interviews are very common. There will be two interviews (one in early March and one during finals week) where you write some code and then meet with one of the course staff for about 20 minutes to talk about it and tweak it.

## How will grades be determined?

Pre-class textbook exercises will be credit/no-credit. You'll receive credit if you completed all of the assigned reading pages and corresponding exercises before class. Half the point of reading is make sure you're ready for class, so late submissions will not be counted for credit.

Programming exercises will be marked either *satisfactory* or *resubmit*. A *satisfactory* mark means full credit for the assignment; a *resubmit* mark means that you receive no credit but can resubmit up to twice to achieve a *satisfactory* mark.

The overall grade will be determined by tallying up the number of satisfactorily completed assignments:

| Component | Quantity | Points each | Approximate fraction of final grade |
|---|---|---|---|
| Textbook exercises | 20 | 1 | 6% |
| Weekly coding homework | 12 | 10 | 38% |
| Sociotechnical reflections | 12 | 5 | 19% |
| Final project | 1 | 60 | 19% |
| Coding interview | 2 | 30 | 19% |

The exact number of assignments (and therefore the exact grade breakdown) will vary depending on how the semester goes, but should be close to these numbers.

## What else do you need to know?

**Instructor:**

**Steven Bell** `sbell@ece.tufts.edu`

Halligan 112 (click here for directions)

Office hours:

- Wednesdays 10:00am-12pm
- Fridays 3:30am-5pm

- I'm also available other times by appointment, or just drop by my office any time the door is open. I'm generally on campus Tuesday-Friday, and available by Zoom on Mondays.

**Teaching assistants:**

Aliyah Weiss `Aliyah.Weiss@tufts.edu`
Anna Lee `Anna.Lee@tufts.edu`

**Prerequisites:**

There are no prerequisites for this course other than algebra and an eagerness to dive in and learn new things. In particular, we will not assume any prior programming experience.

**Late policy:**

If you don't think you will be able to turn in an assignment on time, you must send me a message via email or Piazza *before the deadline* and tell me the date by which you will submit it. I will grant any reasonable request, but I will hold you to the deadline you specify.

**Academic integrity:**

Collaboration is essential in real-world software development, so you are welcome (and encouraged) to discuss approaches and solutions with other students, and to help each other debug code. Likewise, is is expected that you will use online resources such as Stack Overflow, W3Schools, and GeeksforGeeks as you develop and debug your code.

However, as with learning a sport, the person who puts in the effort gets the gains, and therefore it is essential that every student put in their own effort to learn. Concretely, this means that each student must write all of their own code. It is not acceptable to copy code from another student or from the Internet, whether your are copy-pasting or just typing from another screen.

Likewise, while generative AI tools such as Github Copilot and ChatGPT have promise as development tools which amplify the ability of skilled developers, the emphasis in this course is on building foundational programming skills. *It is not acceptable to use an AI tool to generate code for any assignment related to this course, even as a reference or starting point.* I am well aware that generative AI can crank out solutions for every assignment in this course; introductory Python is one of the things it's best at. Sadly I'm also quite experienced at telling the difference between code that a student wrote and code that AI wrote.

In your submission, you must document anyone you worked with and any sources you used to develop your code (which is good practice in any software development context). Codio has extensive plagiarism-checking tools built-in, and we may use these on code you turn in.

I will report any suspected violations of this policy to Academic Affairs. I'm required to do this by terms of my contract with Tufts, but also by my own conscience. We are all incredibly lucky to be at Tufts; there is no excuse for squandering that opportunity by representing someone else's work as your own.

**ADA accomodations:**

If you need special accommodations (flexibility with certain deadlines, larger type on handouts, etc.), please initiate the process with the Tufts StAAR center (`https://students.tufts.edu/staar-center/accessibility-services`). Accomodations cannot be granted retroactively, so please do this sooner rather than later.