ES 2: Introduction to Computing in Engineering

Spring 2021
Joel Grodstein and 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. If you spend a few minutes with practically any form of media, 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.

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.

Given the time we're living in, we'll be looking at a lot of data related to COVID-19. Hopefully this is fun and empowering; showing ways that you can analyze data for yourself and critically analyze many of the claims being made by scientists, the media, and others.

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

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 domain of "computer geeks"; it's how we get work done when we'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?

The course schedule, videos, slides, assignments and other materials will be posted on the course website: http://www.ece.tufts.edu/es/2.

We will use Canvas only for posting grades and sharing readings published through Tisch Library (i.e., stuff we can't share publicly for copyright reasons).

The textbook is *Think Python, 2nd edition* by Allen B. Downey. It is available as a free PDF from the author (https://greenteapress.com/wp/think-python-2e/), or you can buy a paper copy on Amazon for about $35 (ISBN 978-1491939369).

You will submit assignments through `provide`; a link to the submission portal is on the course website.

If you have a general question about the course content or logistics, please post on Campuswire rather than emailing the teaching staff. You'll usually get a faster response, and everyone benefits from the answer.

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 to be a bit slow; even boring. Don't check out! From past experience, students who think they know everything already tend to slack off and then fall behind in the latter part of the course. We will provide some challenge exercises to push you to think a little more deeply, but we

Communicate. Pandemic restrictions have isolated us in many ways, and even during normal semesters it is easy to struggle unproductively (and even suffer) in silence. The course will be difficult at times, and will require you to develop new skills and new ways of thinking. Because we are committed to helping you through these challenges, we're available before and after class, in weekly office hours, and via Zoom throughout the week. Your TAs also host office hours throughout the week, and Campuswire is open 24/7 for questions and discussion. We encourage you to collaborate with your classmates on every aspect of the course. Communicate early and often!

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?

Reading quizzes and programming micro-exercises (20%)
Each week will have an assigned reading with an accompanying quiz. Each quiz will include some multiple-choice questions and a few short coding problems where you write up to 5 lines of code to practice using particular parts of the Python language to perform simple tasks.

Lab programming exercises (45%)
Most weeks will feature a programming project which requires you to build a simulation or analyze some data. We will often discuss the results of these projects during the recitation section.

Participation in weekly recitation sections (10%) The weekly recitation sections are an opportunity to discuss and debate some of the big questions related to programming and data analysis. Our aim is for these to be highly interactive, but for that to work, you must come prepared and ready for the discussion! We'll evaluate your participation based on your attendance, preparation, and ability to thoughtfully engage the discussion — not on how much you talk.

Oral coding interviews (25%) There will be two oral coding interviews during the semester, one approximately halfway through and one at the end. Much like a programming interview for a job, you'll log onto

an interactive coding website and have a conversation with one of the instructors while you work through a couple of coding problems. We've found this format to be a lot more fun than trying to write code on a traditional pencil-and-paper exam, and it gives you an opportunity to correct mistakes and demonstrate more of what you know in a short period of time.

What else do you need to know?

Instructors:

Joel Grodstein `joelg@ece.tufts.edu`

Office hours: Find me before or after class (Thursday 12-1:15pm, Halligan 111A), or Zoom by appointment.

Steven Bell `sbell@ece.tufts.edu`

Halligan Hall 228D (a few steps down the hall from the 2nd-floor kitchen)

Office hours:

- Mondays 12:30-2:00pm
- Wednesdays 10am-12pm
- I'm also available other times by appointment, or just drop by my office any time the door is open. I'm on campus Mondays and Wednesdays, and available by Zoom the rest of the week.

Teaching assistants:

Joel Dungan `Joel.Dungan@tufts.edu`

Office hours: Thursday 9:30-11, via Zoom

Zengxu Yang `Zengxu.Yang@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:

You have 8 no-penalty late days which you may use on lab assignments. To use one or more late days, include the line `LATE_DAYS = N` at the top of your Python program, where `N` is the number of late days you'd like to use. To keep our bookkeeping sane, you may only use late days in whole-day increments; an assignment 5 minutes late is one late day, as is an assignment 23 hours late.

Even in tough circumstances we still want you to give your best effort, so any assignment submitted after the deadline (plus any applied late days) will still count for half credit, up to seven days late. After seven days, we and the course have moved on, and we will not accept work for credit.

ADA accomodations:

If you need special accommodations (extra time on exams, 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.