# Syllabus (Final) - CSE 6040/x: Intro to Computing for Data Analysis, Fall 2019

This course is a hands-on introduction to programming techniques relevant to data analysis and machine learning. Programming exercises use Python and SQL. It assumes some prior programming experience (not necessarily in Python), but asks you to review and extend that experience in the context of data analysis.

- Instructor and course creator: Professor Richard (Rich) Vuduc (https://vuduc.org)
- Co-creators: Vaishnavi Eleti and Rachel Wiseley

## What will I learn?

You will build "from scratch" the basic components of a data analysis pipeline: collection, preprocessing, storage, analysis, and visualization. You will see several examples of high-level data analysis questions, concepts and techniques for formalizing those questions into mathematical or computational tasks, and methods for translating those tasks into code. Beyond programming and best practices, you'll learn elementary data processing algorithms, notions of program correctness and efficiency, and numerical methods for linear algebra and mathematical optimization.

The basic philosophy of this course is that you'll learn the material best by actively doing. Therefore, you should make an effort to complete all assignments, including any ungraded ("optional") parts, and go a bit beyond on your own (see *How much time and effort are expected of you?* (https://hackmd.io/Kb2VkGkdTfe9LQZEIGVJGQ#How-much-time-and-effort-do-you-expect-of-me), below).

## How will I do all that?

*(Assignments and grading.)*

The main vehicle for learning new material is a collection of **Jupyter (https://jupyter.org) lab notebooks**, each of which introduces some concepts and reinforces them with programming exercises. There are accompanying videos, but these are intended as a guide to the material, not all of the content, which the notebooks cover.

The notebooks are part of your grade mainly to force you to go through the material. However, you should expect to spend **extra time** on your own and with your peers thinking about that material, looking things up to better understand things, and really trying to master the material.

The way we assess how well you know that material is through three exams (two midterms and one final exam). These are regarded by many prior students as the toughest part of the course, but you'll have access to old exams for practice as early as the second or third week of the course.

Overall, the breakdown of your final grade is as follows:

- Notebooks: 50%
- Midterm 1: 10%
- Midterm 2: 15%
- Final exam: 25%

For Georgia Tech students, the threshold for the equivalent of an "A" grade is 90%, for a "B" 80%, a "C" 70%, a "D" 60%, and anything below is regarded as "not passing."

> There is approximately one notebook or exam due every week. The assignments vary in difficulty but are weighted roughly equally. Some students find this pace very demanding; the reason we set it up this way is that we believe learning to program is like learning a foreign language, which demands constant and consistent practice.

## What should I know already?

*(Prerequisites.)*

You should have at least an undergraduate-level understanding in the following topics. You'll have about a month at the beginning of the course to get up to speed on this material if you have any gaps or haven't touched these subjects in a while.

- Programming proficiency in some programming language
- Basic calculus
- Probability and statistics
- Linear algebra

What does "programming proficiency" mean? We use Python in this course and assume your prior programming experience (variables, functions, loops, and data structures) is enough that you can get up to speed in Python. Armed with basic Python, this course then aims to fill in gaps in your programming background that might keep you from succeeding in other programming-intensive courses of Georgia Tech's MS Analytics program (http://analytics.gatech.edu/), most notably, CSE 6242 (http://poloclub.gatech.edu/cse6242/).

If you already have a significant programming background, consider placing out. If you have no programming background, you will need to ramp up very quickly. See below for more specific guidance on what we expect on the two hardest gaps to fill, namely, programming proficiency and linear algebra.

## When are things due?

*(Deadlines and late submission policies.)*

Because students from all over the world take this course, we have standardized on all assignments being due at 11:59 UTC (https://en.wikipedia.org/wiki/Coordinated_Universal_Time) time of the designated date.

Please make sure you are aware of the due date and time for your local area. UTC is an international standard time that uses a 24-hour convention (e.g., 11:59 UTC is distinct from 23:59 UTC) and has no notion of "daylight savings time" changes. **You** need to figure out what all that means and how to translate that into your local time. **We will not grant extensions based on your misunderstanding of how to translate dates and times.** You may wish to consult online tools, like the Time Zone Converter; here is an example of TZC for 11:59 UTC on August 19, 2019 (https://www.timeanddate.com/worldclock/converter.html?iso=20190819T115900&p1=1440&p2=25&p3=33).

**Late policy.** For your lab notebooks, you get an automatic 72-hour extension on every assignment **unless otherwise specified**; please refer to the course schedule. (This extension does not apply to exams.) However, you will lose points every day the assignment is late, and we will not accept any assignment after the 72-hour period.

The penalty is a deduction of 15% of the value of the assignment each day. For instance, if the assignment is worth a total of 25 points, then you will lose (0.15 * 25) = 3.75 points out of 25 for each day it is late, up to 3 days. Keep in mind that there are many assignments, so any given assignment is only worth a few percent of your final grade.

**Exam procedures.** For the exams, you will receive a window of about one week in which to attempt the exam, with a hard deadline to submit and **absolutely no extensions.** We will announce the exact format of the exam during the semester.

## How much time and effort do you expect of me?

*(From the instructors' perspective, this section of the syllabus is the most important to accept before you sign up!)*

At Georgia Tech, this course is a 3 credit-hour the graduate-level (Masters degree) course. So what does that mean?

The "3 credit hours" part translates into an average amount of time of about 10-12 hours per week. However, the actual amount of time you will spend depends heavily on your background and preparation. Past students who are very good at programming and math reporting spending much less time per week (maybe as few as 4-5 hours), and students who are rusty or novices at programming or math have reported spending more (say, 15 or more hours).

The "graduate-level" part means you are mature and independent enough to try to understand the material at more than a superficial level. That is, you don't just watch some videos, go through the assignments, and stop there; rather, you spend some extra time looking at the code and examples in detail, trying to cook up your own examples, and coming up with self-tests to check your understanding. Also, you will need to figure out, quickly, where your gaps are and make time to get caught up.

As noted above, in past runs of this course we've found the two hardest parts for many students are catching up on (a) basic programming proficiency and (b) linear algebra, which are both prerequisites to this course. We'll supply some refresher material but expect that you can catch up. Here is some additional advice on these two areas.

**Programming proficiency.** Regarding programming proficiency, we expect that you have taken at least one introductory programming course in some language, though Python will save you the most time. You should be familiar with basic programming ideas at least at the level of the Python Bootcamp that most on-campus MS Analytics students take just before they start. A good textbook reference for Python as you go through this course is Jake Vanderplas's A Whirlwind Tour of Python (https://jakevdp.github.io/WhirlwindTourOfPython/).

We can also recommend several online resources, like CS 1301x (https://www.edx.org/course/computing-in-python-i-fundamentals-and-procedural-programming-2), which is Georgia Tech's undergraduate introduction to Python class. Students who struggled with this course in the past have reported success when taking CS 1301x and re-taking this class later. Beyond that, code drill sites, like CodeSignal (https://codesignal.com) and codewars.com (https://www.codewars.com) (the latter's absurdly combative name notwithstanding) can help improve your speed at general computational problem solving. Please spend time looking at these or similar resources.

Part of developing and improving your programming proficiency is learning how to find answers. We can't give you every detail you might need; but, thankfully, you have access to the entire internet! Getting good at formulating queries, searching for helpful code snippets, and adapting those snippets into your solutions will be a lifelong skill and is common practice in the "real world" of software development, so use this class to practice doing so. (During exams, you will be allowed to search for stuff on the internet.) It's also a good skill to have because whatever we teach now might five years from now no longer be state-of-the-art, so knowing how to pick up new things quickly will be a competitive advantage for you. Of course, the time to search may make the assignments harder and more time-consuming, but you'll find that you get better and faster at it as you go, which will save you the same learning curve when you're on the job.

**Math proficiency.** Regarding math, and more specifically, your linear algebra background, we do provide some refresher material within this course. However, it is non-graded self-study material. Therefore, you should be prepared to fill in any gaps you find when you encounter unfamiliar ideas. We strongly recommend looking at the notes from the edX course, Linear Algebra: Foundations to Frontiers (LAFF) (http://ulaff.net/). Its website includes a freely downloadable PDF with many nice examples and exercises.

## Can I work with others? Also: What can I ask of the teaching assistants (TAs)?

*(Collaboration policy.)*

You may collaborate with your peers on the lab notebooks at the "whiteboard" level. That is, you can discuss ideas and have technical conversations with other students in the class, which we especially encourage on the online forums. However, each student should write-up and submit his or her own notebooks. **Taking shortcuts will here will only hurt you on the exams.**

But what does "whiteboard level" mean? It's hard to define precisely, but here is what we have in mind.

The **spirit** of this policy is that we do not want is someone posting their solution attempt (possibly with bugs) and then asking their peers, "Hey can someone help me figure out why this doesn't work?" **That's asking others (including the instructors and TAs) to debug your work for you.** That's a no-no.

**"But what can I do instead?"** In such situations, try to reduce the problem to the simplest possible example that also fails. Please see Stackoverflow's guidelines on "MREs" – minimal, reproducible examples (https://stackoverflow.com/help/minimal-reproducible-example). Posting code, in that case, would be OK. (And the process of distilling an example often reveals the bug!)

In other words, it's fine and encouraged to post and discuss code examples as a way of learning. But you want to avoid doing so in a way that might reveal the solution to an assignment that you are being asked to produce.

*When posting questions in the online forums, the same policy outlined above applies.* Indeed, your peers will answer questions much faster if you pare it down to an MRE, and the instructors will advise the TAs to prioritize answering "well-formed" MREs over large code snippets.

You must do all exams completely on your own, **without any assistance from others.** You can do internet searches, but you cannot post questions or actively communicate with others during the exam period.

**Honor code.** All course participants—you and we—are expected and required to abide by the letter and the spirit of the edX Honor Code. In particular, always keep the following in mind:

- Ethical behavior is extremely important in all facets of life. Honest and ethical behavior is expected at all times.
- You are responsible for completing your own work.
- Any learner found in violation of the edX Honor Code will be subject to any or all of the actions listed in the edX Honor Code.

**Important note: No GitHub repos!** Students often ask if they can post their work on GitHub or in any other public source code repository. The answer is "no!" Doing so is, essentially, publishing your homework solutions. If we determine that you have done so, we will consider it a violation of the honor code and take action accordingly.

## Will I need school supplies?

*(Books, materials, equipment.)*

The main pieces of equipment you will need are a pen or pencil, paper, an internet-enabled device, and your brain!

We highly recommend the following two references for this course.

- William McKinney. Python for Data Analysis: Data wrangling with Pandas, NumPy, and IPython, 2nd edition (http://shop.oreilly.com/product/0636920050896.do). O'Reilly Media, September 2017. ISBN-13: 978-1449319793. Buy on Amazon (https://www.amazon.com/Python-Data-Analysis-Wrangling-IPython/dp/1491957662/ref=sr_1_4?s=books&ie=UTF8&qid=1513180913&sr=1-4&keywords=python+for+data+analysis)
- Jake Vanderplas. A Whirlwind Tour of Python (https://jakevdp.github.io/WhirlwindTourOfPython/). The full "source" of this book and a PDF version are freely available online (see previous link).
- Jake Vanderplas. Python for Data Science (https://jakevdp.github.io/PythonDataScienceHandbook/). The full "source" of this book and a PDF version are freely available online (see previous link).

## I have more questions. Where do I go for help?

*(Course discussion forum and office hours.)*

The main way for us to communicate is the online discussion forum, hosted on the class discussion site, Piazza (https://piazza.com). *We will post instructions on how to reach this site when the course opens.* We will make all course announcements and host all course discussion there. Therefore, it is imperative that you access and refer to this forum when you have questions, issues, or want to know what is going on as we progress. You can post your questions or issues anonymously, if you wish. You can also opt-in to receive email notification on new posts or follow-up discussions to your posts.

> The professor's email situation is bad, so do not expect timely responses on email queries.

Here are some pro-tips to improve the response time for your questions:

1. Make your post public (rather than private to the instructors), so that anyone in the class can see and respond to your post.
2. Adhere to the "Collaboration Policy," above (see *Can I work with others?* (https://hackmd.io/Kb2VkGkdTfe9LQZEIGVJGQ#Can-I-work-with-others)). If you create a post that violates this policy, the instructors may ignore your post or even delete it.
3. Post during the week rather than the weekend; the instructors are also trying to maintain some semblance of work-life balance, so you can expect slower responses over the weekend.
4. Be sure to tag your post with the relevant notebook assignment so we can better triage issues. (In Piazza, a "tag" is also called a "folder,"" though unlike desktop folders, you can place a post in more than one folder.)

**What if my question is private in nature?** In that case, you can make your post private to the instructors. (After pressing `new post` to create the post, look for the `Post to` field and select `Individual student(s)/instructor(s)` and then type `Instructors` to make the post visible only to all instructors—it's important to include all instructors so that all of them will see and have a chance to address your post, which will be faster than addressing only one person.)

**Office hours (GT students only).** We will have live "dial-in" office hours, to-be-scheduled. Watch Piazza for an announcement and logistical details.

**Accommodations for individuals with disabilities (GT students only).** If you have learning needs that require special accommodation, please contact the Office of Disability Services at (404) 894-2563 or http://disabilityservices.gatech.edu/ (http://disabilityservices.gatech.edu/), as soon as possible, to make an appointment to discuss your special needs and to obtain an accommodations letter.

## Topics and Schedule for Fall 2019

The topics are divided into roughly three units, as outlined below. A more detailed schedule will be posted when the class begins, but the typical pace is 1 or 2 topics per week and 1 notebook (homework assignment or exam) per week.

**Module 0: Fundamentals.**

- Topic 0: Course and co-developer intros
- Topic 1: Python bootcamp review + intro to Jupyter
- Topic 2: Pairwise association mining
  - Default dictionaries, asymptotic running time
- Topic 3: Mathematical preliminaries
  - probability, calculus, linear algebra
- Topic 4: Representing numbers
  - floating-point arithmetic, numerical analysis

**Module 1: Representing, transforming, and visualizing data.**

- Topic 5: Preprocessing unstructured data
  - Strings and regular expressions
- Topic 6: Mining the web
  - (Ungraded; notebook only) HTML processing, web APIs
- Topic 7: Tidying data
  - Pandas, merge/join, tibbles and bits, melting and casting
- Topic 8: Visualizing data and results
  - Seaborn, Bokeh
- Topic 9: Relational data (SQL)

**Module 2: The analysis of data.**

- Topic 10: Intro to numerical computing
  - NumPy / SciPy
- Topic 11: Ranking relational objects
  - Graphs as (sparse) matrices, PageRank
- Topic 12: Linear regression
  - Direct (e.g., QR) and online (e.g., LMS) methods
- Topic 13: Classification
  - Logistic regression, numerical optimization
- Topic 14: Clustering
  - The k-means algorithm
- Topic 15: Compression
  - Principal components analysis (PCA), singular value decomposition (SVD)
- Topic 16: Putting it all together
  - (Notebook only) Exact topic TBD, e.g., deep neural networks