PPHA 30550: Intro to Programming for Public Policy

Jeff Levy
levyjeff@uchicago.edu
Keller 3101

Spring Quarter, 2019

Course Information

Section 1: T & Th 9:30 AM - 10:50 AM
Keller 1002

Section 2: M & W 11:00 AM - 12:20 PM
Keller 1002

April 1st - June 15th, 2019

Office Hours

I will be available at my office (Keller 3101) on Tuesdays and Thursdays from 11-noon, or other times by appointment. Information on TA availability will be pending.

Teaching Assistants

Darshant Sumant - darshansumant@uchicago.edu
Ke Duan - kduan005@uchicago.edu
Ruolin Fang - ruolin@uchicago.edu
Camilo Arias - cariasmartelo@uchicago.edu
Guan He - guanh@uchicago.edu
Pei Zhou - tonypzhou@uchicago.edu

Course Objectives

This programming and data course is geared toward public policy students who have either no past programming experience, or minimal experience primarily

in prorpietary platforms like Stata and SAS. The goal is to prepare students both for future Harris courses and for entry-level positions working in policy research, such as research assistants in academia or as interns at think tanks. This course is part one of a two-part sequence, followed by Data Skills for Public Policy (PPHA 30531). Over the quarter, students will learn:

- The basics of public policy programming: the languages used, how to set up needed software, coding best practices, and how to work collaboratively.
- The Python programming language:
  - The basics, including logic control, loops, functions, classes, methods, and input-output.
  - Data exploration and visualization
  - Data APIs and web scraping
  - Econometrics and machine learning
- Simulations and agent based models
- The basics of creating websites
- The basics of working with big data, including distributed computing with Apache Spark (time permitting)

This course will take a narrow view of computer programming, focusing on learning skills necessary to work as a policy researcher, while passing over some more advanced skills that are common in computer science but rare or unused in data science and research. This includes, for example, an understanding of multiprocessing, unit testing, or the creation graphical user interfaces (GUIs). However, the skills learned in this class will provide a foundation for the student to pursue these topics in the future.

Software and Resources

It is strongly recommended that you bring a laptop to every class. You can obviously come without it, but it will be difficult to keep up if you cannot try things along with the lecture and during exercises. Mac or PC (or Linux) is up to you, however I do not use Macs and only rarely use Linux, and so may not be able to help you during class if you have issues related to the operating system.

There are no required text books for this class. As one of the most popular and fastest-growing computer languages in the world, Python is extremely well supported online. I expect students will primarily be using the official Python documentation and StackOverflow, which will be discussed in class.

For the data sections, however, I strongly suggest purchasing the text Python for Data Analysis 2nd Edition by Wes Mckinney, which is available online or in the school bookstore. Not only is it very useful both as a quick reference

and when read comphrehensively as a guide, it is also written by the author of Pandas, the package used for data analysis in Python. The package is free and open-source, so this is also a good way of giving back to the creator. If you purchase this it is very important you get the 2nd edition, as the original is too outdated.

There are two pieces of software that are required for this class, and three that I suggest, all of which are free:

- *You must come to class on day one* with the Anaconda Python Distribution installed already. Please select version 3.7. You can also use version 3.6 if that is what you already have installed previously. No version of Python 2.x is acceptable.

- *You must come to class on day one* with the GitHub Desktop installed. You may also use the Git command line interface if you have it installed from before and know how to use it, though it also comes as part of the Desktop download.

- The Atom open-source text editor. I suggest Atom, but perfectly viable alternatives include Sublime, Vim, and many others. You may also chose to use one of the IDEs (integrated development environments) that comes with Anaconda, such as Spyder. Whatever choice you make, please have it installed and ready to go on day one as well.

- The Notepad++ text editor. This is strictly optional, but I find this lightweight text editor useful for viewing raw data alongside my code when necessary. You can choose it as your text editor for coding, as well.

- A command line shell that offers more features than the default. This is also strictly optional, as all computers will have their own basic version. I like, for example, ConEmu for Windows, which provides handy tabs and other features that make working in it easier.

Note that overall, the software environment you choose for developing code is entirely personal preference. You simply must have some distribution of the programming language you will work in, some place to write your code, and some place to run your code. It is, however, easiest for this class if everyone chooses Anaconda + Atom, so we are all looking at the same thing.

Attendance

There is no attendance policy, and you don't need to give me excuses not to come. However, you will be fully responsible for the material covered in class, and while code from class may be posted, it will rarely if ever come with the explanations provided in class.

If you experience issues with attending class or completing work due to child care, please speak with me directly so we can find an accomodation.

Academic Integrity

Standards of academic conduct are set forth in the University's Academic Integrity guide.

As it pertains to comptuer code, it is *required* that all sources are cited in the comments of any code you submit for this class. If you find a solution on StackOverflow (or anywhere else online) then you must include a comment with the relevant URL. If you work with a classmate on an assignment *you must list each other's names in comments at the top.*

Your code must always be original and uncopied, but some similarities are to be expected. This can be somewhat subjective; for example, if you find a solution that involves a list comprehension on StackOverflow you must cite that link, but if you are familiar with list comprehensions and write similar code on your own, it does not need to be cited. How this is determined will be based on your entire body of work and covered class material, and will be at the discretion of the TAs. When working with classmates, it is reasonable to discuss approaches and solutions, but I advise you to never actually look at each other's code. This way you will more easily avoid any semblance of copying.

It is expected that your assignments will be a combination of your completely original code and code you have writen based on inspiration found elsewhere. It is hard to explicitly state the appropriate balance. Note however that if "too much" of your code is determined to be unoriginal based on citations, you may be given a chance to redo it with no penalty. If "too much" of your code is determined to be unoriginal and uncited, you may fail the assignment and be subject to further disciplinary actions.

Homework, Exams, and Grading

Your grade will consist of weekly assignments and one final project. There are no exams in this class. Assignments will be given out for weeks two through eight, and will be due by the start of the first class each week, as judged by the timestamp on the submission versus the start of class time. Late work is not accepted without prior approval or with a documented emergency.

Your grade will be calculated as 75% weekly assignments, 25% final project. I expect final grades will use the standard Harris curve of 1/8 A, 1/4 A-, 1/4 B+, 1/4 B, 1/8 B-, though this is not guaranteed.

Weekly assignment grading

- 65%: You correctly answer the question
- 25%: Your code runs without errors
- 10%: Your code is easy to read, commented appropriately, and follows correct Python style.

Partial credit will be given, so make sure you always turn in what you have. Note that the two lesser parts of the grading depend on you meeting some minimal level of the first; if you've written two lines of code on a project that calls for twenty or thirty, you will not be getting much credit for it running without errors, for example.

Also note that this means you can essentially answer a question correctly while still failing the assignment if your code has an unresolved error and doesn't follow Python style. Clearly in the real world you cannot leave a project with bugs in it, no matter how "close" it is to being correct.

Final Project

Your final project will be an original Python program that showcases the skills you have learned throughout the quarter. It will make up 25% of your final grade, and will be due by the end of the last class in week ten. Details will follow mid-way through the quarter.

Course Outline

This outline may be subject to change.

Week 1: Introduction

1. Introduction, software review and setup, and programming for public policy research
2. GitHub, case studies in public policy research

Week 2: Python basics

1. Running Python, writing Python, and understanding data types
2. Python logic control statements and loops

Week 3: Python functions

1. Python functions and lambdas
2. Python classes and methods

Week 4: Applying what you've learned

1. Agent based modeling, part 1
2. Agent based modeling, part 2

Week 5: The Pandas dataframe

1. Introduction to the Pandas dataframe
2. Basic dataframe operations, introduction to NumPy

Week 6: Data visualization

1. Introduction to Matplotlib
2. Customizing plots in Matplotlib

Week 7: Introduction to web scraping

1. Using Pandas data APIs, introduction to html
2. Python requests and BeautifulSoup

Week 8: Building websites

1. Basic web development with Python, plus live examples
2. Writing simple Django

Week 9: Stats, econometrics and machine learning

1. Statsmodels, NumPy, and Scikit-Learn, part 1
2. Statsmodels, NumPy, and Scikit-Learn, part 2

Week 10: Review, overflow, advanced topics

- Help with final projects, review of any previous material
- If time allows: Big data and Apache Spark, other advanced topics