INTRODUCTION TO PROGRAMMING FOR PUBLIC POLICY

JAMES SAXON

- Contact: James Saxon (jsaxon@uchicago.edu), TAs (2017 TBD)
- Discussion Board: Chalk. Please use – we will monitor.
- Meeting Day/Time: Section 1, MW 9-10:20am; Section 2: MW 1:30-2:50
- Lab Sessions: Thursday and Friday afternoons, time TBD.

Course Aims

The past decade has witnessed an explosion in the collection of 'big data,' and the sophistication and accessibility of the tools required to analyze those data. This has spurred government agencies and policy analysts to embrace novel, data-driven approaches to policy creation and evaluation.

This is an introductory course in programming and data analysis for public policy students with no prior coding experience; it is the first in Harris's new data science sequence. It is for anyone who wants to gather, explore, and share raw quantitative data – or work with others who do. The course has three goals:

1. We will first introduce students to the tools required to write and share code: text editors, the command line, and version control (git).
2. Students will learn to think algorithmically, translating self-contained questions into python programs. We will cover the fundamentals of the language including types, control, functions, input/output, and scripts. We will touch on debugging and (time-permitting) computability.
3. We will then cover tools and recipes for retrieving, cleaning, visualizing, and analyzing data.
   - Data science libraries: manipulating data with pandas, plotting with matplotlib and plotly, and running basic statistical and geographical analysis (GIS). The pandas structures resemble R, and are a useful groundwork for the second course in this series.
   - Relational databases (SQL): selecting and aggregating data from databases.
   - Web scraping and APIs: how to retrieve and use public data from the web.

Ultimately, students should be comfortable using what they've learned in further Harris/Chicago courses in programming and statistics (incl. Policy Lab) – and in research after leaving Harris. They should be confident independently finding and exploring new packages for those projects. They should know enough to productively collaborate on projects with engineers, and understand the potential of such work.

Throughout we'll emphasize good practices for collaborative code development. Particularly after the first weeks, we'll highlight how these skills apply to evaluating and improving policy.

Books and Resources

Online documentation ('docs') will provide the principal written resources for the class; sources are listed for each week. For many code projects/'packages' and languages, these sources are literally *the* standards that define how to use the products – and unlike books will keep up with the packages as they evolve. Locating these sources and using them to identify the methods or recipes you need

is an important skill in coding. The most relevant starting sites for us will be for python 3.5 and pandas. You will also likely find the question site Stack Overflow very useful.

This said, two of the O'Reilly books are very good and happen to be available electronically through the UC library: Think Python (Allen Downey) and Python for data analysis (Wes McKinney). Seeing the material multiple times will help – so please use these resources!!

Assignments and Grading

Weekly assignments (65%). Assignments will be posted on the class GitHub site at least one week before they are due. Work will be collected through Chalk and GitHub, so get a student account here. The first week's assignment will be graded only for completeness, and I will downweight the lowest grade by one half. Each subsequent assignment will be assessed on:

- **Correctness (60%)**: Most questions will be evaluated directly on chalk. In some cases you will have to submit plots, etc. to GitHub.
- **Style and performance (40%)**: A TA will review your code from GitHub. An automatic script will collect work at 1:30am, Wednesday morning. *You are responsible for ensuring that your push was successful.* A per-question rubric out of 4 is as follows:
  0. No apparent effort – the code is absent or effectively incomplete.
  1. A clear start has been made but the code hangs, crashes or otherwise provides an incomplete answer.
  2. The code is largely complete but gets the wrong answer. Readily identifiable modifications would result in a working solution. Commenting is absent or not meaningful.
  3. The answer is correct and the code is readable but uncommented *OR* it is plagued by a single small error but well-commented.
  4. The code is readable and idiomatic (pythonic), and employs the functionality covered in class to succinctly solve the problem. The are no bugs – the solution is correct. The commenting is appropriate. If relevant, it runs "quickly."

Since we may review challenging parts of the homeworks in class on Wednesday, late work submitted within one week will count for 80%; notify the TAs to 'pull' it when it is complete. You must complete (make a concerted effort) on eight of the nine assignments to pass the course. (This includes pass/fail students.)

Final Projects (25%). Working in pairs, students will ask a simple policy question. To answer it they will identify at least two disjoint data sources, merge them, perform a simple but correct statistical analysis and create a simple (but possibly dynamic) dashboard to illustrate this. See final for full details and due dates. A proposal including a dataset and a question is due in Week 5 (October 31).

Participation (10%). I hope that you will actively use the discussion board. Asking questions in a public forum means that I and the TAs do not have to answer the same questions repeatedly. It gives your classmates an opportunity to answer the questions. Asking good questions is an important and demanding skill. You will receive credit both for questions that demonstrate effort, as well as for answering your classmates.

Curve. I expect to follow the standard Harris curve: 1/8 A, 1/4 A-, 1/4 B+, 1/4 B, 1/8 B-. If I feel that it better maps the effort of the class, I may instead adopt 1/6 A, 1/3 A-, 1/6 B+, 1/6 B, 1/6 B-, which is more generous almost everywhere.

Plagiarism policy. Writing code is substantially different from writing essays: it is standard practice to find individual functions or google things that don't work, and copy a line or two from the manual or stack overflow. I encourage you to discuss general strategies for solving problems with your classmates and friends. Questions and answers on the discussion board will naturally include code. However – you should never ask to see another's solutions, and you absolutely should not copy code from your classmates. No one but you should type your code. If you find more than a single line/method, you should attribute the source in your comments.

Where to Work / How to Compute

Part of the overhead to doing computation is getting the software running. Students are therefore encouraged to install the software required for the class on their personal laptops. This will enable them to straightforwardly continue using the skills that they develop, when the quarter is over.

You will need the command line (cygwin on PCs, Terminal on Mac), python (I will only support the Anaconda distribution), and a text editor (Atom). Installation instructions are posted for macs and PCs. (If you're using Linux, you're probably not in this class.) This installation is the entire first week's homework, but it will also be used in the first week's classes! At a minimum, you should have Atom (and cygwin, for Windows) installed *before the first class.* We will hold multiple lab sessions in the first week, when will provide direct computing support. You **must** get the help you need at that point – we won't provide this support for the entire quarter.

Bring your laptop to every class – short demos will be an important part of the lectures.

Weekly Schedule

Week 1: Welcome to the Course – and to the Command line! The first class is the only one for which there will be no live demos. The homework assignment requires that your installation is complete. There will be an additional TA session in the first week (on Tuesday) to assist with this. **It is absolutely required that your installation work. You MUST come to one of the three TA sessions to get any help or, if none of the three are possible, contact me or the TAs.** I understand that computer mishaps happen, but one-on-one installation tech support after the first week is at the discretion of the TAs.

1. Welcome to the course: expectations and case studies of effective use of data in transforming public policy decisions. Navigating the command line.
2. Writing basic scripts with an editor (Atom). Running scripts from the command line. Getting basic answers to your questions, with bash. Uploading them with Git.

- **Slides**: Welcome, bash, Git.
- **Readings**:
  - Git: Hello World, GitHub Guides.
  - Shell: Learning the Shell, William E. Shotts, Jr, parts 2, 3, 5, and 6.
  - Data and Technology in Government: Innovative State (Aneesh Chopra), and The Responsive City (Stephen Goldsmith and Susan Crawford).
- **Assignment**: Ensure that your installation is complete by running this script. Command line fu: answer some basic questions about crime and salaries in Chicago. Create a student GitHub account, and upload the first week's work.

Week 2: Introduction to Python.

1. First steps with python: data types and operations A first shell script. Syntax and semantics of the language; comments. Introduction of the standard data types and operations: `int`, `float`, `string`, `dict`, `list`, etc.
2. Basic formal logic: if, not, and, & or. Control statements (`if`, `else`, `break`, `continue`) and iterating with `for` and `while` loops.

- **Slides**: Python starter/scripts, Simple Program Notebook, Variables and Types Notebook, Control Statements.
- **Readings**: Think Python (Downey), Chapters 1, 2, 10, and 11. For an alternative take, consult chapters 1-5 of the official Python Tutorial.
- **Assignments**: Simple algorithmic problem solving for several Project Euler type problems. Drawing with turtles.

Week 3: List comprehension; functions, classes, and libraries.

1. List Comprehension.
2. Functions: parameters, defaults, and return values. Scope. Classes and libraries: imports and member functions.

- **Slides**: List comprehension, file formats, Functions, Classes and Modules.
- **Readings**: Think Python chapters 3 and 5-8.
- **Assignment**: More Project Euler type problems.

Week 4: Scripts and an example from Athens.

1. Scripts revisited: programs, arguments, and reusable code. Fixing your code: debugging and asking questions.
2. A large-scale example: Athenian Taxes.
3. Complexity (time permitting): big-O notation, computability, and pre-computation.

- **Slides**: debugging, and complexity.
- **Readings**:
  - Python: official documentation for argparse, Think Python Chapter 20 on debugging and 21 on complexity.
  - Asking for help. Eric Steven Raymond wrote an important (though somewhat snarky) piece on How To Ask Questions The Smart Way, to get people to respond with the answer you need. Thinking carefully about your question will often bring you to the answer!
- **Assignments**: Solve Tic Tac Toe. Students will work in pairs to implement the computer's strategy for tic tac toe. Assignment description and repositories.

Week 5: Pandas introduction and file formats. Data Visualization with Python: matplotlib and pandas

1. Pandas data types. Data frames, series. Basic exploration, slicing and plots.
2. Reading basic files. Complex data imports: csv and json files.

- **Slides**: Introduction to Pandas, Worked Examples.
- **Readings**
  - Pandas: Official Documentation, with tutorials. I think Greg Reda's is easier to understand for new users. Once you've read that, Tom Augspurger's Modern Pandas (with Jupyter notebooks) gives a bit more detail.

  - Matplotlib Beginner's Guide and Python for Data Analysis, Ch. 8 (McKinney).
- **Assignments**: Pandas and python. Problem statement here.

Week 6: Reading and Scraping Static Websites; Querying APIs.

1. Reading and scraping html. Beautiful Soup. HTTP requests. School data.
2. RESTful APIs and hidden APIs: Census, weather, and health.
3. The Internet and the world wide web (time-permitting).

- **Slides**: The Internet, the Web, and HTML, APIs and Scraping.
- **Readings**:
  - The Internet, explained and 40 maps that explain the internet, by Timothy B. Lee at Vox (not Tim Berners-Lee, the inventor of the web!).
  - Beautiful Soup.
- **Assignments**: Assemble data from the Virginia elections site.

Week 7: Relational Databases.

1. Relational databases and Structured Query Language (SQL). Selecting. Data types. Pandas integration.
2. Aggregation in SQL and Pandas: `group by`, `order by`, `limit`, `max`, `avg`, etc.

- **Slides**: SQL, see also the examples directory.
- **Readings**: SQL Cookbook Chapters 1-3 and SQLite Tutorial. The psycopg2 basic model usage.
- **Assignments**: query data on the American Time Use Survey. Create a salaries database. Plot through pandas.

Week 8: Statistical Tools and Dashboards.

1. Scipy statistical tests. Statsmodels – regressions from data
2. Dashboards with Plotly.

- **Resources**: Plotly tutorial, and scipy and statsmodels documentation.

Week 9: Large-Scale Example – Weather and Crime.

Week 10: Geographic Information Systems (Time Permitting).

1. Introduction to GIS in GeoPandas. Shapefiles and data – what's available? Making a map: projections and coordinate reference systems.
2. Spatial joins and aggregation.

- **Readings**: GeoPandas and the shapely users manual.
- **Assignment**: finish your final project!!!