# Welcome: To Programming and to Python!

Jamie Saxon

Introduction to Programming for Public Policy

September 19, 2016

# Why Learn to Code?

1. Technology powers the modern world.
   - Gain currency with technology so you can help govern it.
   - What's at stake when services go wrong (VA, healthcare.gov).
   - Understand the potential of algorithms to improve policy.
2. Expand your own toolset.
   - Find, manipulate, and share data to get answers and promote solutions.

# This is an Amazing Moment to Learn

1. Governments are getting on board...
   - 'One size fits all' bureaucracy doesn't cut it.
   - Modern interface for services.
   - Target interventions (money) where it's most needed.
2. And they're learning to share their data.
   - Most states have some data portal presence; many are very good.
3. Software is easier and more powerful than ever.
   - Mapping, internet, etc.: making awesome stuff has never been easier.

# How the Class Is Structured

# What We Will and Won't Cover

- Thinking algorithmically with python.
  - Building blocks of code from the ground up.
- First-pass of low-level tools: the command line.
- Fundamentals of databases and the web.

---

- Higher-level analysis 'recipes.'
  - Build your own projects from large, free components.
  - Wrangle data to get and share information, and create solutions.

---

- However: not a management or policy course.

# What We Will and Won't Cover

- Thinking algorithmically with python.
  - Building blocks of code from the ground up.
- First-pass of low-level tools: the command line.
- Fundamentals of databases and the web.

≈5 weeks

---

- Higher-level analysis 'recipes.'
  - Build your own projects from large, free [illegible]
  - Wrangle data to get and share information, and create solutions.

≈5 weeks

---

- However: not a management or policy course.

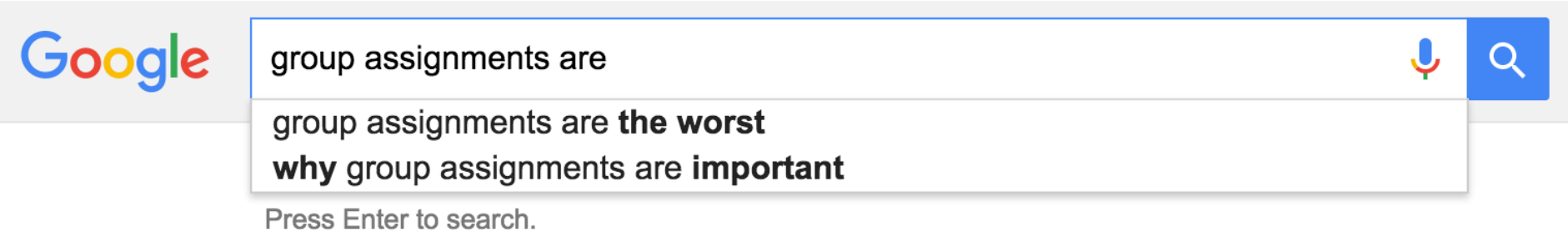
# Assignments

- Weekly assignments posted on the class **Github site**.
- You will also submit them through Github.
  - There are detailed instructions in this week's homework, and we'll cover this in more detail, next week.
  - But if you have trouble with Github, speak up fast!
- Collaborative, large scale final project for exam. I will provide templates, and you will propose the project.

# Yes, some group work.

**Real projects require collaboration.**

- Open source software is an incredibly important...
  - Linux runs your phone; Apache is the most common web server.
- Coders use Github to share work. Using Github is a skill.
  - You don't need to be in the same place, or even talk to your partner.
- Github shows who wrote or modified every line of code.
- In some cases, the 'roles' will be very well definied.



# Additional Resources

- TA: XXX YYY (xyyy@uchicago – but contact with Piazza!).
- XXX and I will host a 'clinic' (office hours) for 2-3 hours, starting from 16h on Thursdays in the Cathey Learning center.
- Use the class **Piazza** discussion board.

- New class: I will regularly ask for feedback. **Please** tell me (anonymously, if you like) if a lecture or assignment is confusing, or if you're frustrated.

# Coding for Public Policy

- 311 System of Boston: improve constitutent interactions and services.
- Mapping Trees in New York: making something beautiful.
- Prioritizing building inspections in New York: public safety.
- Policing in Charlotte/Mecklenberg: intervening with 'at risk' officers.
- Poverty by Census tract: studying and understanding the world.

# Approaching the Command Line

- Please open Terminal (Mac) or Cygwin (Windows).
- If you have not yet installed this (Windows), you can use **tmpnb.org**.
  - This is not the real McCoy, and you will need cygwin very soon!

# The Fundamental Commands

- **pwd**: print working directory
- **cd**: change directory
- **mkdir**: create a directory
- **rm(dir)**: remove a file (directory)
- **ls**: list (files and folders)
- **mv**: move or rename a file
- **chmod**: change permissions on a file

- **ssh/scp**: secure connections (not in this course)

# Notes on the Directory Structure

- **'..'**: means 'backwards one directory'
- **' '**: means 'my home directory'
- **'/'**: is 'root'

- : is a 'wildcard' (match anything or nothing)

- On **cygwin**, hard drive that you're accustomed to is 'hidden' at /cygdrive/c/

# Editing Text and Writing Code: Vim or Atom

- A computer executes your code by 'interpreting it' or runnng a pre-compiled 'binary.'
  - Interpreters: are computer programs that follow your instructions real-time
  - Compilers translate what you write into something the computer understands 'intuitively' (ones and zeros = binary).
- Either way, your code cannot have any extraneous characters.
  - So, needless to say (?), Microsoft Word won't cut it.

- I propose to use **Atom** (gui) or **vim**: Vi IMproved (vi = visual).
  - The advantage of 'vi' is that it's all from the command line.
  - Emacs (also command line) has many partisans, as well.
- In some cases we'll use Jupyter.


python
TM

# What is Python?

**Python is a popular interpreted, high-level, dynamic programming language with automatic memory management.**

**Interpreted**: computer 'runs your instructions,' so you can:

- Run **interactively**: execute one line of code at a time.
- Or **script**: write down and save all of your commands.
- Opposed to languages that are '**compiled**' into a 'machine readable' executable (of 1's and 0's).

**Why Python?** Python hides a lot of the complexity from you.

- You don't have to worry about moving bits (1s and 0s) around: it is **high-level** and takes care of **memory management**.
- It is **dynamic**: it figures out the **types** of variables at 'run-time.'

# Creating a First Script

Create a file `hello_world.py`, using vim Atom, TextEdit, etc.

Write `print("hello world")` in this file and save it.

Navigate to the directory that holds that file:
`cd /Users/jsaxon/Documents/...`

Type: `python hello_world.py`

- To go further, we need the rules and building blocks of Python...

Through the next two lecture, we'll discuss:

**Control**

**Types**

**Assignment, Operators, and Methods**

Some of them will be mixed together.

Launch python on your computer, or open a jupyter notebook:

**Command line: 'python' (must be python 3).**

**Local Jupyter: Anaconda Navigator**

**Online Jupyter: tmpnb.org or try.jupter.org**

Click 'New' in the upper right corner, then Python 3.