

ECOG 315 / ECON 181, Summer 2025
Advanced Research Methods and Statistical Programming
Week 2 Lecture Slides

Matthew N. White

Howard University

June 6, 2025

Week 2 Administrivia

- ▶ If you did not successfully issue a PR for your survey, please do that
 - ▶ Luke
 - ▶ Clinton
 - ▶ Phillippe?
 - ▶ Alejandra
- ▶ If you aren't on Discord, please accept invite
- ▶ If you are on Discord, please keep tabs on it; only means of reaching you

Actually Working with Python: Spyder

- ▶ You **can** write and edit Python code in any text editor
- ▶ And then run Python code from a command line with `python`
- ▶ Using an **interactive development environment** (IDE) just make things nicer
- ▶ Open up Spyder (e.g. from Windows start menu)
- ▶ There are a lot of panes; I close all of them except console and editor
- ▶ Can open them later from View menu, Panes option
- ▶ Console pane: live Python environment, can run code commands one by one
- ▶ Editor pane: file editor; write scripts / programs, run with green arrow in toolbar

How to Learn Python (Outside of Class)

- ▶ We will give you hands-on instruction with Python, don't worry
- ▶ If you want to learn on your own, I recommend Kevin Sheppard's notes
- ▶ There's a copy in `/materials/setup/KevinSheppardPython.pdf`
- ▶ It's from four years ago; current Python versions are 3.10 to 3.13
- ▶ Recommended reading order: Ch 1, 2, 4, 9, 10, 12, 18, 3, 6, 7, 5, 11, 15, 29
- ▶ Data operations: Ch 8, 16, 14, 17, 23
- ▶ More math stuff: Ch 19, 20, 21, 22
- ▶ General coding: Ch 13, 24, 25, 26, 28

Crash Course in Brain Surgery

- ▶ The only way to learn to program in any language is to do it
- ▶ Don't just read things: jump in and experiment with code
- ▶ Don't be afraid to make mistakes or encounter difficulty
- ▶ I am intentionally “throwing you into the deep end”
- ▶ In repo: `/code/ProcessSurveys.py` is code example to get started
- ▶ Will go over here, but I encourage you to play with and edit it

Python Programming Mentality (1/2)

- ▶ Think of Python workspace as a nested series of boxes (or **scopes**)
- ▶ Each function, variable etc lives in some box, referenced by name
- ▶ You can make new boxes inside of existing boxes
- ▶ Stuff in box A usually can't see into box B
- ▶ If you refer to a name in a scope, Python looks for it in that box
- ▶ If that fails, it looks for that name in the box that contains the box
- ▶ And so on until it gets to the top-level scope– there's no more “up”!
- ▶ If it can't find the name there, then Python will throw an error

Python Programming Mentality (2/2)

- ▶ Top level scope is called `--main--`
- ▶ A new scope is created every time you run a function– local “box”
- ▶ Scopes are discarded when finished running code
- ▶ Python has automatic memory allocation and garbage collection
- ▶ Never need to declare memory space in advance
- ▶ Memory freed automatically when there are no “references” to object
- ▶ Can manually delete objects, but that’s rarely needed

Basic Data Structure: List

- ▶ Workhorse data structure in Python is a `list`
- ▶ Exactly what it means in English: ordered list with repetition allowed
- ▶ Denote with brackets and commas: `A = [0, 3, 2, -17, 1]`
- ▶ **Anything** can go in a list, no matter what else is in the list
- ▶ This is fine: `B = [3.14, itertools, "oh hi Mark", all_of_Hamlet]`
- ▶ You can have a list as an element of a list: `C = [A, B]`
- ▶ Addition on lists is concatenation: `D = A + B`

Accessing Items in a List

- ▶ You can access one item of a list by its index with brackets: `A[3]`
- ▶ Python uses zero-indexing: the first element of a list has index zero
- ▶ You can look at **slices** of lists using colon notation: `A[1:4]`
- ▶ Important! Index actually points *between* elements
 - ▶ So `A[1:4]` says “elements of `A` between index 1 and index 4”
 - ▶ Those are the second, third, and fourth elements in human terms
- ▶ Can take all elements after or before an index: `A[2:]` or `A[:3]`
- ▶ Can index from *end* of list with negative numbers

Importing Stuff into a Workspace

- ▶ Python has a lot of stuff, but it's not “all there” by default
- ▶ When you start a Python kernel, only the basics are loaded in
- ▶ Can bring in additional material with the `import` command
- ▶ rename things as you `import` them, often for lazy typing
- ▶ Important: differentiate b/w installing package vs importing it
 - ▶ **Installed:** package exists on your computer in Python environment
 - ▶ **Imported:** stuff has been “summoned” into current workspace
- ▶ Can also import your own files from the same directory

Example Python Code: Processing Student Surveys

- ▶ Best way to teach you is to walk you through example code
- ▶ Sync your fork with the course repo, and pull down from remote
- ▶ Open `/code/ProcessSurveys.py` in Spyder or other editor
- ▶ Has functions to read in my survey, and all of yours
- ▶ And then to produce histograms of responses
- ▶ Written to show you a bunch of commonly useful things

Introduction to Jupyter Notebooks

- ▶ Communicating scientific ideas is sometimes aided by reader interaction
- ▶ **Jupyter notebooks** provide a way to integrate text, math, and code
- ▶ Sync your fork with the course repo, and pull down from remote
- ▶ In Anaconda prompt, type `jupyter notebook`
- ▶ Navigate to your fork on your computer, then go to `/code/`
- ▶ Open `ExampleNotebook.ipynb`
- ▶ Cells can be run individually with `Shift+Enter`

Anaconda and Virtual Environments

- ▶ There are a lot of Python packages, and you will have multiple projects
- ▶ Project A requires CoolPackage v0.8 or higher; Project B requires v0.7 or lower
- ▶ Not a problem! conda can manage virtual Python environments
- ▶ Can have multiple collections/configurations of packages on your computer
- ▶ Manage each separately, switch between them with a single command
- ▶ We probably won't use this much, but will install more packages
- ▶ Open up Anaconda Prompt (from Windows start menu, e.g.); terminal will open

Installing New Packages

- ▶ Example: installing HARK, Econ-ARK's primary software package
- ▶ In Anaconda prompt, type `pip install econ-ark`, accept
- ▶ `pip` is the most common **package manager**, draws on Python Package Index
- ▶ `conda` is also a package manager, but **slightly** fussier
- ▶ HARK is now available for use on your computer
- ▶ But we won't be working with it yet

Step One: Pitching a Research Project

- ▶ On **Tuesday, June 10, at 1:30pm**, you will practice your pitch
- ▶ On **Friday, June 13, at ????**, you will give your pitch to ???
- ▶ There are 12 groups, and your pitch is 5-10 minutes; 150 minute time slot!
- ▶ Tuesday: really rapid advice from Alan and me; written feedback later
- ▶ Friday: substantive feedback & questions from wider audience
- ▶ If you want slides on Tues, you must submit them **as a PR by 1pm Tuesday**
- ▶ Please put them in PDF form (export to PDF); I do not have PowerPoint

Advice for Giving a Research Pitch

- ▶ You are excited about this idea, and it is interesting
- ▶ If you don't care, then no one else will care; you should care a lot
- ▶ Speak loudly and clearly; take your time and enunciate your words
- ▶ Counterbalance: you're on a tight schedule, so don't dawdle
- ▶ Be mindful of what the audience knows and what they don't know
- ▶ **Briefly** explain key concepts before going too far in
- ▶ This is just a **pitch**, so you don't need to have all the answers
- ▶ Slides are a memory tool and/or visual guide, not a script

Structuring Your Research Pitch (1/3)

- ▶ Order of information is critical in all contexts
- ▶ First: what are you investigating, in 1-2 sentences?
- ▶ No detail, no background, just tell them what the subject area is
- ▶ Lets the audience get their mind in the right mode
- ▶ **Then** provide appropriate background information
- ▶ Specific policy change, important history, etc
- ▶ Then state your research question **very specifically**

Structuring Your Research Pitch (2/3)

- ▶ After telling people what the question is, convince them it matters
- ▶ Why do we want to know the answer? What would someone do if they knew?
- ▶ How would human welfare improve if they took those actions?
- ▶ You can skip this if it's glaringly obvious, like “keep people alive”
- ▶ **Briefly** tell them what we already know: prior research
- ▶ This is **not** a literature review; just a summary / overview
- ▶ What's your angle? Why is this new, as far as you know?

Structuring Your Research Pitch (3/3)

- ▶ Try to provide an outline of **how** you're going to answer your Q
- ▶ What data do you **need**– what is **required** to be in the dataset to do this?
- ▶ Have you identified a dataset that will do this? Tell us about it!
- ▶ Do you have some ideas for where to look? Only feasible if you can get the data!
- ▶ **Briefly** explain your current thoughts on econometric method: what will you **do**?
- ▶ At most **one or two** equations / econometric specifications
- ▶ What are your current concerns about this strategy? Why might it not be valid?
- ▶ Don't worry if you have none, the audience will tell you their concerns!