

QSS20: Modern Statistical Computing

Unit 02: Catch-up and Python Basics

Agenda

- ▶ Intros
- ▶ Recap course
- ▶ Python basics
 - ▶ Intro and variables
 - ▶ Lists and basic list comprehension
 - ▶ 'numpy' arrays
- ▶ Work session







Agenda

- ▶ **Intros**
- ▶ Recap course
- ▶ Python basics
 - ▶ Intro and variables
 - ▶ Lists and basic list comprehension
 - ▶ 'numpy' arrays
- ▶ Work session

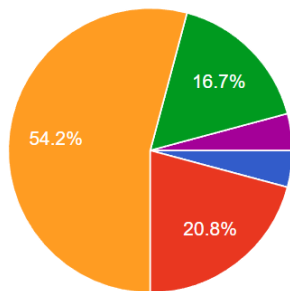
Meet and introduce your neighbor!

- ▶ Preferred name
- ▶ Favorite class at Dartmouth so far and why?
- ▶ If you could have any data source at your disposal, what would it be and what's a question you would ask?

A bit about me

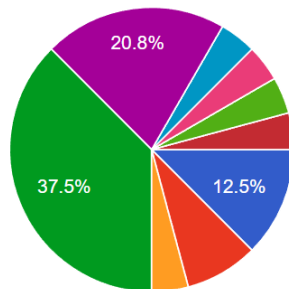
Where	What	Languages
	BA in Cultural Anthropology	None
	PhD in Sociology, emphasizing organizations and education	
Berkeley D-Lab  <small>GEORGETOWN UNIVERSITY</small> MASSIVE DATA INSTITUTE <small>McCourt School of Public Policy</small>	Organizer and trainer in computational methods	 python™
	Postdoctoral Fellow	 python™

Experience with Python (24 replies)



- No experience
- Beginner (I can do a few operations)
- Familiar (I have developed at least one project)
- Intermediate (Multiple quarters of experience)
- Advanced (I would feel comfortable teaching it to others)

Policy areas of interest



- International relations/security policy
- Labor markets/employment
- Education policy
- Health policy
- Climate science/the environment
- Climate justice policy
- Housing policy
- Financial policy & market insights (also Health policy)
- Macro market trends and reactionary...

Some FAQs from the intro survey

- ▶ **Q:** how much prior Python/SQL/data science knowledge is needed to do well in the course?
 - ▶ **A:** **None!** The course is designed to take you from no previous exposure to these languages to be able to complete and do well on all assignments (there are no exams; only project-based psets and a final project). We're also providing extra support by starting with basics and group tutoring to get everyone to same place.
- ▶ **Q:** are there materials you recommend for extra self study?
 - ▶ **A:** the course is designed to be self-contained through the combination of slides + in-class activities + DataCamp + psets. But if you need extra help with certain concepts after going through the assigned materials, we can provide extra content-specific online resources.

Where we are

- ▶ Intros
- ▶ **Recap course**
- ▶ Python basics
 - ▶ Intro and variables
 - ▶ Lists and basic list comprehension
 - ▶ 'numpy' arrays
- ▶ Work session

Recap

What do you remember about the course structure?

Close your laptops for a moment and discuss with your neighbor.

Content of QSS20: Modern Statistical Computing

Topics:

- ▶ Data wrangling and viz
- ▶ APIs and web-scraping
- ▶ Text as data
- ▶ Supervised machine learning

Workflow tools:

- ▶ Git/GitHub
- ▶ Command line (shell)
- ▶ LaTeX

Components:

- ▶ DataCamp modules
- ▶ Class activities
- ▶ Problem sets
- ▶ Final project

Course TAs and group tutor

TA: Ramsey Ash

- ▶ Ramsey took QSS20 in winter 2022 and was group tutor last quarter
- ▶ Will be present during most of classes as tech support/coding resource
- ▶ Will help with grading and be a resource via OH & Piazza

TA: You-Chi (Eunice) Liu

- ▶ Eunice took QSS20 in spring 2021
- ▶ Taking a 3A class, so go visit her in OH!
- ▶ In addition to grading, resource via OH & Piazza

Group tutor: Eleanor Sullivan

- ▶ Eleanor took QSS20 in winter 2022
- ▶ Leads group tutoring sessions, may also be a resource via Piazza

Group tutoring sessions

- ▶ To support all students, especially those with less coding background
- ▶ Thanks to Peer Tutoring Program:
<https://students.dartmouth.edu/academic-skills/Peer-Tutoring-Program/about-peer-tutoring-program>
- ▶ Tutoring session times **starting next Sunday, 01/15**: Sun 8-9pm; Mon 7-8pm; Thurs 9-10pm (these may change)
- ▶ Sign up for tutoring here (also on course page): <https://grouptutoring.dartmouth.edu/terms/23W/groups/12210>

Office hours

- ▶ **Required** that you attend office hours or group tutoring at least once
- ▶ **My office hours**
 - ▶ **Two slots:** Mondays 2:15-3:15 pm; Wednesdays 2:15-3:15 pm
 - ▶ **Location:** Silsby 103 (main floor)
 - ▶ In-person drop-ins and groups are welcome!
 - ▶ But if you want to meet privately and/or via Zoom, you can sign up in advance (**no later than the midnight before**) via Calendly (I'll attach to online course schedule):
<https://calendly.com/jaren-haber-qss>
 - ▶ When signing up, please let me know if you want privacy and indicate your preferred format:
 1. In person (at my office)
 2. Zoom
 - ▶ If my door is closed during office hours, that means I am in a private meeting. Please wait in the hall.
- ▶ Check Piazza for TA office hours

What to expect in an average class





Time window	What
3:30-3:40	Warm-up & recap
3:40-4:10	Slides; DataCamp questions
4:10-4:15	Break into small groups
4:15-5:00	Work with group on class activity; Ramsey and I circulate around
5:00-5:15	Reconvene as a group and go over questions; outline any prep for next class
5:15-5:20	Anonymous sticky notes with lesson & question

Might deviate if we have visitors/guest speakers, e.g., related to final project/SIP

DataCamp: make sure to join via our specific course page so assignments show up

See bottom of course syllabus main page for join link.

Post to datacamp on Piazza if you need access via a different email.

TITLE ▾	ASSIGNEES ▾	STATUS	DUE BY ▴	C ▾	A ▾	CR ▾	DETAILS
 Introduction to Python Python Lists Chapter	Organization	DUE SOON	Jan 9, 15:30 EST	19	27	70%	View
 Introduction to Python Functions and Packages Chapter	Organization	DUE SOON	Jan 9, 15:30 EST	24	27	88%	View
 GitHub Concepts Introduction to GitHub Chapter	Organization	DUE SOON	Jan 11, 15:30 EST	4	26	15%	View
 Introduction to Shell Manipulating files and directories Chapter	Organization	DUE SOON	Jan 11, 15:30 EST	4	27	14%	View

Meant as auxiliary tool/playing a minor role so that you're prepared for in-class activities and so we don't need to review basic syntax. So graded on completion-only basis and only 5% of grade, but if you'd prefer to skip, can reapportion the 5% to the first problem set

Five problem sets

- ▶ Problem set one will be on GitHub in next few days and is due Sunday, 01.15
- ▶ **Others:** see online course schedule
- ▶ **For each:**
 - ▶ Start well in advance (at least 3-4 days) and space out the parts (Pset 1 should be largely review from COSC 1 and the initial DataCamp modules)
 - ▶ May devote some class time pre deadline to work on the pset/answering questions

Ways to get help on problem set

- ▶ **Office hours with me and the TAs**
- ▶ **Post on Piazza and TA or I will answer within 24 hours during weekdays; by Sunday night if weekend**

Structure of final project

- ▶ **Options:** Pick a data source from from [course syllabus](#)
- ▶ **Milestone 1:** memo or plan for what question you'll ask and analyses you'll run
- ▶ **Milestone 2:** set up your repository and start coding
- ▶ **Final outputs (see course website for more details):**
 - ▶ Final presentation (done in Beamer; LaTeX-based powerpoint software)
 - ▶ Short 10-page report (done in LaTeX)
 - ▶ Github repo and readme with all code to reproduce analyses

Q&A

Any questions about course structure and expectations?

Where we are

- ▶ Intros
- ▶ Recap course
- ▶ **Python basics**
 - ▶ **Intro and variables**
 - ▶ Lists and basic list comprehension
 - ▶ 'numpy' arrays
- ▶ Work session

Why Python?

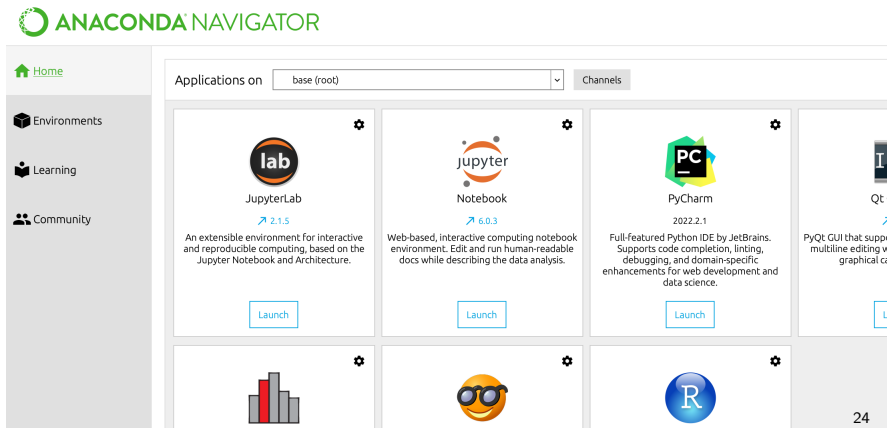
- ▶ Four programming languages used to varying degrees in data science: Python; R; STATA; and SAS
- ▶ Python: has superior libraries for machine learning, natural language processing, and other techniques previewed in this course and later in the sequence
- ▶ Open-source which means free to download and use

Terminology/ways of interacting with code in Python

- ▶ **Jupyter notebooks:** Focus in most of this class (e.g., `this_is_my_script.ipynb`)
 - ▶ Where to work with this: **Local Python installation**; JHub; Google colab
 - ▶ Great setup for working with smaller datasets; testing code; integrating code, text, and figures
- ▶ **Scripts:** useful for final projects (e.g., `this_is_my_script.py`)
 - ▶ Can run from command line or IDE (integrated development environment): place to write and test code (e.g., Spyder; PyCharm; VS Code)
 - ▶ Essential when working with larger datasets; “production-ready” code
- ▶ Useful in either case: text editors to inspect code and data in various file formats (e.g., .csv, .json, pkl, .R)
 - ▶ Examples: Sublime text; Vim; Atom

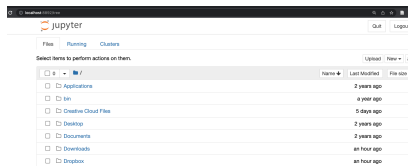
How do I open a jupyter notebook? Point and click way

1. Make sure Anaconda is installed (see course website)
2. Go to Applications and click on the Anaconda-Navigator icon
3. Something like this should pop up (be patient). Click on the **Launch** icon under Jupyter Notebook



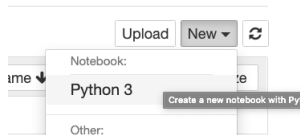
How do I open a jupyter notebook? Point and click way

4. That should open up a browser window or tab with your full set of directories/files



5. Then:

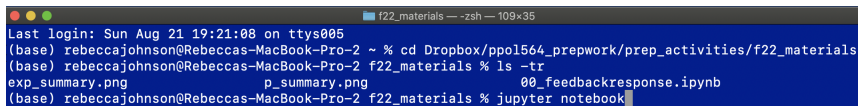
- To create a new notebook, click New in right hand corner



- If opening an existing notebook, navigate to folder where it's stored, click on the file, and notebook should open in a new browser tab

How do I open a jupyter notebook? Terminal way

1. Navigate to folder where you want to create a new or open an existing notebook
2. Type `jupyter notebook` into your terminal (Mac) or terminal emulator (Windows)

A screenshot of a terminal window with a dark blue background and white text. The window title bar shows 'f22_materials' and '-zsh' with a size of '109x35'. The terminal output shows the user's last login, the current directory being set to a path in Dropbox, and the files in the directory. The command 'jupyter notebook' is entered at the prompt.

```
f22_materials — -zsh — 109x35
Last login: Sun Aug 21 19:21:08 on ttys005
(base) rebeccajohnson@Rebeccas-MacBook-Pro-2 ~ % cd Dropbox/ppol564_prepwork/prep_activities/f22_materials
(base) rebeccajohnson@Rebeccas-MacBook-Pro-2 f22_materials % ls -tr
exp_summary.png          p_summary.png          00_feedbackresponse.ipynb
(base) rebeccajohnson@Rebeccas-MacBook-Pro-2 f22_materials % jupyter notebook
```

3. If on Windows and run into issues, see here:

<https://stackoverflow.com/questions/41034866/running-jupyter-via-command-line-on-windows>

How do I edit, save, and compile a notebook?

Break for screensharing example

Variables/objects and types: creating variables

Can follow along here: https://github.com/jhaber-zz/QSS20_public/blob/main/activities/00_introclass.ipynb

```
1 my_name = "Rebecca"
2 my_birth_month = 9
3 my_birth_day = 19
4 my_birth_year = 1988
5 frac_of_month = 19/30
6 female = True
```

Things to note:

- ▶ No spaces or dots in name of object (can use underscores though)
 - ▶ Why no dots? In Python, dots are meaningful! they represent the methods and attributes that we'll discuss in later slides today
- ▶ = is what's called the *assignment operator*; when you learn R, you'll see that you typically use <- instead
- ▶ I created objects of different types; the quotes denote Rebecca is a string; the lack of quotes for birth month, etc. denote integer (or float)

Variables/objects and types: checking types

```
9] ## check types
print(type(my_name))
print(type(my_birth_day))
print(type(frac_of_month))
print(type(female))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
<class 'bool'>
```

Things to note:

- ▶ **Int versus float:** no decimals versus decimals
- ▶ **String versus boolean:** we told Python that it was boolean by setting it equal to True without quotes; this is encoded in python as 1 = True; 0 = False and we'll see is very useful later for data aggregation/summaries

Variable/objects and types: transforming types

```
## convert between types
### change female from boolean (True or False)
### to integer (1 = True; 0 = False)
female_int = int(female)
print(type(female_int))
print(female_int)
```

```
<class 'int'>
1
```

```
### change birth month from integer
### to float
bmonth_float = float(my_birth_month)
print(bmonth_float)
```

```
9.0
```

Where we are

- ▶ Intros
- ▶ Recap course
- ▶ **Python basics**
 - ▶ Intro and variables
 - ▶ **Lists and basic list comprehension**
 - ▶ 'numpy' arrays
- ▶ Work session

Lists: how to create

```
## create a list
list_new = [9, 19, 1988]
list_existing = [my_birth_month, my_birth_day,
                 my_birth_year]

print(list_new)
print(list_existing)
print(type(list_existing))
print(len(list_existing))

[9, 19, 1988]
[9, 19, 1988]
<class 'list'>
3
```

Things to note:

- ▶ `list_new` I created from scratch; `list_existing` I combined the objects I created earlier in the code
- ▶ Either way, use `[` with commas separating list elements
- ▶ `len` is a built-in function in Python (doesn't require us to import a package) that works with lists in addition to other types of objects

Other things about lists covered in the DataCamp module on lists

- ▶ Methods that operate on lists, using syntax: `name_of_list.method()` like `round`, `max`, `reverse`, etc.
- ▶ Subsetting lists using the syntax: `name_of_list[0:3]`

Basic list comprehension

- ▶ **Goal:** iterate over list elements and do something:
 - ▶ Filter: select a subset of list elements based on some condition
 - ▶ Transform: modify the elements of the list
 - ▶ General: modifies each element in the same way
 - ▶ Conditional: modifies some elements in some way; others in a different way
- ▶ In future week, we may discuss distinctions between using list comprehension for these tasks versus `for` loops (latter used commonly in R and STATA; in Python, you can use list comprehension for almost everything you'd use a `for` loop for and list comprehension has many advantages!)

Example task

Want to convert the list with the three birthday elements—`[9, 19, 1988]`—into a single string: `"09-19-1988"`

General transformation

```
## copy over list to give more informative name
bday_info = list(list_existing)
print(bday_info)

## convert each element to a string
bday_info_string = [str(num) for num in bday_info]
print(bday_info_string)

[9, 19, 1988]
['9', '19', '1988']
```

Breaking this down:

- ▶ `str(num)` is the step that's doing the transformation
- ▶ `for num in bday_info` iterates over each of the three elements in the `bday_info` list
- ▶ `num` is a totally arbitrary placeholder; we could use `i`, `e1`, or whatever; key is that it's the same between the **iteration** and **transformation**

Conditional transformation

What if we want to not just convert each element to string, but add a 0 if the str is one-digit? (So pad the 9 with a 0 as it's converted to a string)?

Conditional transformation

```
## conditional transformation - add a 0 if only 1-char long
bday_info_string_pad = ['0' + num if len(num) == 1
                        else num
                        for num in bday_info_string]
print(bday_info_string_pad)

['09', '19', '1988']
```

Breaking this down:

- ▶ **What stayed the same?** The iterating through elements for `num` in `bday_info`
- ▶ **What changed?** We added a condition using `if` and `else`, and using the built-in `len()` function we covered earlier
 - ▶ If it's a 1-character string, it uses the `+` to paste the string `'0'` onto it
 - ▶ Otherwise, it keeps the string as is

Filtering using list comprehension

- ▶ We can use similar syntax to take a list—in this case, a length-3 list—and filter to a smaller number of elements based on some condition
- ▶ This is usually better coding practice than subsetting using the position of an element in a list since its more robust to lists ordered in certain ways

Filtering using indexing versus list comprehension

```
## filtering
month_day_subset = bday_info_string_pad[0:2]
print(month_day_subset)
month_day_subset_lc = [el for el in bday_info_string_pad
                       if el != '1988']
print(month_day_subset_lc)

['09', '19']
['09', '19']
```

Things to note:

- ▶ **Filtering using indexing:** relies on elements being in a certain order in the list; inclusive on the left hand side (so includes the first element \Rightarrow index 0; second element \Rightarrow index 1) but not inclusive on the right hand side
- ▶ **Filtering using list comprehension:**
 - ▶ Same for `el in list` syntax we saw earlier
 - ▶ Returns `el`: just return the list element without any transformations
 - ▶ Adding an if condition: keeps the element if it's not equal to 1988 (has drawbacks in generalizability we'll discuss in a few slides)

Robustness to reordering

```
### list comprehension more robust to
### reordering
new_list = ['1988', '09', '19']
### indexing gives us wrong answer
print(new_list[0:2])
### list comp. gives us right answer
print([el for el in new_list
       if el != '1988'])

['1988', '09']
['09', '19']
```

Things to note:

- ▶ When the list is reordered, filtering list elements using indexing gives us the wrong answer
- ▶ If our pattern is correct, **filtering using list comprehension** is more robust

Especially powerful when combined with regular expressions (regex) that we'll cover later

```
## example of regex to separate days versus months
### import module
import re
### month pattern is 01...09 or 11 or 12
month_pattern = r'0[1-9]|1[1-2]'
example_date_str = ['09', '30', '01', '12', '11', '19']
### keep element in list if element matches pattern
keep_months = [el for el in example_date_str
                if re.search(month_pattern, el)]
keep_months

['09', '01', '12', '11']
```

From lists to array: background on python modules

- ▶ In previous slide, we used the `import re` statement
- ▶ This imports a module named `re` that has functions (in our case: `re.search()`) that are not automatically built into Python
- ▶ Common modules we'll use:
 - ▶ This week/pset one: `numpy`
 - ▶ Next couple weeks/pset two: `pandas` and ones for plotting

Where we are

- ▶ Intros
- ▶ Recap course
- ▶ **Python basics**
 - ▶ Intro and variables
 - ▶ Lists and basic list comprehension
 - ▶ **'numpy' arrays**
- ▶ Work session

Lists versus numpy arrays

Two main (practical) differences:

1. **Lists can store heterogeneous data types; arrays cannot:** a single list can combine a string, integer, etc; an array must hold a single type of data
2. **For various reasons, arrays are more memory efficient for large computational tasks:** uses less memory to *store* the same data ([explainer on this](#)); in stats or the ML course you may learn about matrix operations and 2-dimensional numpy arrays are good for these

Numpy arrays: creating

```
## import the module
import numpy as np
## convert keep_months to a 1d array
keep_months_arr = np.array(keep_months)
print(type(keep_months_arr))
print(keep_months_arr)
print(keep_months_arr.shape)

<class 'numpy.ndarray'>
['09' '01' '12' '11']
(4,)

## create a 2-d array
months_2d = np.array(['09', '10'],
                      ['September', 'October'])
months_2d

array(['09', '10'],
      ['September', 'October'], dtype='<U9')
```

Things to note:

- **import numpy as np** is called **aliasing**; we use an abbreviation to refer to the package name (np is arbitrary but commonly used)
- **shape** is an **attribute** of arrays; this is different than something like `len()` that's a function
- To create the 2-d array, we use a list of lists

Numpy arrays: filtering/subsetting

Task: how can we pull out the month names (second row; both columns) from the `months_2d` array?

```
## pull out months (second row)
## and all columns
months_2d[1, :]

array(['September', 'October'], dtype='<U9')
```

Wrapping up

Covered:

- ▶ Intros
- ▶ Recap course
- ▶ Python basics
 - ▶ Intro and variables
 - ▶ Lists and basic list comprehension
 - ▶ 'numpy' arrays

Work session

Order of priority:

1. Getting working version of Python installed for first problem set
2. Starting problem set one (short so more doable closer to deadline)
3. Completing DataCamp assignments due this week
4. Creating accounts: Piazza; GitHub; Overleaf