



Lecture 4: Processing Linguistic Data

LING 1340/2340: Data Science for Linguists

Na-Rae Han

Objectives

- ▶ Homework 1: What linguistic datasets did you all process?
- ▶ Tools:
 - ◆ Git and GitHub
 - ◆ Jupyter Notebook
 - ◆ Using DataCamp tutorials

**You should be
taking NOTES!**



First thing to do every class

```
MINGW64:/c/Users/narae/Documents/Data_Science

narae@X1Yoga MINGW64 ~
$ cd Documents/Data_Science/

narae@X1Yoga MINGW64 ~/Documents/Data_Science
$ pwd
/c/Users/narae/Documents/Data_Science

narae@X1Yoga MINGW64 ~/Documents/Data_Science
$ ls
Class-Exercise-Repo/  languages/

narae@X1Yoga MINGW64 ~/Documents/Data_Science
$ ls -la
total 12
drwxr-xr-x 1 narae 197121 0 Jan 10 14:30 ./
drwxr-xr-x 1 narae 197121 0 Jan  8 18:33 ../
drwxr-xr-x 1 narae 197121 0 Jan 10 14:30 Class-Exercise-Repo/
drwxr-xr-x 1 narae 197121 0 Jan  8 18:34 languages/

narae@X1Yoga MINGW64 ~/Documents/Data_Science
$ |
```

```
pwd
cd dir1/dir2
cd ..
cd
ls
ls -la
```

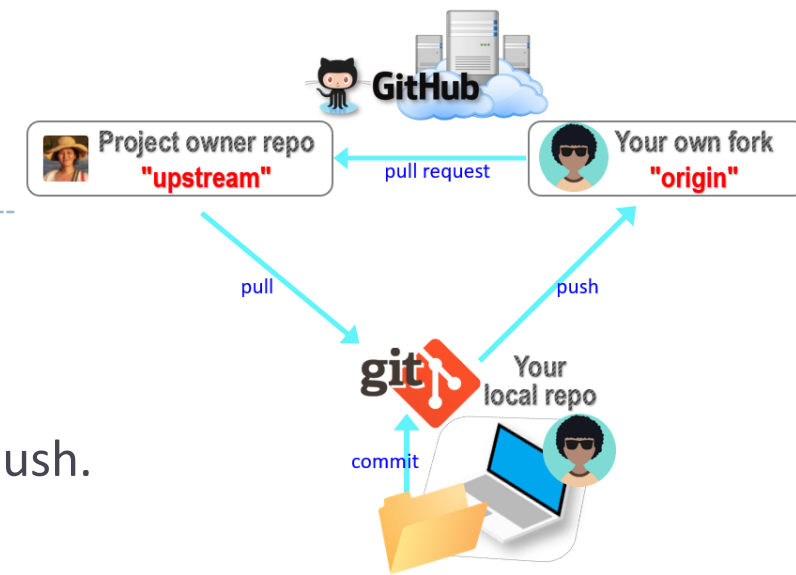
Hit **TAB** for auto-completion.

Up **↑** / Down **↓** arrow to use previous command

Ctrl + c to cancel

Last step: Sync your repos
(Class-Exercise-Repo especially)

Your workflow



1. **Housekeeping:** Check YOUR WORK via "git status".

- ◆ Your local repo is clean: no unsaved/uncommitted work.
- ◆ Your GITHUB fork already has your latest commit: there's nothing to push.

2. **Housekeeping:** Bring in updates from OTHERS.

- ◆ On your **GitHub fork**, check what updates have accumulated in the upstream repo.
- ◆ Through "Sync fork → Compare", make sure those updates don't have conflicts with your fork. Don't press that green "Update Branch" button!
- ◆ Back on **command line**, pull from upstream. Now your local repo is synced with the original repo.
- ◆ Finally, sync your GitHub fork by pushing. The universe is in order now!

3. **Work on your homework, to-do, etc.**

- ◆ *Now* start your homework. Make some commits along the way.
- ◆ Push to your GitHub fork for one last time.
- ◆ Submission time: Create a **pull request**. Make sure your pull request doesn't have conflicts.

HW1: processing pull request, merging

- ▶ With everyone working on their own files/folders, merging is conflict-free:

The screenshot shows a GitHub pull request interface for the repository 'Data-Science-for-Linguists-2023 / HW1-Repo'. The pull request is titled 'Alex HW 1 #9' and is created by 'alocampo'. It shows 2 commits being merged into the 'main' branch. The interface includes navigation tabs for Code, Issues, Pull requests (2), Actions, Projects, Wiki, Security, Insights, and Settings. A comment from 'alocampo' is visible, stating 'No description provided.' Below the comment, a commit history shows 'alocampo added 2 commits 3 days ago', with sub-commits 'set up for assignment' and 'Alex HW1 Submission'. A branch protection rule is highlighted, stating 'Require approval from specific reviewers before merging' and 'This branch has no conflicts with the base branch'. A 'Merge pull request' button is visible at the bottom of the rule section. The right sidebar shows settings for Reviewers, Assignees, Labels, Projects, Milestone, Development, and Notifications.

HW1: sync your HW1-Repo

1. Configure "upstream" remote:

```
git remote add upstream https://github.com/Data-Science-for-Linguists-2023/HW1-Repo.git
```

2. Check your GitHub fork, make sure there are no conflicts with upstream

3. Pull from upstream:

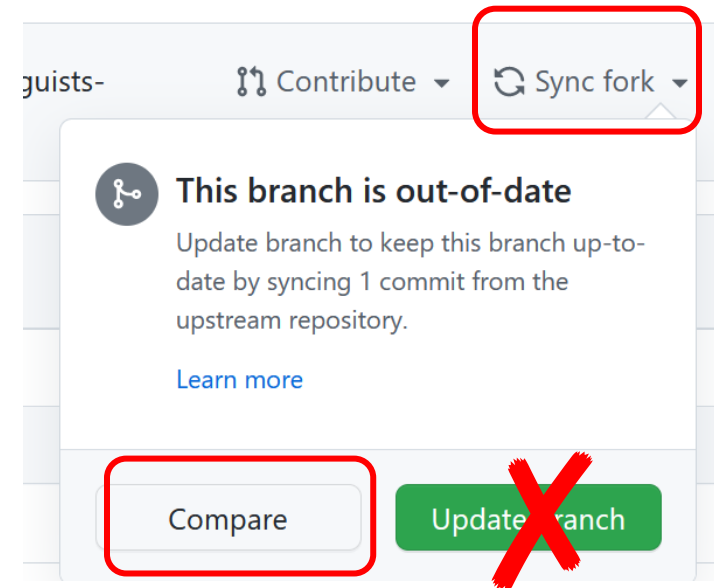
```
git pull upstream main
```

4. Push to your GitHub fork:

```
git push
```

Everyone's repos
are synced.

Now, everyone has
everyone's homework
submission.



HW1: Review

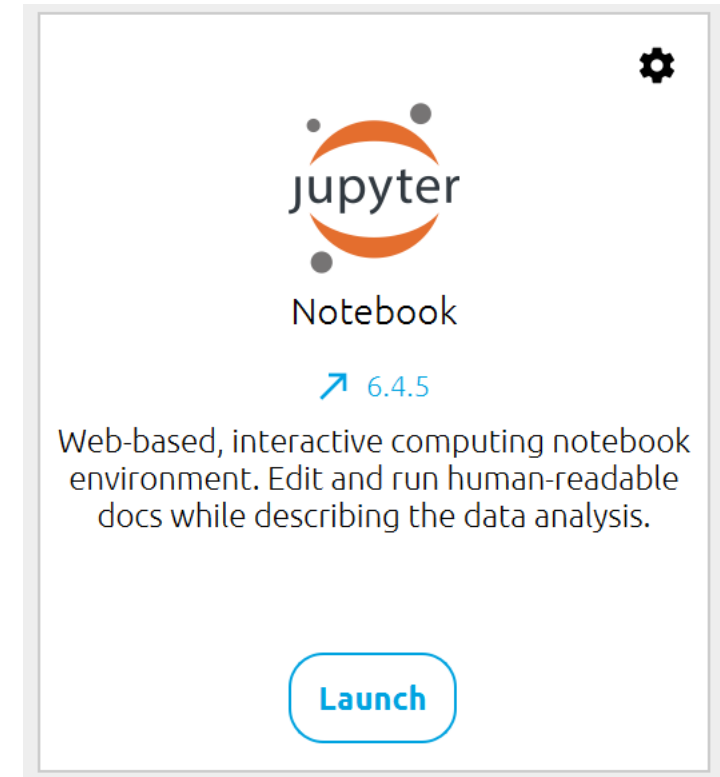
- ▶ What did you all work on?
- ▶ You wish list: what new skills would you like to learn?
- ▶ What is the `.gitignore` file?
- ▶ Why did we exclude data files from Git?
- ▶ What is up with that "your_file_here.txt" blank file? What is `git rm`?
- ▶ Jupyter Notebook: do you like it?

Term Project overview

- ▶ Go over the term project guidelines:
 - ◆ <https://naraehan.github.io/Data-Science-for-Linguists-2023/project>
- ▶ Also talk about what makes a good/successful project, ways to come up with project ideas

Jupyter Notebook

- ▶ Allows you to create and share documents that contain live code cells, output, equations, visualizations and explanatory text.
- ▶ Learn how to use it. Your Python code should be in the Jupyter Notebook format:
 - ◆ `xxxx.ipynb`
- ▶ You can launch it from the command line.
 - ◆ Move into the desired directory, and then execute `jupyter notebook &`
 - ← '&' is not necessary, but it lets you keep using the terminal
 - ◆ If it doesn't work, then edit your system's path variable or just use a shortcut provided by your OS.



Course Group on DataCamp

▶ Video-based, interactive tutorials

The screenshot displays the DataCamp user interface. At the top, the DataCamp logo is on the left, followed by navigation tabs for 'LEARN' and 'WORKSPACE'. A search bar labeled 'Search Catalog' is in the center, and 'My Account' with a dropdown arrow is on the right. A notification bell icon is also present. A dark blue sidebar on the left contains a list of navigation options: My Progress, My Bookmarks, Organizations, Custom Tracks, Career Tracks, Skill Tracks, Courses, Practice, Projects, Assessments, and Live Events. The main content area is titled 'My Progress' and features a welcome message: 'Welcome back, Na-Rae! Reach 250 XP today to continue your streak ...'. To the right of this message is a dark blue box showing a circular progress indicator with '0' and '0 day streak' and '0 XP'. Below this, there are two course cards. The first is 'Introduction to R' under the 'LEARN' category, with a green lightbulb icon, a progress bar showing 'Intro to basics' as the first step, and a clock icon indicating '4 hours to go'. A bright green button below it says 'Keep Making Progress'. The second card is 'Introduction to Python' under the 'PRACTICE' category, with an orange dumbbell icon.

We get FREE access this semester -- all you can learn!
Use **Pitt email address** to sign up.

How to use DataCamp

▶ Topics for the next couple of weeks:

- ◆ numpy library
- ◆ pandas library
- ◆ visualization libraries such as matplotlib

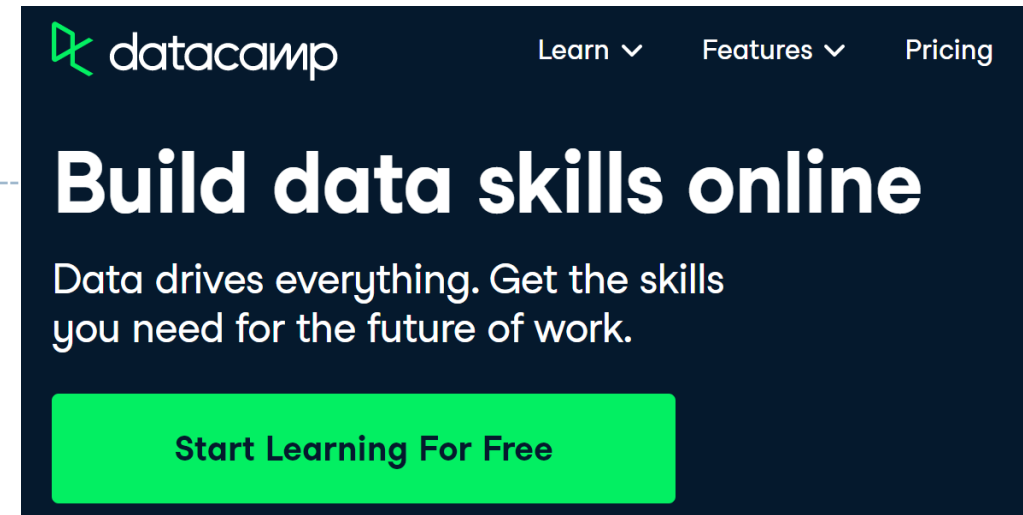
▶ Which video tutorials? Find them on our [Resources page](#):

- (DataCamp) Career Track: Data Scientist with Python (includes all courses below and more) [track]
 - (DataCamp) Introduction to Python for Data Science, Ch.4 NumPy [course] [Ch.4 Numpy]
 - (DataCamp) Intermediate Python for Data Science. Focus on Matplotlib, Numpy & Pandas. [course]
 - (DataCamp) Data Manipulation with pandas [course]
 - (DataCamp) Joining Data with pandas [course]

▶ Great learning resource, but **not mandatory**. They *complement* the textbook nicely.

▶ Online exercise interface needs some getting used to.

➔ next slide



Exercise

Subset and conquer

Subsetting Python lists is a piece of cake. Take the code sample below, which creates a list `x` and then selects "b" from it. Remember that this is the second element, so it has index 1. You can also use negative indexing.

```
x = ["a", "b", "c", "d"]
x[1]
x[-3] # same result!
```

Remember the `areas` list from before, containing both strings and floats? Its definition is already in the script. Can you add the correct code to do some Python subsetting?

Instructions

100 XP

- Print out the second element from the `areas` list (it has the value `11.25`).
- Subset and print out the last element of `areas`, being `9.50`. Using a negative index makes sense here!
- Select the number representing the area of the living room (`20.0`) and print it out.

Take Hint (-30 XP)

script.py

Light Mode

```
1 # Create the areas list
2 areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75,
3         "bathroom", 9.50]
4 # Print out second element from areas
5 print(areas[1])
6
7 # Print out last element from areas
8 print(areas[-1])
9
10 # Print out the area of the living room
11 print(areas[5])
```

To run multiple lines of code, select them and press **Ctrl + ENTER**

To run a single line of code, with the cursor on the line press **Ctrl + ENTER** (No line selection necessary)

Run Code Submit Answer

iPython Shell

Slides

```
# Print out last element from areas
print(areas[-1])
```

```
11.25
9.5
```

```
print(areas[5])
```

```
20.0
```

```
In [1]: %pprint
```

`dir()` to find out what data objects have been pre-loaded

By default, iPython has "pretty printing" turned on. As a result, list items are printed on separate lines.

To turn this on/off, execute `%pprint`

Wrapping up

- ▶ To-do #3 out
 - ◆ Study numpy
- ▶ Learn:
 - ◆ numpy and pandas. DataCamp has good tutorials.
- ▶ Office hours this week
 - ◆ Na-Rae: Tue 2:30-4 (yesterday), Fri 11-12:30
 - ◆ Emma: Tue, Thu 9-10:30am