

Lecture 3: Processing Language Data, Git/GitHub

LING 1340/2340: Data Science for Linguists

Na-Rae Han

Objectives

- ▶ What do linguistic data look like?
- ▶ Homework 1: What did you process?
- ▶ How does collaborating through GitHub work?

- ▶ Tools:
 - ◆ Git and GitHub
 - ◆ Jupyter Notebook
 - ◆ DataCamp tutorials

**You should
be taking
NOTES!**



First thing to do every class

```
MINGW64:/c/Users/narae/Documents/Data_Science
narae@T450s MINGW64 ~
$ cd Documents/Data_Science/

narae@T450s MINGW64 ~/Documents/Data_Science
$ pwd
/c/Users/narae/Documents/Data_Science

narae@T450s MINGW64 ~/Documents/Data_Science
$ ls
Inaugural-Address-Project/  foo/  planets/

narae@T450s MINGW64 ~/Documents/Data_Science
$ ls -la
total 12
drwxr-xr-x 1 narae 197121 0 Aug 30 22:43 ./
drwxr-xr-x 1 narae 197121 0 Aug 28 16:29 ../
drwxr-xr-x 1 narae 197121 0 Aug 28 16:32 Inaugural-Address-Project/
drwxr-xr-x 1 narae 197121 0 Aug 28 23:19 foo/
drwxr-xr-x 1 narae 197121 0 Aug 29 17:11 planets/

narae@T450s MINGW64 ~/Documents/Data_Science
$ |
```

Back to Class-Practice-Repo

<https://github.com/naraehan/Class-Practice-Repo>

▶ Activity1

- ◆ Team reports on 2 real linguistic data sets

▶ Todo1

- ◆ Your To-do 1 submissions

▶ Lots of files! I have merged in everyone's contributions.

▶ But your own fork does not have those.

Keeping your fork up-to-date

- ▶ The original repo ("upstream") will keep changing.
 - ◆ How to keep your copies (GitHub fork and local repo) up-to-date?
- ▶ Cloning already configured your GitHub fork as "origin":

```
narae@T450s MINGW64 ~/Documents/Data_Science/Class-Practice-Repo (master)
$ git remote -v
origin https://github.com/narae-student/Class-Practice-Repo.git (fetch)
origin https://github.com/narae-student/Class-Practice-Repo.git (push)
```

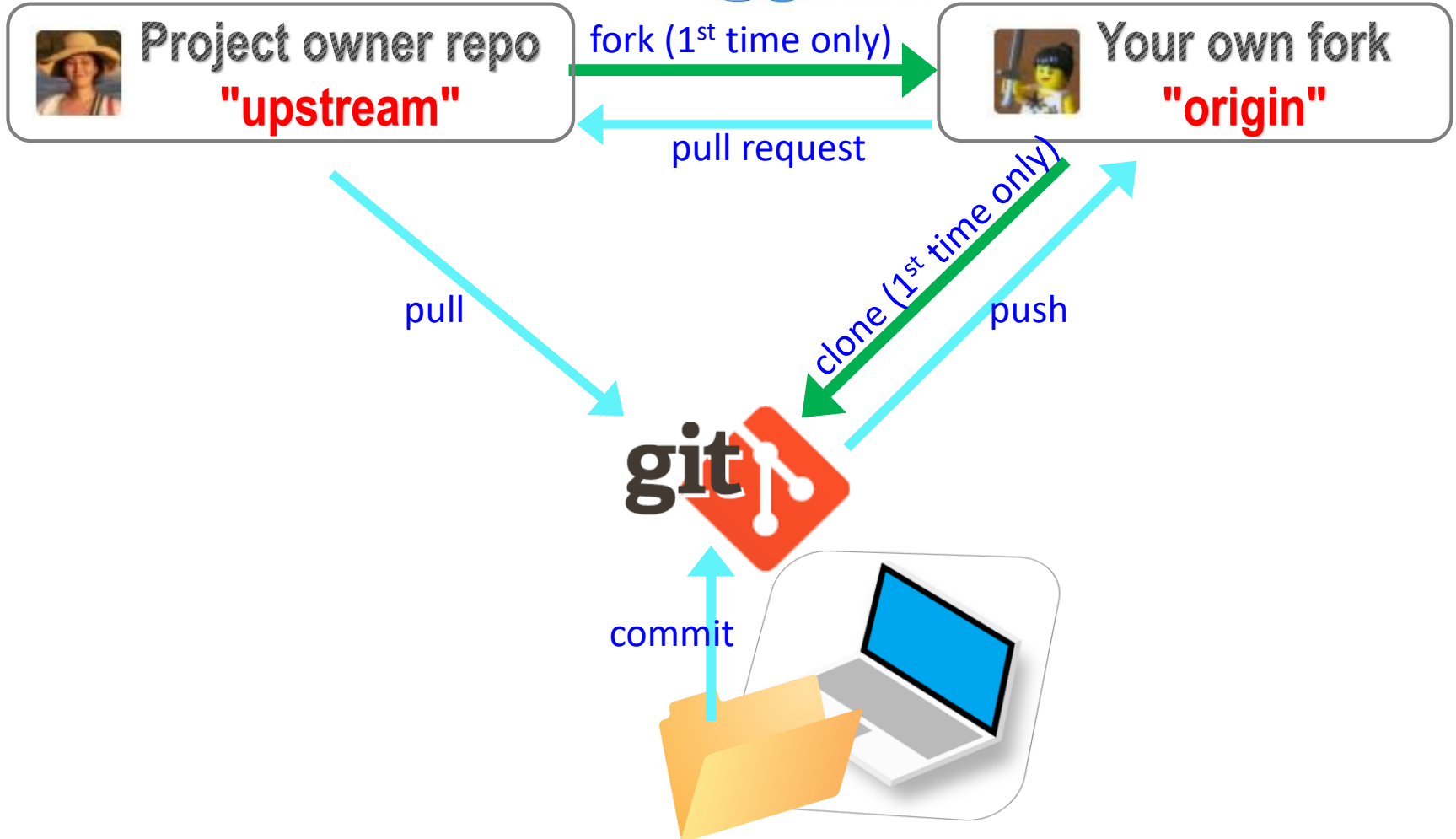
- ▶ Configure the original repo as another remote: "upstream"
 - ◆ `git remote add upstream <GitHub-repo-URL>`
- ▶ When it's time to sync, pull from upstream:
 - ◆ `git pull upstream master`
- ▶ Pushing should be done to your GitHub fork ("origin").
 - ◆ `git push origin master`

You might be able to leave out "origin master".

Two remotes: "origin", "upstream"

```
MINGW64:/c:/Users/narae/Documents/Data_Science/Class-Practice-Repo
narae@T450s MINGW64 ~/Documents/Data_Science
$ ls
Class-Practice-Repo/  HW1-Repo/  Inaugural-Address-Project/  foo/  planets/
narae@T450s MINGW64 ~/Documents/Data_Science
$ cd Class-Practice-Repo/
narae@T450s MINGW64 ~/Documents/Data_Science/Class-Practice-Repo (master)
$ ls
README.md  activity1/  todo1/
narae@T450s MINGW64 ~/Documents/Data_Science/Class-Practice-Repo (master)
$ git remote -v
origin  https://github.com/narae-student/Class-Practice-Repo.git (fetch)
origin  https://github.com/narae-student/Class-Practice-Repo.git (push)
narae@T450s MINGW64 ~/Documents/Data_Science/Class-Practice-Repo (master)
$ git remote add upstream https://github.com/naraehan/Class-Practice-Repo.git
narae@T450s MINGW64 ~/Documents/Data_Science/Class-Practice-Repo (master)
$ git remote -v
origin  https://github.com/narae-student/Class-Practice-Repo.git (fetch)
origin  https://github.com/narae-student/Class-Practice-Repo.git (push)
upstream https://github.com/naraehan/Class-Practice-Repo.git (fetch)
upstream https://github.com/naraehan/Class-Practice-Repo.git (push)
narae@T450s MINGW64 ~/Documents/Data_Science/Class-Practice-Repo (master)
$ |
```

The fork triangle





Up-close with linguistic data

▶ Two data sets:

- ◆ Annotated Learner Texts, by Gina Peirce
- ◆ Istanbul Greek, by Matthew Hadodo

▶ Discussion points:

- ◆ What kind of data set is this? Content? Format? Purpose?
- ◆ How was the data created, collected, and processed?
- ◆ How did the researcher organize the data?
- ◆ At present, what can you *do* with this data?
- ◆ After learning data-science methods, what do you think you will be able to do with the data?
- ◆ Anything else you noticed?

Not yet published

- ▶ Please delete the Istanbul Greek data from your laptop.

HW1: processing pull request, merging

- ▶ With everyone working inside their own directory, merging is conflict-free:

The screenshot shows a GitHub pull request interface. At the top, the repository name is 'naraehan / HW1-Repo'. On the right, there are buttons for 'Unwatch' (1), 'Star' (0), and 'Fork' (17). Below this, navigation tabs include 'Code', 'Issues' (0), 'Pull requests' (3), 'Projects' (0), 'Wiki', 'Settings', and 'Insights'. The main title of the pull request is 'finalized hw1.ipynb and hw1.html #13', with an 'Edit' button to its right. A green 'Open' button is next to the text 'kkairis wants to merge 1 commit into naraehan:master from kkairis:master'. Below the title, there are statistics: 'Conversation' (0), 'Commits' (1), and 'Files changed' (2), along with a green progress bar showing '+12,459 -0'. A comment from 'kkairis' is shown, stating 'No description provided.' Below the comment, a commit is listed: 'finalized hw1.ipynb and hw1.html' with commit hash 'b31071d'. On the right side, there are settings for 'Reviewers' (No reviews—request one), 'Assignees' (No one—assign yourself), 'Labels' (None yet), 'Projects' (None yet), and 'Milestone' (No milestone). At the bottom, a green box contains a checkmark and the text 'This branch has no conflicts with the base branch. Merging can be performed automatically.' Below this, there is a green 'Merge pull request' button and a link to 'open this in GitHub Desktop' or 'view command line instructions'.

Many forks and merges

Contributors

Traffic

Commits

Code frequency

Punch card

Network

Members

Dependents

Owners

5

naraehan

seaweedbeast

als333

ktlandin

kkairis

jos141

narae-student

briblue3

master

master

master

master

master

master

HW1: sync your HW1-Repo

1. Make sure your `.gitignore` is properly configured.

```
narae@T450s MINGW64 ~/Documents/Data_Science/HW1-Repo (master)
$ cat .gitignore
*/data/**
```

2. If not, in the correct directory, execute:

```
echo "*/data/**" > .gitignore
```

3. Configure "upstream" remote:

```
git remote add upstream https://github.com/naraehan/HW1-Repo.git
```

4. Pull from upstream:

```
git pull upstream master
```

5. Probably not necessary to push to your GitHub fork.

HW1: sharing code



- ▶ In your team, decide on 1 homework out of 3 you will try out together as a group.
 - ◆ Best to go with smaller & simpler data set.
- ▶ 2 students might need to manually download the data set.
- ▶ Start Jupyter Notebook script, "Kernel" -> "Restart & Clear Output"
- ▶ Each member runs the Jupyter Notebook script cell-by-cell, while script author walks them through each cell.

Git and GitHub are complicated.

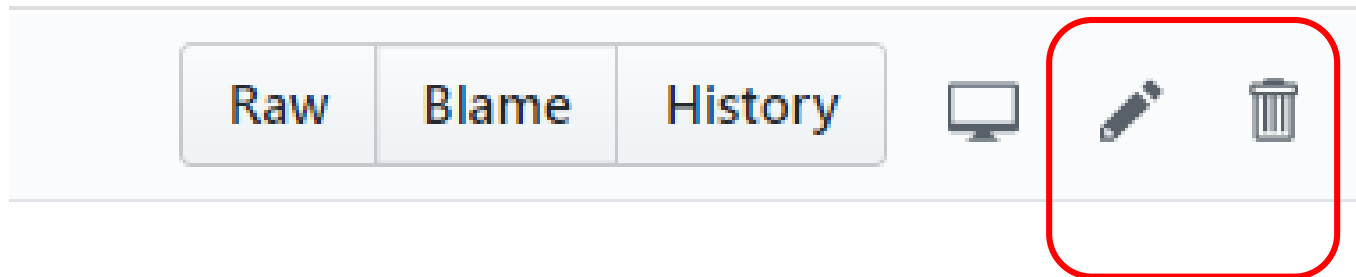


- ▶ They are powerful tools.
 - ▶ There are a lot of abstract, high-level concepts involved.
 - ▶ Concepts do not make sense before you get hands-on.
 - ▶ You cannot get hands-on without the right context.
-
- ← We will learn slowly, learning various pieces as we go.
 - ← You need to be patient, careful and methodical. Make sure you don't rush, and follow instructions.

Git and GitHub are complicated.



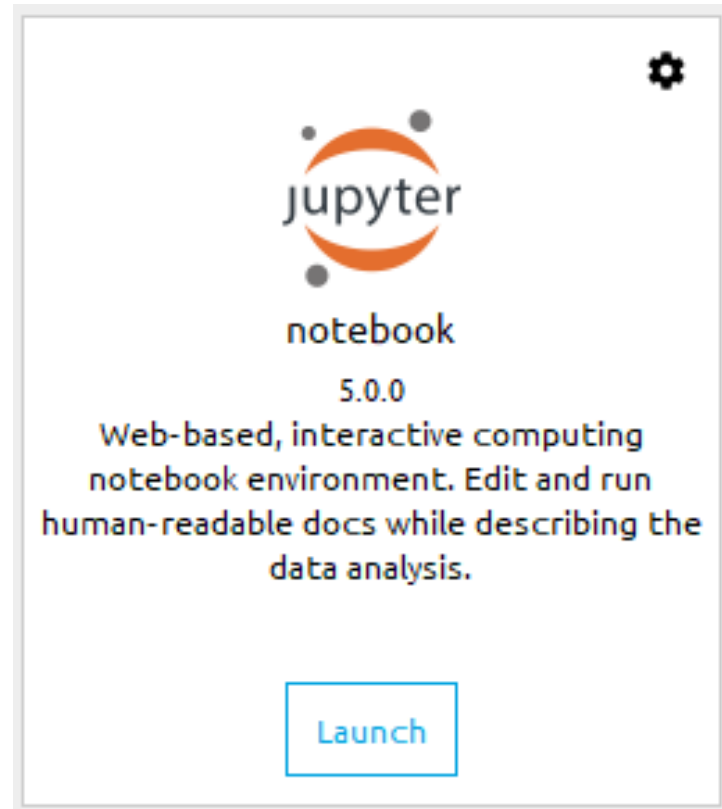
- ▶ We will follow some ground rules.
- ▶ **DO NOT EDIT A REPOSITORY'S CONTENT THROUGH GITHUB.**



DO NOT USE!!

Jupyter Notebook

- ▶ Allows you to create and share documents that contain live code, 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



Course group on DataCamp





Data Science for Linguists PREMIUM

Buy More Seats

Courses

Active Courses (4) Curriculum

Skill Search

Name	Enrolled	Completed	
 Intro to Python for Data Science	2	3	Assign
 Intermediate Python for Data Science	2	0	Assign
 Introduction to R	0	2	Assign
 Natural Language Processing Fundamentals in Python	1	0	Assign

Don't find the correct course? [Check out the Curriculum](#)

Manage Your Group

- Members
- Assignments
- Courses**
- Settings
- Export Group Data

Approved
Congratulations! Your DataCamp Classroom Account was approved. All invited students have full access until **23 Nov 2017**.

How to use DataCamp

- ▶ Topics for the next couple of weeks:
 - ◆ numpy library
 - ◆ pandas library
 - ◆ visualization libraries such as `matplotlib`
- ▶ The video tutorials are linked as "assignments"
 - ◆ Great learning resource, but not mandatory.
 - ◆ They *complement* the textbook nicely.
- ▶ Online exercise interface needs some getting used to.
 - ➔ next slide



Subset and conquer

100xp

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 = list["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

- Print out the second element from the `areas` list, so `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 and print it out.
- [Take Hint \(-30xp\)](#)

script.py

```

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.)

Submit Answer

IPython Shell

Slides

```

18.0,
'living room',
20.0,
'bedroom',
10.75,
'bathroom',
9.5]

```

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`.

```
In [4]: print(areas[-1])
9.5
```

```
In [5]: %pprint
```

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



Wrapping up

- ▶ **To-do 2: due on Tuesday.**
 - ◆ Study numpy, make your own notebook. Submit via Class-Practice-Repo.
- ▶ **Office hours**
 - ◆ Mon/Wed 2:30 -- 4pm. G17 CL
 - ◆ Also by appointment
- ▶ **Learn:**
 - ◆ Git, GitHub
 - ◆ Jupyter Notebook
 - ◆ numpy, pandas