

Lecture 3: Processing Linguistic Data, Git/GitHub

LING 1340/2340: Data Science for Linguists

Na-Rae Han

Objectives

- ▶ To-do 1: What linguistic data did you find?
- ▶ HW1: What did you process?
- ▶ GitHub: completing the fork triangle
- ▶ DataCamp tutorials

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

**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

To-do #1

- ▶ What linguistic data sets did you look at?
 - ◆ Corpus data?
 - ◆ Non-corpus data?

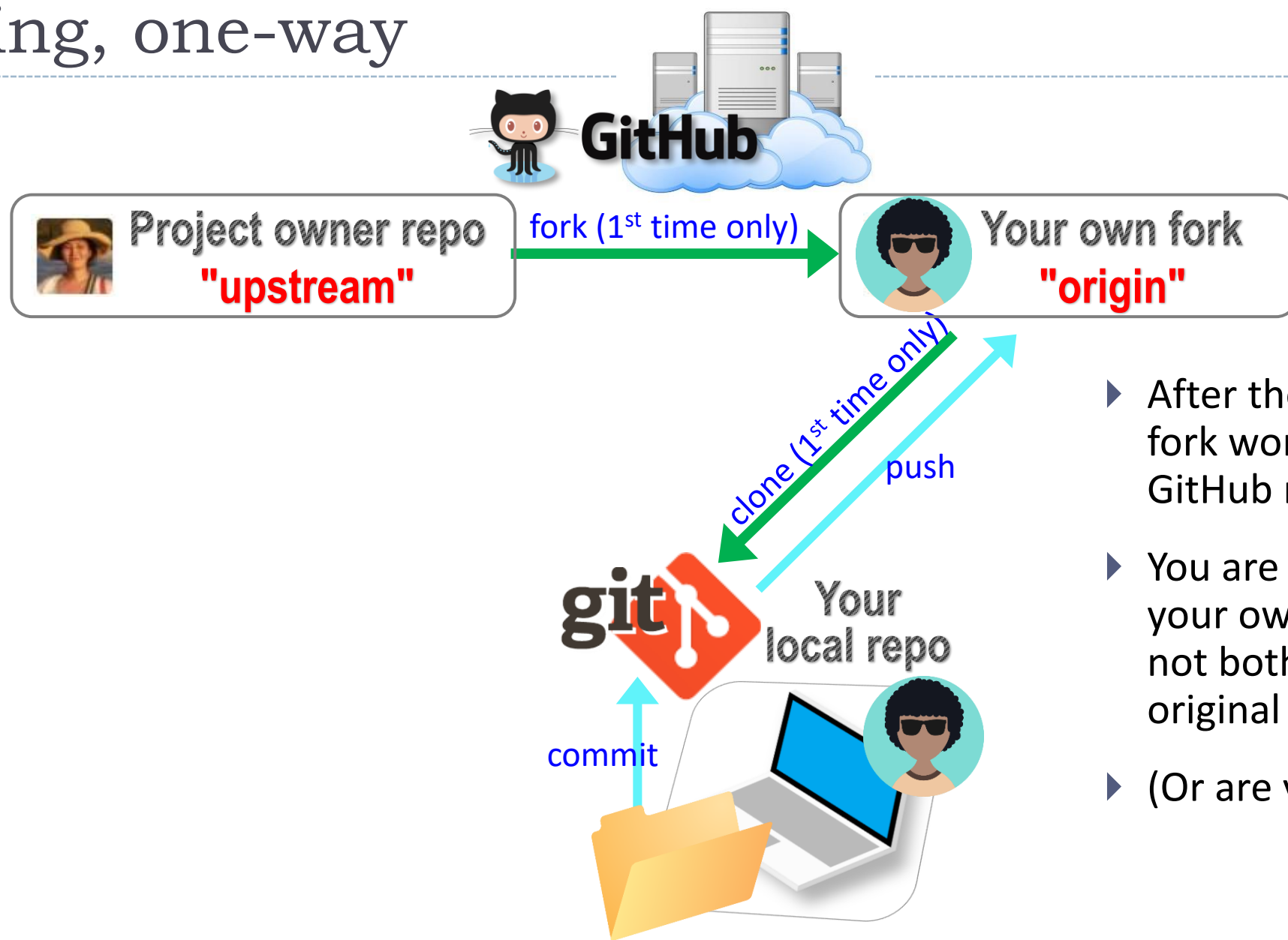
- ▶ What makes a dataset a corpus?

Back to Class-Exercise-Repo

<https://github.com/Data-Science-for-Linguists-2022/Class-Exercise-Repo>

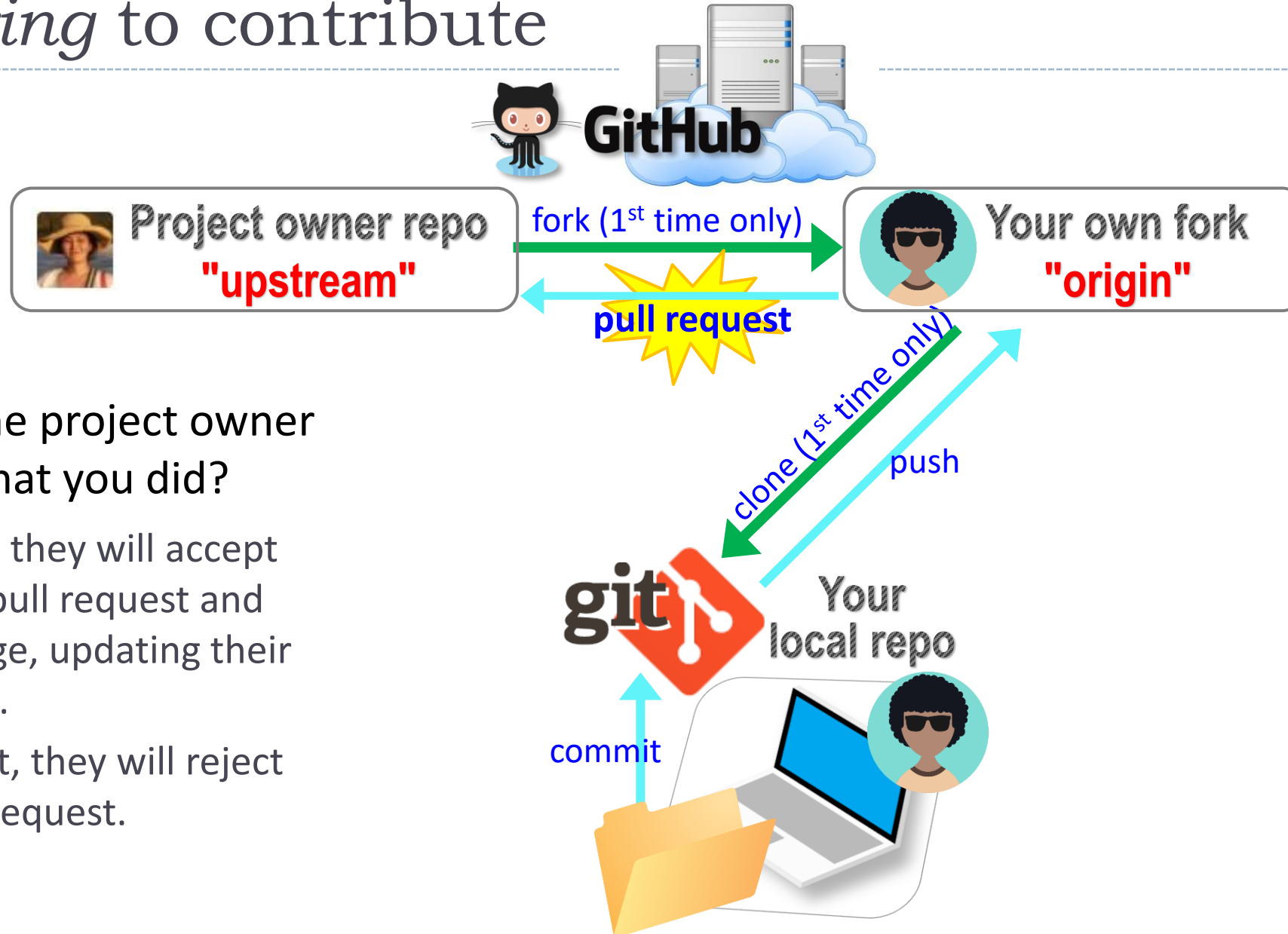
- ▶ Todo1
 - ◆ Your To-do 1 submissions
- ▶ Lots of files -- I have merged in everyone's contributions.
- ▶ **But! Your own fork does not have those.**

Forking, one-way



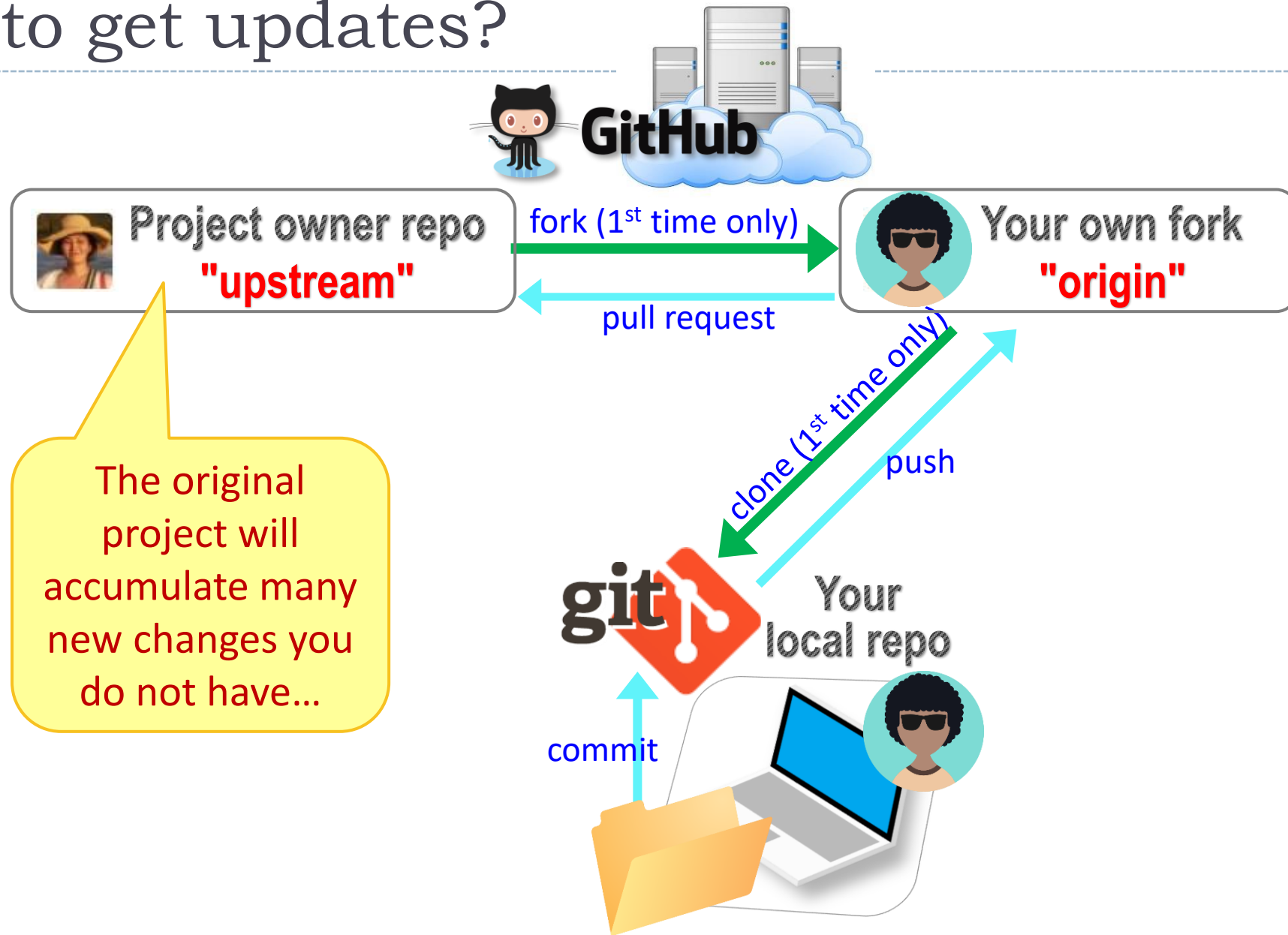
- ▶ After the spin-off, your fork works as if your own GitHub repo.
- ▶ You are content to do your own development, not bothering the original project owner...
- ▶ (Or are you??)

Offering to contribute



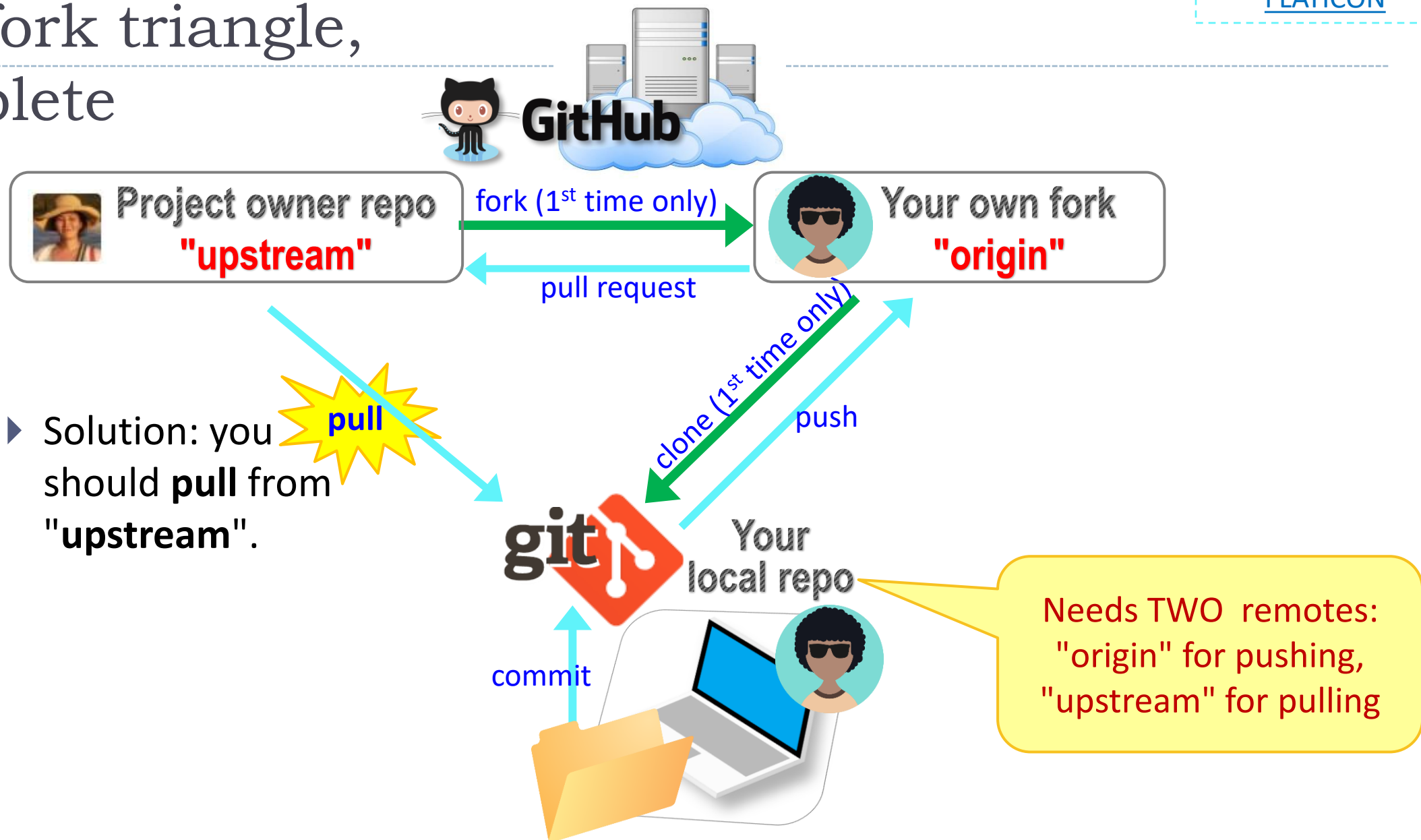
- ▶ Will the project owner like what you did?
 - ◆ If so, they will accept the pull request and merge, updating their repo.
 - ◆ If not, they will reject the request.

How to get updates?



The original project will accumulate many new changes you do not have...

The fork triangle, complete



- ▶ Solution: you should **pull** from "upstream".

Keeping your fork up-to-date

- ▶ The original repo ("upstream") will have new changes from other users.

- ◆ How to keep your copies (GitHub fork and local repo) up-to-date?

- ▶ Cloning already configured your GitHub fork as "origin":

```
narae@T480s MINGW64 ~/Documents/Data_Science/Class-Exercise-Repo (main)
$ git remote -v
origin https://github.com/narae-student/Class-Exercise-Repo.git (fetch)
origin https://github.com/narae-student/Class-Exercise-Repo.git (push)
```

- ▶ Configure the original repo as another remote: "upstream"

- ◆ `git remote add upstream <GitHub-repo-URL.git>`

- ▶ When it's time to sync, pull from upstream:

- ◆ `git pull upstream main`

- ▶ Pushing should be done to your GitHub fork ("origin").

- ◆ `git push origin main`

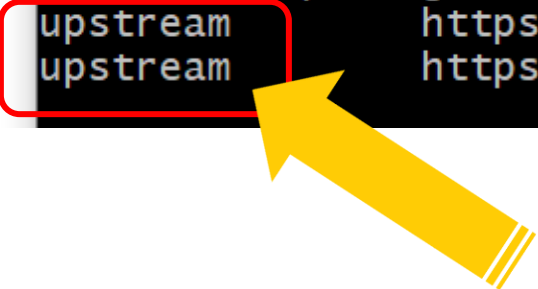
You might be able to leave out "origin main".

Two remotes: "origin", "upstream"

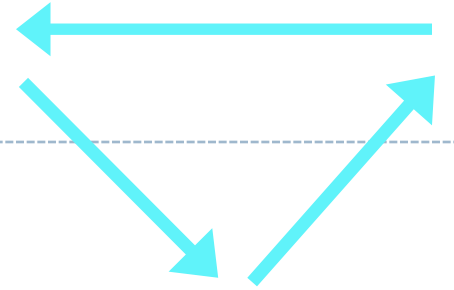
```
narae@T480s MINGW64 ~/Documents/Data_Science/Class-Exercise-Repo (main)
$ git remote -v
origin https://github.com/narae-student/Class-Exercise-Repo.git (fetch)
origin https://github.com/narae-student/Class-Exercise-Repo.git (push)

narae@T480s MINGW64 ~/Documents/Data_Science/Class-Exercise-Repo (main)
$ git remote add upstream https://github.com/Data-Science-for-Linguists-2022/Class-Exercise-Repo.git

narae@T480s MINGW64 ~/Documents/Data_Science/Class-Exercise-Repo (main)
$ git remote -v
origin https://github.com/narae-student/Class-Exercise-Repo.git (fetch)
origin https://github.com/narae-student/Class-Exercise-Repo.git (push)
upstream https://github.com/Data-Science-for-Linguists-2022/Class-Exercise-Repo.git (fetch)
upstream https://github.com/Data-Science-for-Linguists-2022/Class-Exercise-Repo.git (push)
```



The fork triangle: workflow



▶ On your **laptop**

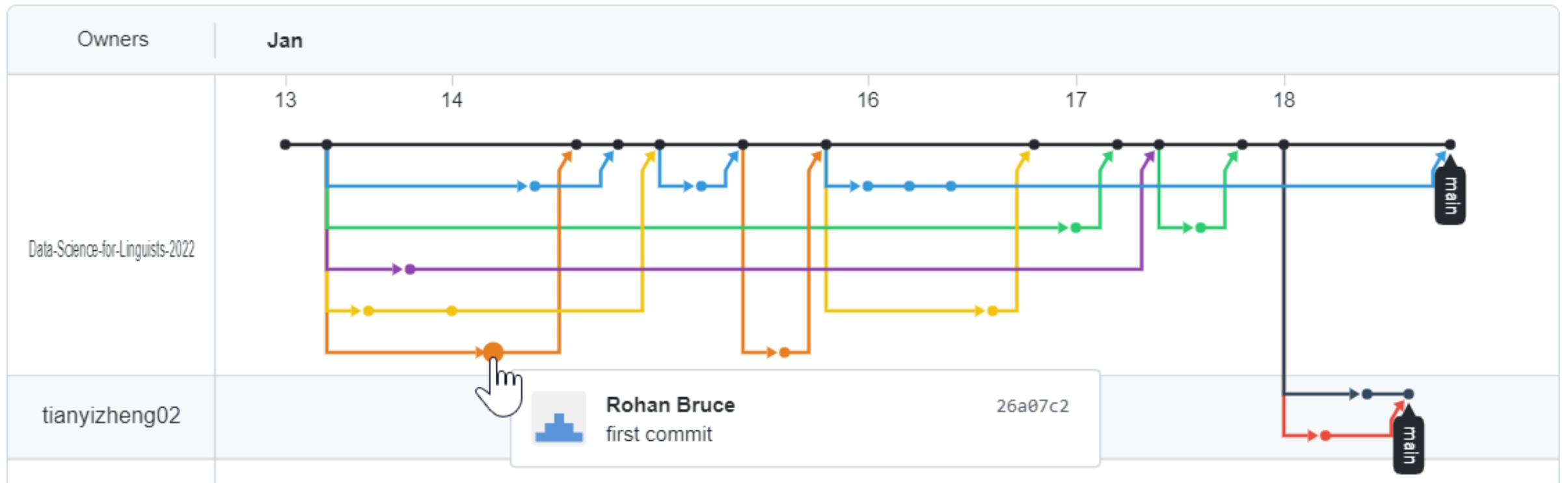
1. Check your local repo's status: `git status`. Get it to a clean state.
2. Pull from "upstream", syncing your local repo: `git pull upstream main`. Your local repo now has all latest changes.
 - ◆ If there is a merge conflict, you will need to resolve it. (fingers crossed)
3. Do your work! New files, edits, etc.
4. Do your usual local Git routine: `git add` and `git commit`.
5. Push new versions to your own GitHub fork ("origin"): `git push (origin main)`

▶ On **GitHub**

1. Check your forked repo. It should have your new work.
2. Create a **pull request** for the original repo ("upstream") owner.
3. Give it some time, and check back on the status of your pull request.

Many forks and merges

- ▶ <https://github.com/Data-Science-for-Linguists-2022/Class-Exercise-Repo/network>



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 a repository named 'Data-Science-for-Linguists-2022 / HW1-Repo'. The pull request is titled 'BenM Submission #6' and is submitted by user 'bam197'. It shows 6 commits and 2 files changed. The commit history includes: 'Add PROIELReader', 'Fix some issues with PROIELReader', 'Fix an error in PROIELReader involving empty tokens', 'Add Jupyter Notebook file', 'Add more example sentences with translations', and 'Finalize HW1 notebook'. A green checkmark indicates 'This branch has no conflicts with the base branch'. A 'Merge pull request' button is visible. The right sidebar shows 'Reviewers' (No reviews), 'Assignees' (No one—assign yourself), 'Labels' (None yet), 'Projects' (None yet), 'Milestone' (No milestone), 'Linked issues', and 'Notifications' (Unsubscribe).

HW1: sync your HW1-Repo

1. Configure "upstream" remote:

```
git remote add upstream https://github.com/Data-Science-for-Linguists-2022/HW1-Repo.git
```

2. Pull from upstream:

```
git pull upstream main
```

3. 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?

Git and GitHub are complicated.

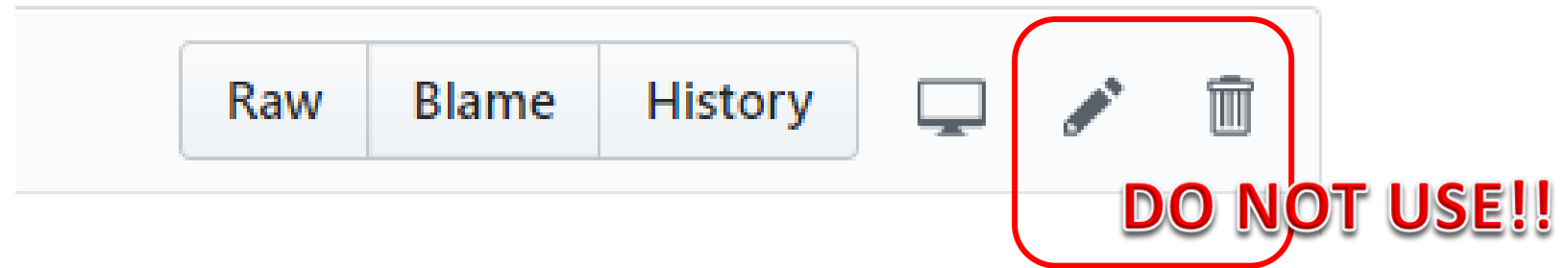


- ▶ They are powerful tools.
 - ▶ There are a lot of abstract, high-level concepts involved.
 - ▶ Concepts do not make sense before you get hands-on.
 - ▶ You cannot get hands-on without the right context.
-
- ← We will learn slowly, learning various pieces as we go.
 - ← You need to be patient, careful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



- ▶ Don't accidentally commit a file! Be mindful of what you add. Avoid using:
 - ◆ `git add .`
 - ◆ `git add *`
- ▶ For now, do not **delete** or **re-name** any previously committed file.
 - ◆ If you must: use `git rm` and `git mv`.

Course Group on DataCamp

▶ Video-based, interactive tutorials

datacamp LEARN WORKSPACE Search Catalog My Account

My Progress

Welcome back, Na-Rae!
Reach 250 XP today to continue your streak ...

0 day streak
0 XP

LEARN
Introduction to R >

● Intro to basics
⌚ 4 hours to go

Keep Making Progress

PRACTICE
Introduction to Python >

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

How to use DataCamp

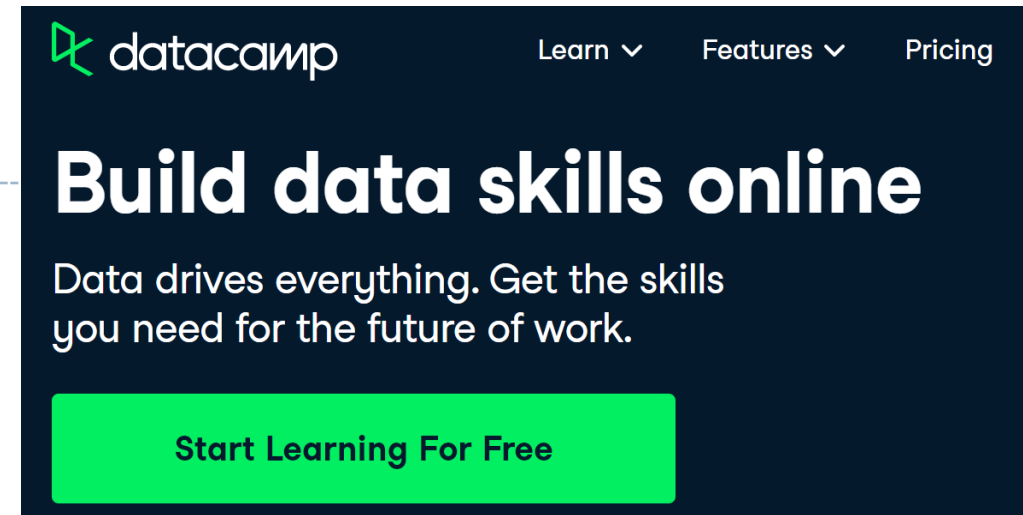
- ▶ Topics for the next couple of weeks:
 - ◆ numpy library
 - ◆ pandas library
 - ◆ visualization libraries such as matplotlib

- ▶ 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 #2 is out: due Thu.
 - ◆ Study numpy, make your own study notes as JNB. Submit via Class-Exercise-Repo.
- ▶ Try out DataCamp tutorials! Especially the numpy chapter.
- ▶ Learn:
 - ◆ Git, GitHub
 - ◆ Jupyter Notebook
 - ◆ numpy
 - ◆ Pandas