Lecture 3: Processing Linguistic Data, Git/GitHub

LING 1340/2340: Data Science for Linguists

Jevon Heath

Objectives

- HW1: What did you process?
- GitHub: completing the fork triangle
- DataCamp tutorials

- Tools:
  - Git and GitHub
  - Jupyter Notebook
  - OS X Terminal: enable color

You should be taking NOTES!

First thing to do every class

```
(base) LI-FVFZ40VAL415:Home jsh82$ cd ~/DataScience2020/
(base) LI-FVFZ40VAL415:DataScience2020 jsh82$ pwd
/Users/jsh82/DataScience2020
(base) LI-FVFZ40VAL415:DataScience2020 jsh82$ ls
Blank repo          Home                lecture2.pptx
Class-Exercise-Repo languages           lecture3.pptx
HW1-Repo            lecture1.pptx       ~$lecture3.pptx
(base) LI-FVFZ40VAL415:DataScience2020 jsh82$ ls -la
total 17856
drwxr-xr-x  12 jsh82  PITT\Domain Users      384 Jan 13 20:33 .
drwxr-xr-x+ 42 jsh82  PITT\Domain Users     1344 Jan 14 11:29 ..
-rw-r--r--@  1 jsh82  PITT\Domain Users     8196 Jan 10 10:44 .DS_Store
drwxr-xr-x  20 jsh82  PITT\Domain Users      640 Jan  9 14:43 Blank repo
drwxr-xr-x   7 jsh82  PITT\Domain Users      224 Jan  9 12:19 Class-Exercise-Repo
drwxr-xr-x  18 jsh82  PITT\Domain Users      576 Jan 10 10:32 HW1-Repo
drwxr-xr-x  13 jsh82  PITT\Domain Users      416 Jan  7 14:50 Home
drwxr-xr-x   4 jsh82  PITT\Domain Users      128 Jan  9 13:25 languages
-rw-r--r--@  1 jsh82  PITT\Domain Users  3617305 Jan  8 20:41 lecture1.pptx
-rw-r--r--@  1 jsh82  PITT\Domain Users  3015102 Jan  9 12:56 lecture2.pptx
-rw-r--r--@  1 jsh82  PITT\Domain Users  2482853 Jan 13 20:33 lecture3.pptx
-rw-r--r--@  1 jsh82  PITT\Domain Users      165 Jan 13 20:28 ~$lecture3.pptx
(base) LI-FVFZ40VAL415:DataScience2020 jsh82$
```

```
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

Back to Class-Exercise-Repo

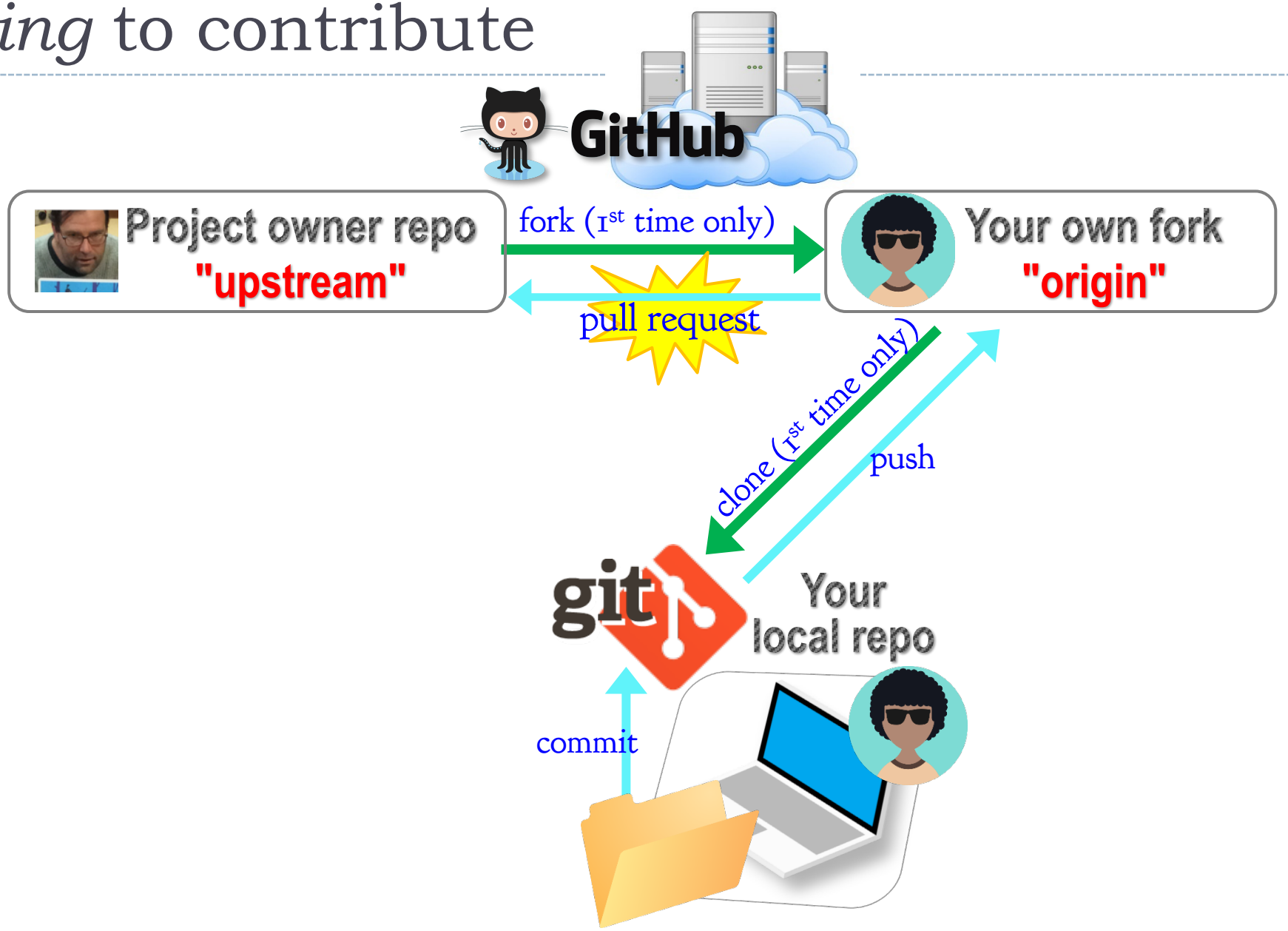
https://github.com/Data-Science-for-Linguists-2020/Class-Exercise-Repo

- Todo1
  - Your To-do 1 submissions

- Lots of files – I have merged in everyone's contributions.
- **But! Your own fork does not have those.**

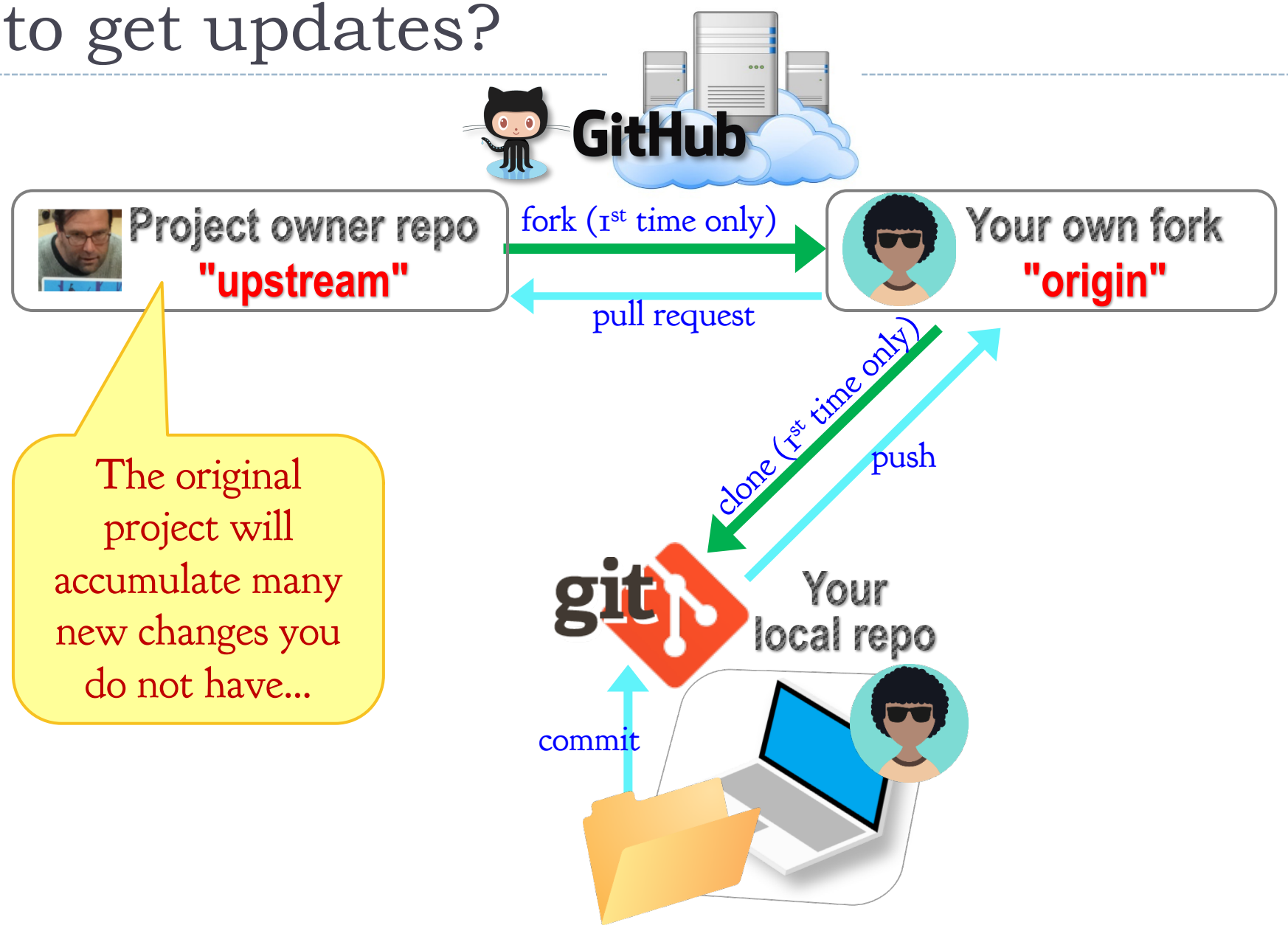
Offering to contribute

*Avatar icons by FLATICON



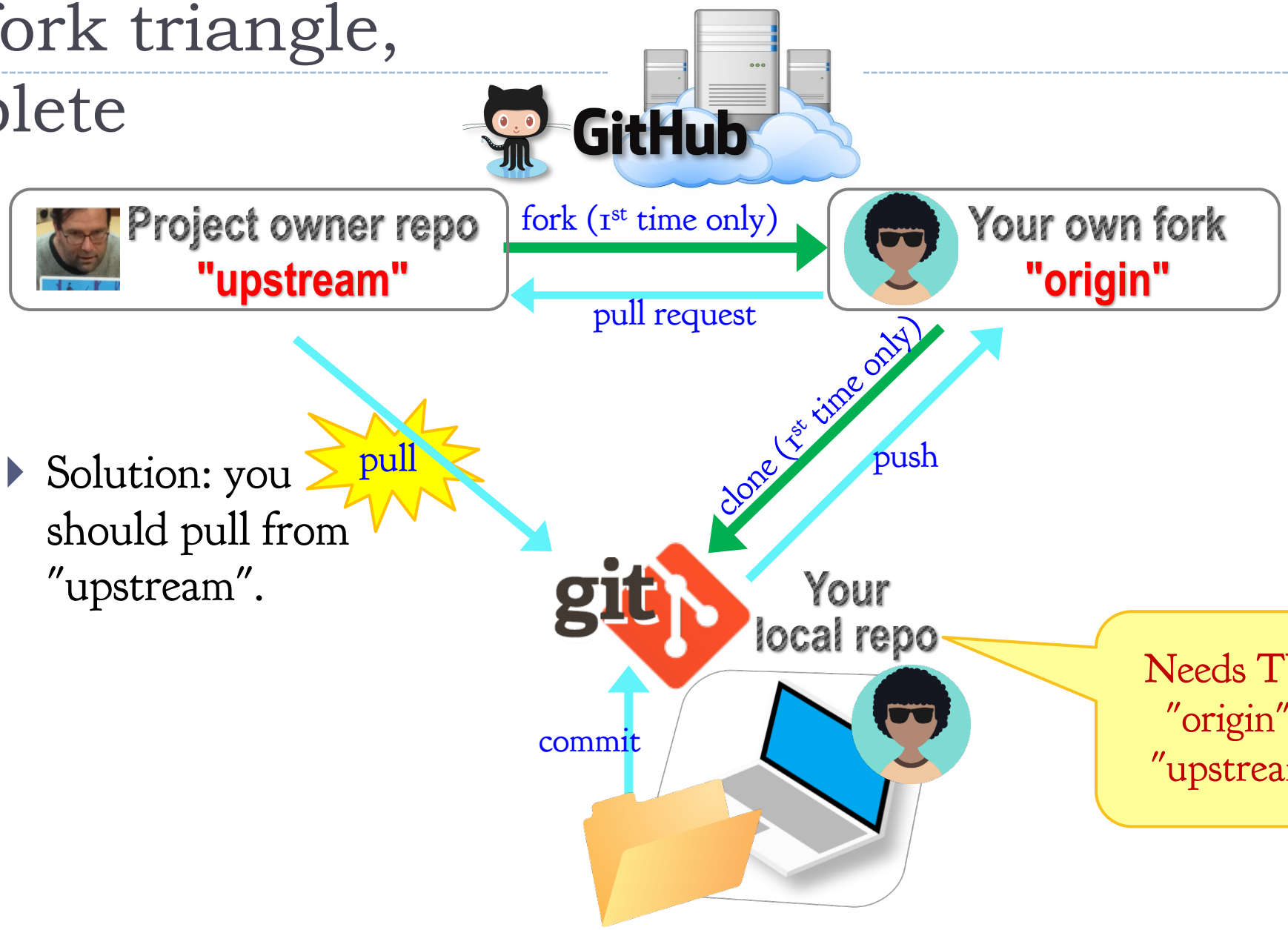
How to get updates?

*Avatar icons by FLATICON



The fork triangle, complete

*Avatar icons by FLATICON



- Solution: you should pull from "upstream".

Needs TWO remotes: "origin" for pushing, "upstream" for pulling

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":

```
(base) LI-FVFZ40VAL415:Class-Exercise-Repo jsh82$ git remote -v
origin  https://github.com/jevonstudent/Class-Exercise-Repo.git (fetch)
origin  https://github.com/jevonstudent/Class-Exercise-Repo.git (push)
(base) LI-FVFZ40VAL415:Class-Exercise-Repo jsh82$
```

- 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".

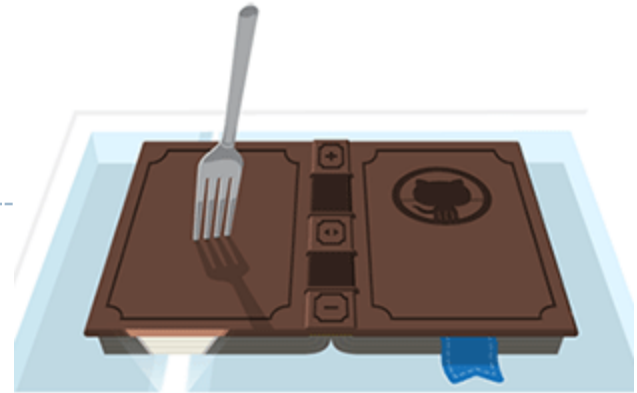
Two remotes: "origin", "upstream"

```
(base) LI-FVFZ40VAL415:jevonstudent jsh82$ cd Class-Exercise-Repo/
(base) LI-FVFZ40VAL415:Class-Exercise-Repo jsh82$
(base) LI-FVFZ40VAL415:Class-Exercise-Repo jsh82$ git remote -v
origin  https://github.com/jevonstudent/Class-Exercise-Repo.git (fetch)
origin  https://github.com/jevonstudent/Class-Exercise-Repo.git (push)
(base) LI-FVFZ40VAL415:Class-Exercise-Repo jsh82$ git remote add upstream https://github.com/Data-S
cience-for-Linguists-2020/Class-Exercise-Repo.git
(base) LI-FVFZ40VAL415:Class-Exercise-Repo jsh82$ git remote -v
origin  https://github.com/jevonstudent/Class-Exercise-Repo.git (fetch)
origin  https://github.com/jevonstudent/Class-Exercise-Repo.git (push)
upstream        https://github.com/Data-Science-for-Linguists-2020/Class-Exercise-Repo.git (fetch)
upstream        https://github.com/Data-Science-for-Linguists-2020/Class-Exercise-Repo.git (push)
(base) LI-FVFZ40VAL415:Class-Exercise-Repo jsh82$
```

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 master`. 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 master`
- 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.

Forking: summary



- When you **start with someone else's project.**
  - You are *not* a collaborator in their repo. (No push access)
- https://help.github.com/articles/fork-a-repo/
- You fork the original repo into your own GitHub account, creating your own "fork".
- You make changes in your own fork. The original repo is not affected!
- pull request: When you think the original project could benefit from your new work, you ask the owner to "pull" from your fork.
  - Owner of original ("upstream") will review your contribution, and then either merge it or reject it.
- Sync with the original repo by pulling from "upstream"

HW1: processing pull request, merging

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

Data-Science-for-Linguists-2020 / HW1-Repo Private
forked from jevonheath/HW1-Repo

Watch 0 | Star 0 | Fork 10

Code | Pull requests 1 | Actions | Projects 0 | Wiki | Security | Insights | Settings

hw1 #6 Edit

Open joeylivorno wants to merge 1 commit into `Data-Science-for-Linguists-2020:master` from `joeylivorno:master`

Conversation 0 | Commits 1 | Checks 0 | Files changed 2 +406 −0

joeylivorno commented 4 minutes ago

No description provided.

hw1 b17eac8

Add more commits by pushing to the **master** branch on **joeylivorno/HW1-Repo**.

Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request | You can also open this in GitHub Desktop or view command line instructions.

Reviewers
No reviews

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

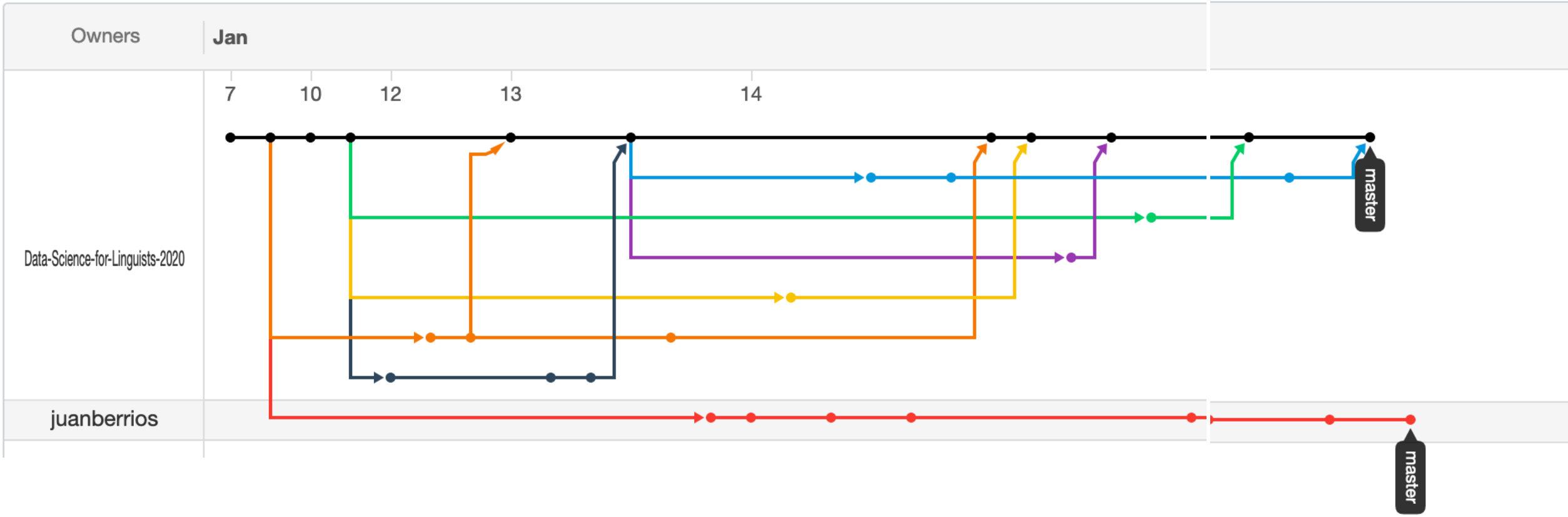
Milestone
No milestone

Notifications Customize
Subscribe
You're not receiving notifications from this thread.

Write Preview

Many forks and merges

- https://github.com/Data-Science-for-Linguists-2020/HW1-Repo/network



HW1: sync your HW1-Repo

1. Configure "upstream" remote:

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

2. Pull from upstream:

```
git pull upstream master
```

3. Push to your GitHub fork:

```
git push origin master
```

Everyone's repos are synced.

Now, everyone has everyone's homework submission.

HW1: Review

- What did you all work on?
- Your 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?

HW1: sharing code

- Pair up. Decide whose homework you will try out together. (author/guest)
  - Best to go with smaller & simpler data set.
- Author should help guest run his/her code.
  - Guest partner will need to manually download the data set, in data/ directory.
  - Guest partner runs the author's original JNB file directly. **Don't copy or rename.**
  - Clear code output first: "Kernel" -> "Restart & Clear Output"
- Guest partner runs the Jupyter Notebook script cell-by-cell, while script author walks them through each cell.

- Go ahead and save (=overwrite) your mate's file.
  ← Oops, you shouldn't have done that.
- No problem! Git to the rescue:
  `git checkout filename.ipynb`

Git and GitHub are complicated.

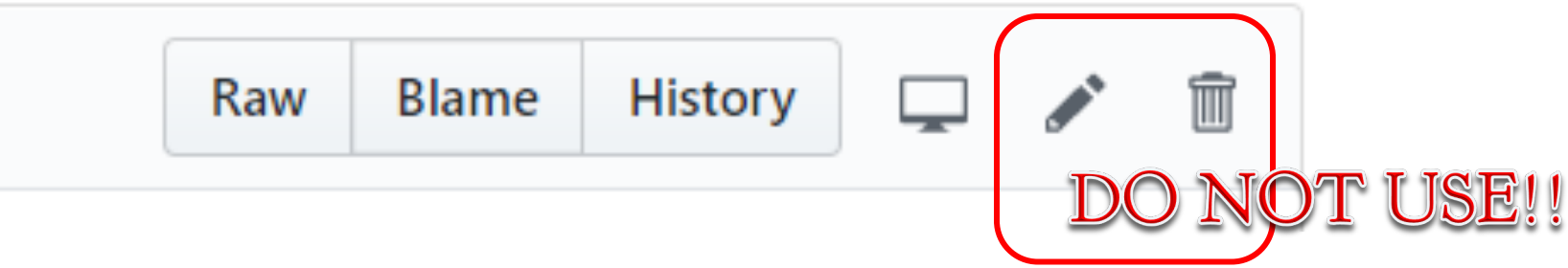
- They are powerful tools.
- There are a lot of abstract, high-level concepts involved.
- Concepts do not make sense before you get hands-on.
- You cannot get hands-on without the right context.

← 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.

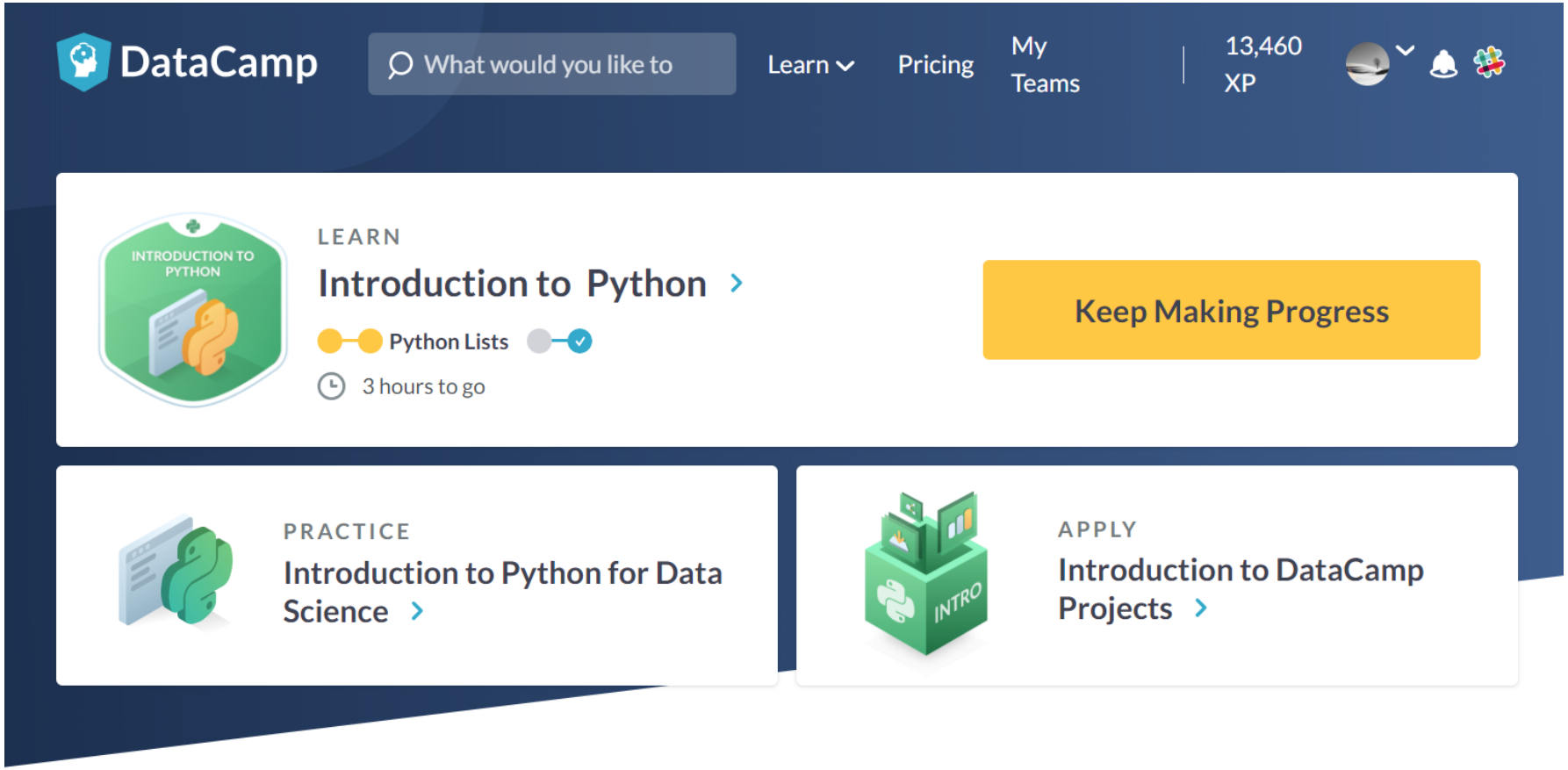
- We will follow some ground rules.
- DO NOT EDIT A REPOSITORY'S CONTENT THROUGH GITHUB.



- 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`.

DataCamp

- Video-based, interactive tutorials



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

https://campus.datacamp.com/courses/intro-to-python-for-data-science/chapter-2-python-lists?ex=7

DataCamp

Course Outline

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

```
# Create the areas list
areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75,
"bathroom", 9.50]

# Print out second element from areas
print(areas[1])

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

# Print out the area of the living room
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]

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

In [5]: %pprint
```

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.

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

Your text editor in shell

- You should be able to launch your text editor from shell and create a new text file in the directory.

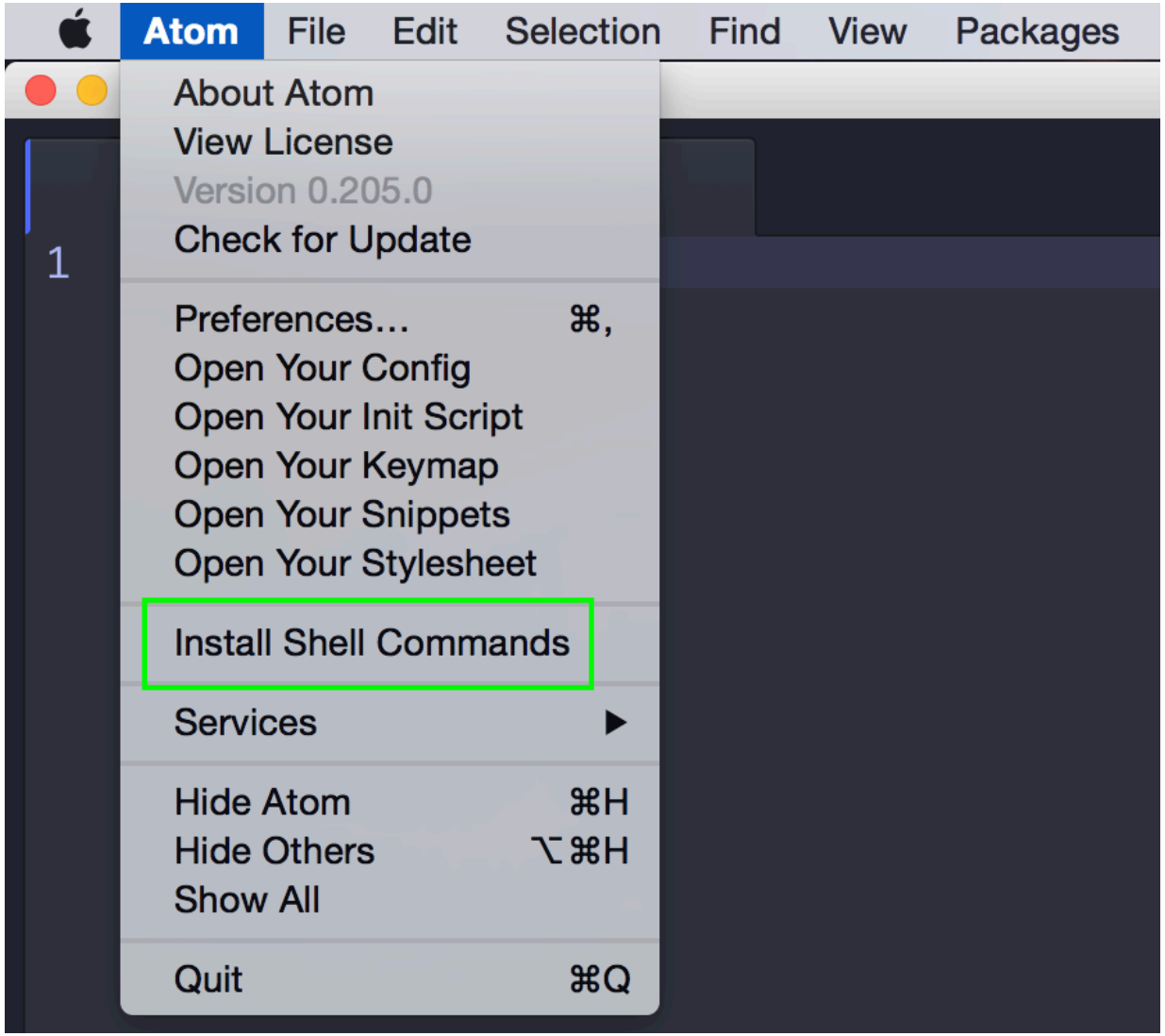
```
(base) LI-FVFZ40VAL415:Class-Exercise-Repo jsh82$ which atom
/usr/local/bin/atom
(base) LI-FVFZ40VAL415:Class-Exercise-Repo jsh82$ atom newfile.txt
(base) LI-FVFZ40VAL415:Class-Exercise-Repo jsh82$ ls
README.md   newfile.txt todo1
```

Atom launches in a new window. I type in some stuff and save file.

New file has been created.

Mac users: configure Atom for shell

- https://stackoverflow.com/questions/22390709/how-to-open-atom-editor-from-command-line-in-os-x
  - "Install Shell Commands"
  - After this, you can launch atom directly from your Terminal (bash shell).



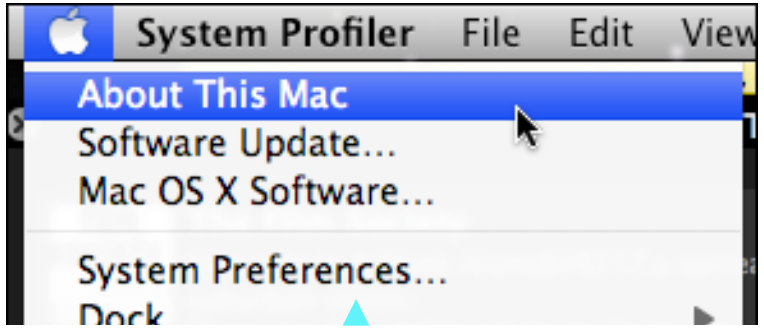
Git is better in color (actually, everything is)

- **Windows** folks are using Git-bash, which has nice colorized Git output

- **Mac** users: There are ways to customize OS X's Terminal…

Adding color to Terminal (Mac only)

1. Open up a Terminal window
2. Type `git config --global color.ui true`
3. For OS X 10.8+, type `nano ~/.bash_profile`.
   - If 10.7 or earlier, replace `~/.bash_profile` with `~/.profile` or `~/.bashrc` or `/etc/profile`.
4. At the bottom, add the two lines of text found at http://tiny.cc/maccolors, save, and exit
5. Run `source ~/.bash_profile`
6. Then go to Terminal > Preferences > Profiles > Text and check "Display ANSI Colors".



Check your OS X version here

```
export CLICOLOR=1
export LSCOLORS=GxFxCxDxBxegedabagaced
```

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?

- Learn:
  - Git, GitHub
  - Jupyter Notebook
  - numpy
  - pandas