Lecture 3: Processing Linguistic Data, Git/GitHub

LING 1340/2340: Data Science for Linguists

Na-Rae Han

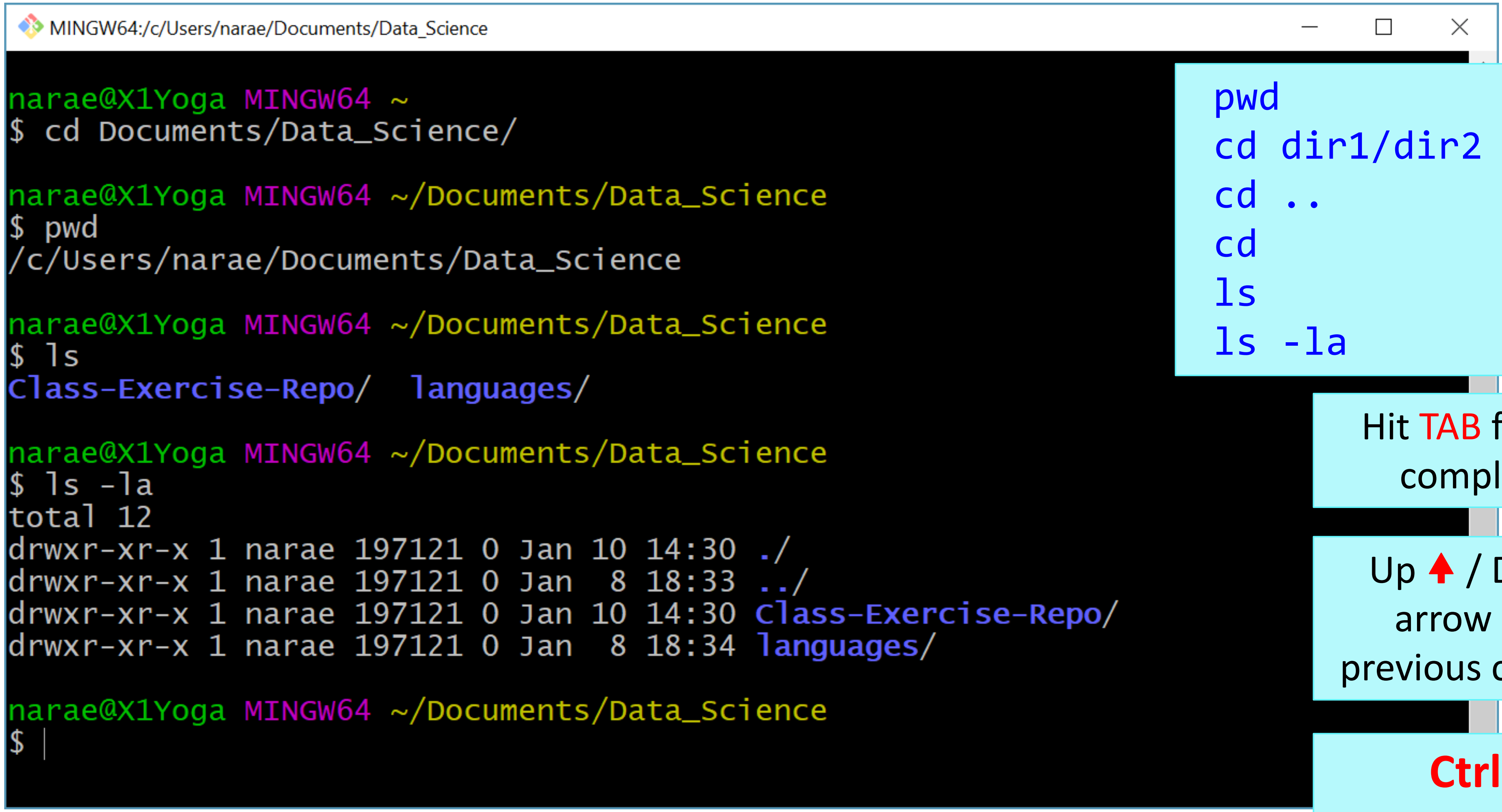
Objectives

- 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



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

Back to Class-Exercise-Repo

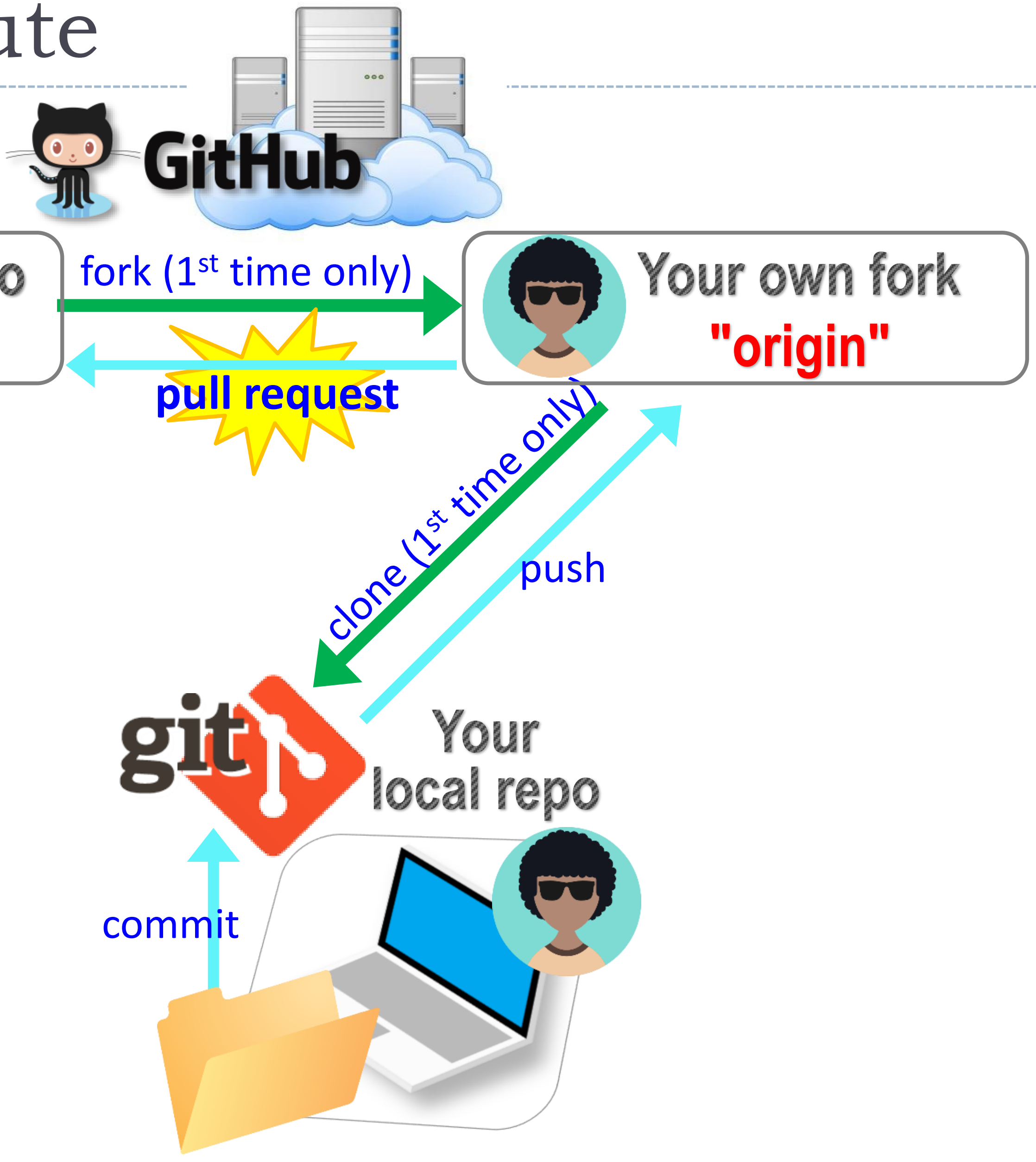
https://github.com/naraehan/Class-Exercise-Repo

- Todo1
  - Your To-do 1 submissions

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

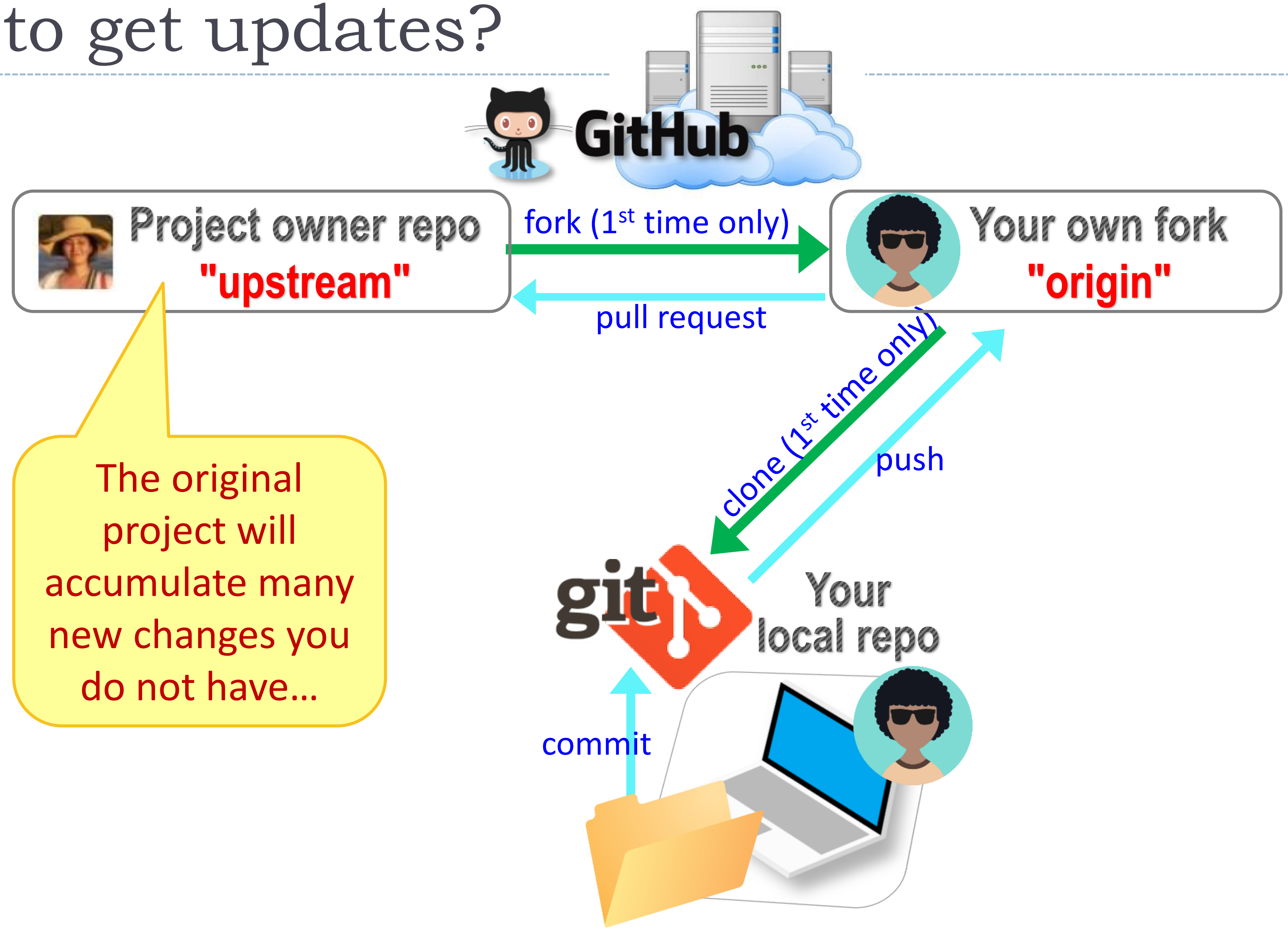
*Avatar icons by FLATICON

Offering to contribute



How to get updates?

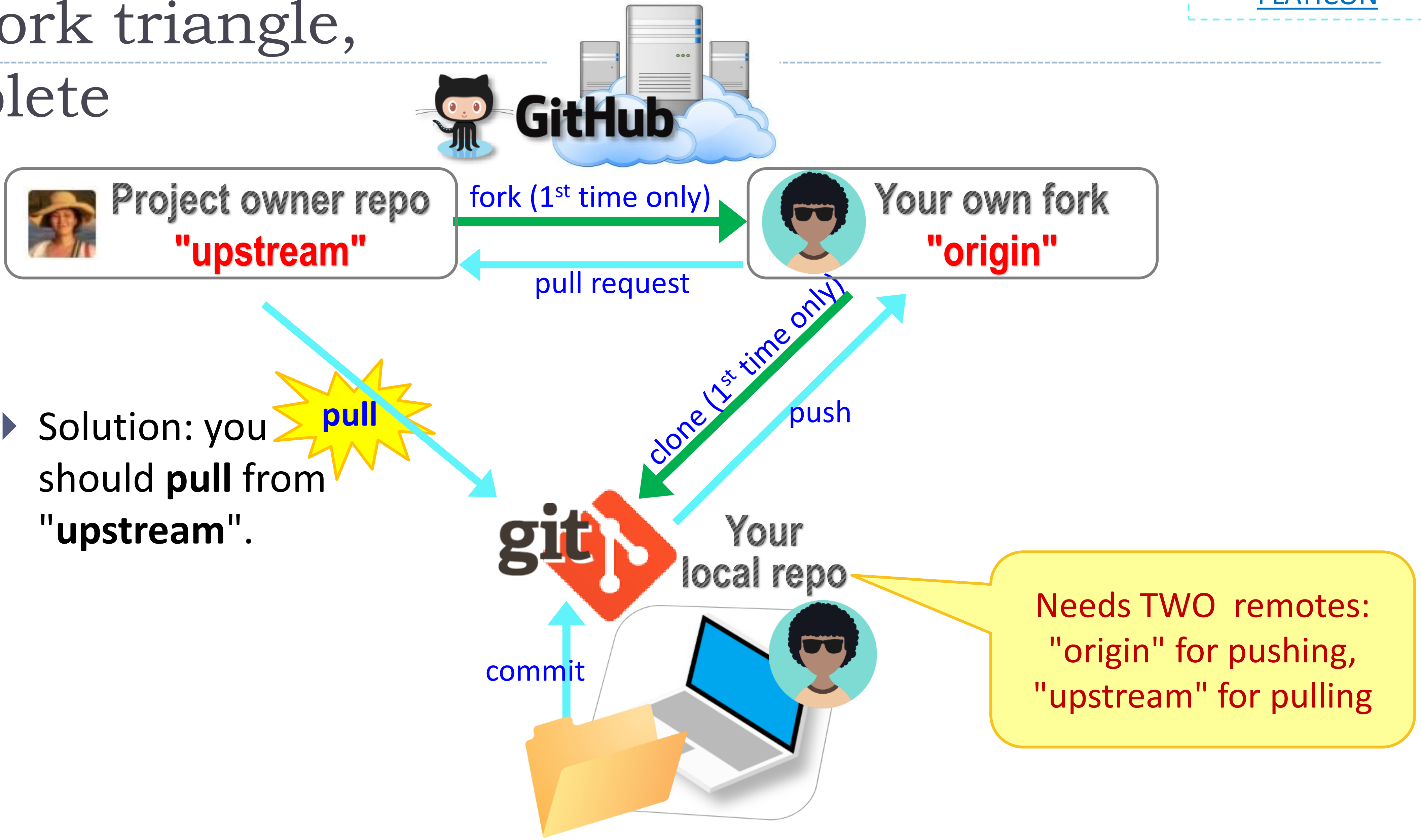
*Avatar icons by FLATICON



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

*Avatar icons by [FLATICON]

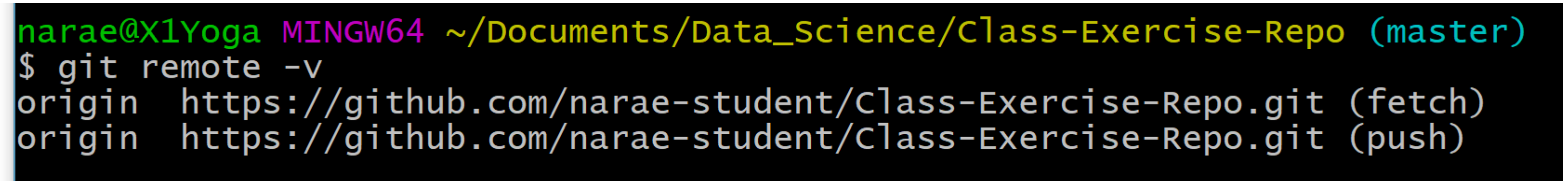
The fork triangle, complete



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

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":

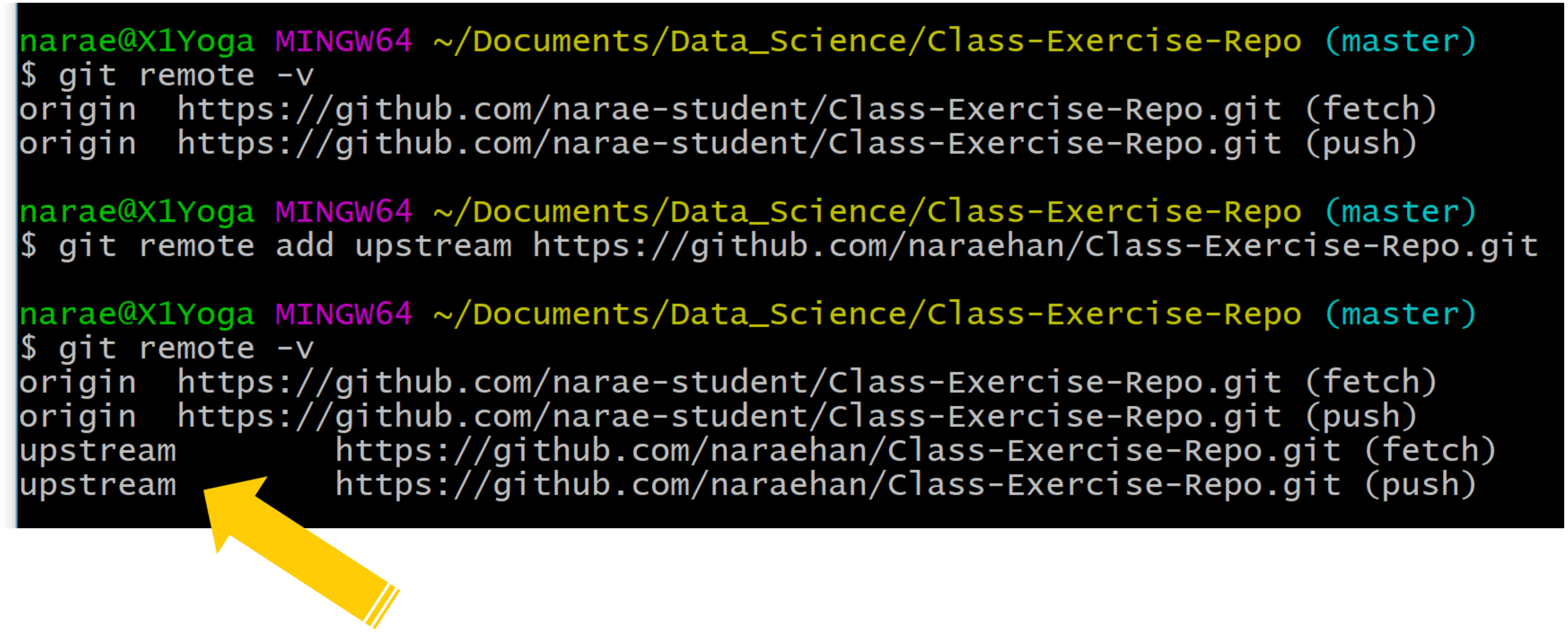


```
narae@X1Yoga MINGW64 ~/Documents/Data_Science/Class-Exercise-Repo (master)
$ git remote -v
origin  https://github.com/narae-student/Class-Exercise-Repo.git (fetch)
origin  https://github.com/narae-student/Class-Exercise-Repo.git (push)
```

- Configure the original repo as another remote: "upstream"
  - `git remote add upstream <GitHub-repo-URL>`
- 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`

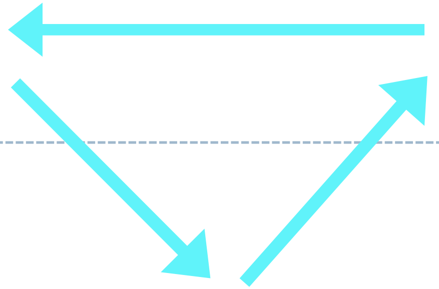
You might be able to leave out "origin master".

Two remotes: "origin", "upstream"



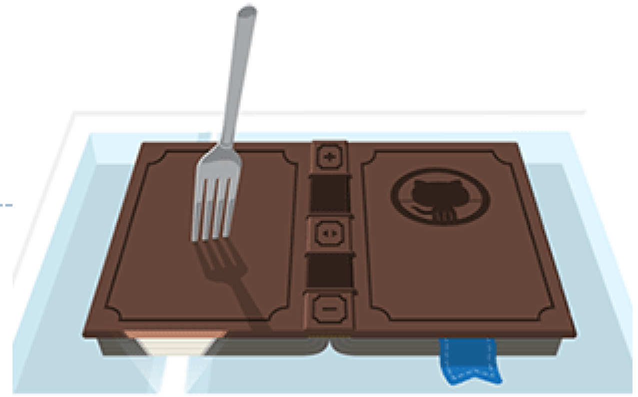
```
narae@X1Yoga MINGW64 ~/Documents/Data_Science/Class-Exercise-Repo (master)
$ git remote -v
origin  https://github.com/narae-student/Class-Exercise-Repo.git (fetch)
origin  https://github.com/narae-student/Class-Exercise-Repo.git (push)

narae@X1Yoga MINGW64 ~/Documents/Data_Science/Class-Exercise-Repo (master)
$ git remote add upstream https://github.com/naraehan/Class-Exercise-Repo.git

narae@X1Yoga MINGW64 ~/Documents/Data_Science/Class-Exercise-Repo (master)
$ 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/naraehan/Class-Exercise-Repo.git (fetch)
upstream        https://github.com/naraehan/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 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.

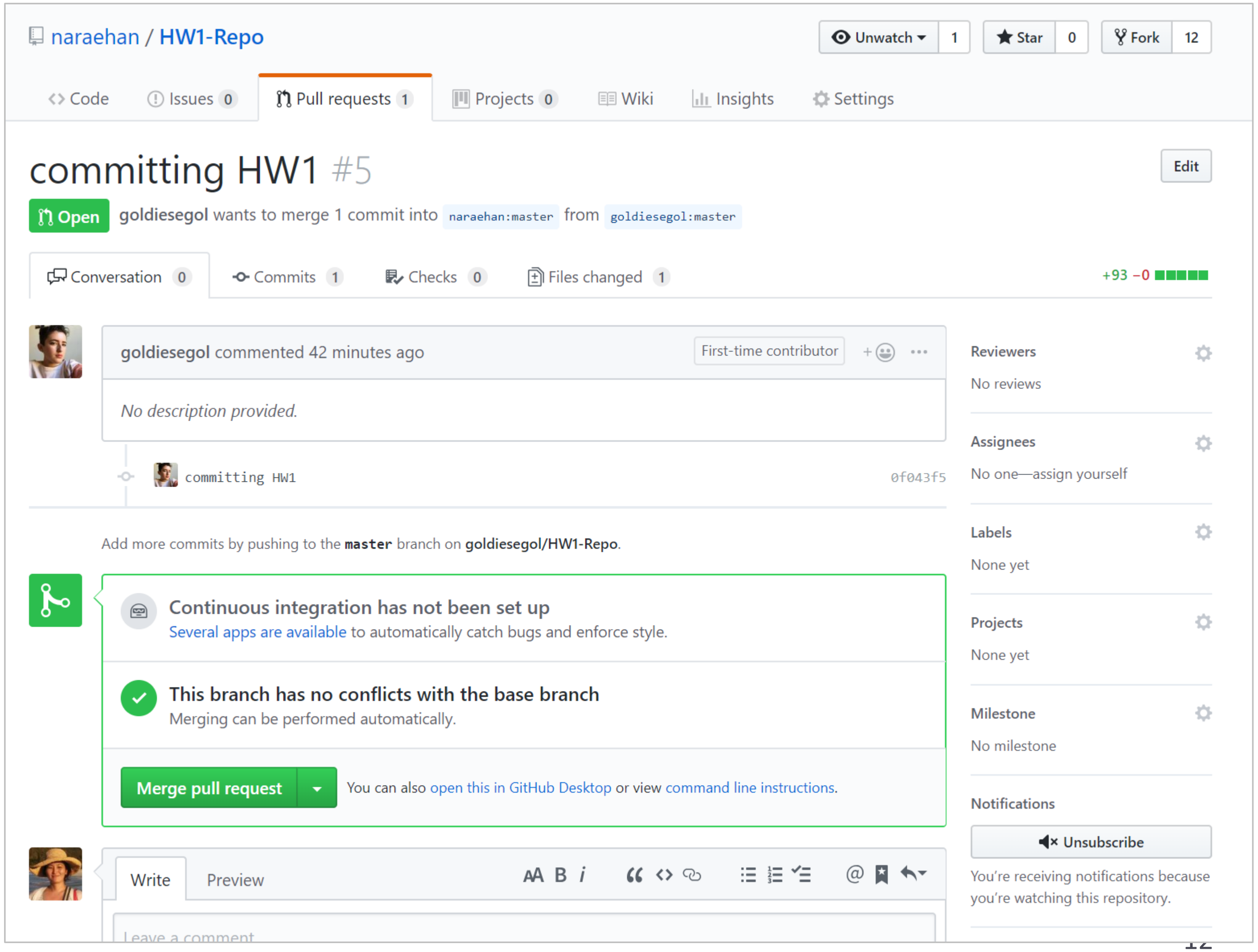
Forking: summary



- 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:



naraehan / HW1-Repo

committing HW1 #5

Open goldiesegol wants to merge 1 commit into `naraehan:master` from `goldiesegol:master`

goldiesegol commented 42 minutes ago First-time contributor

No description provided.

`committing HW1` 0f043f5

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

Continuous integration has not been set up
Several apps are available 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

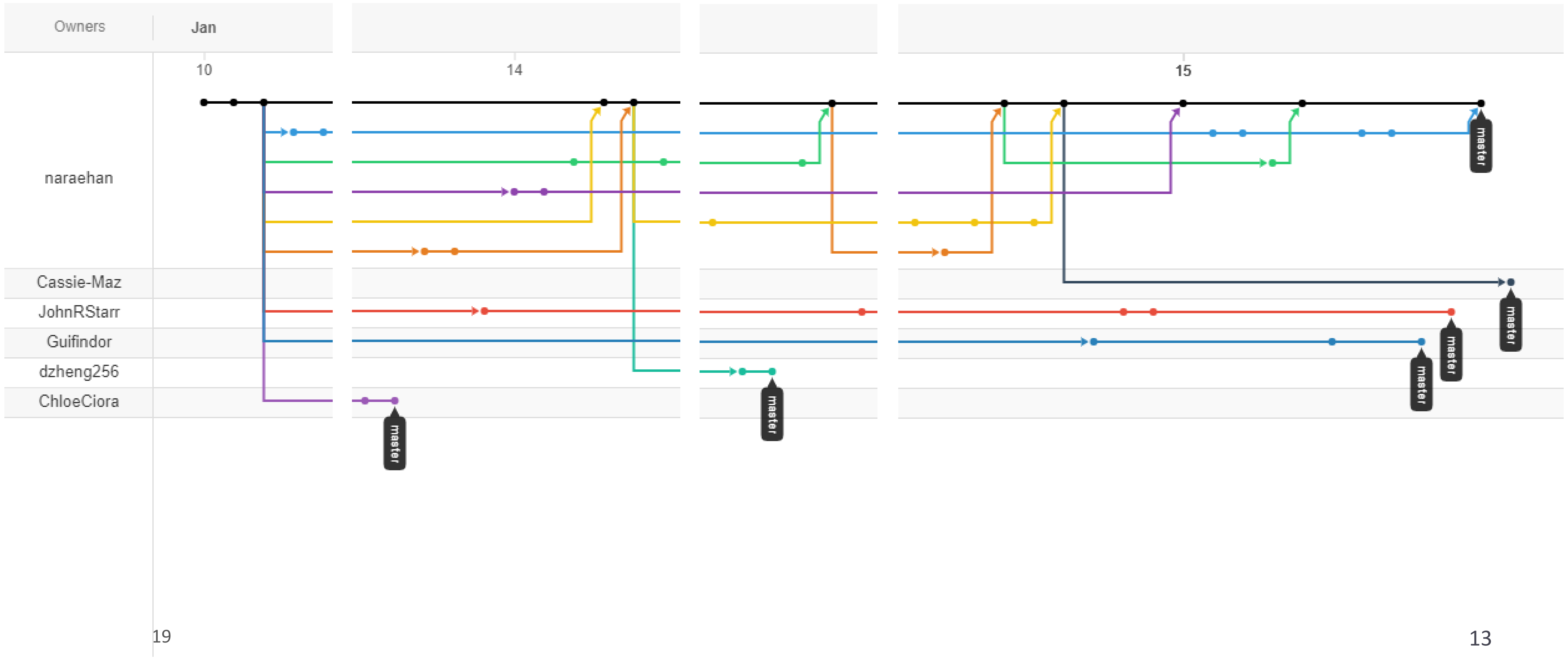
Notifications

Unsubscribe

You're receiving notifications because you're watching this repository.

Many forks and merges

- https://github.com/naraehan/HW1-Repo/network



HW1: sync your HW1-Repo

1. Configure "upstream" remote:

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

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.

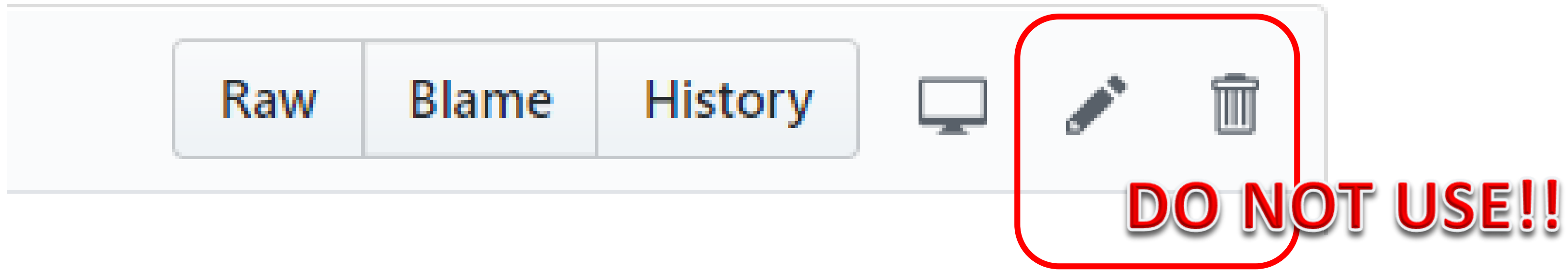
- They are powerful tools.
- There are a lot of abstract, high-level concepts involved.
- Concepts do not make sense before you get hands-on.
- You cannot get hands-on without the right context.

🡐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.

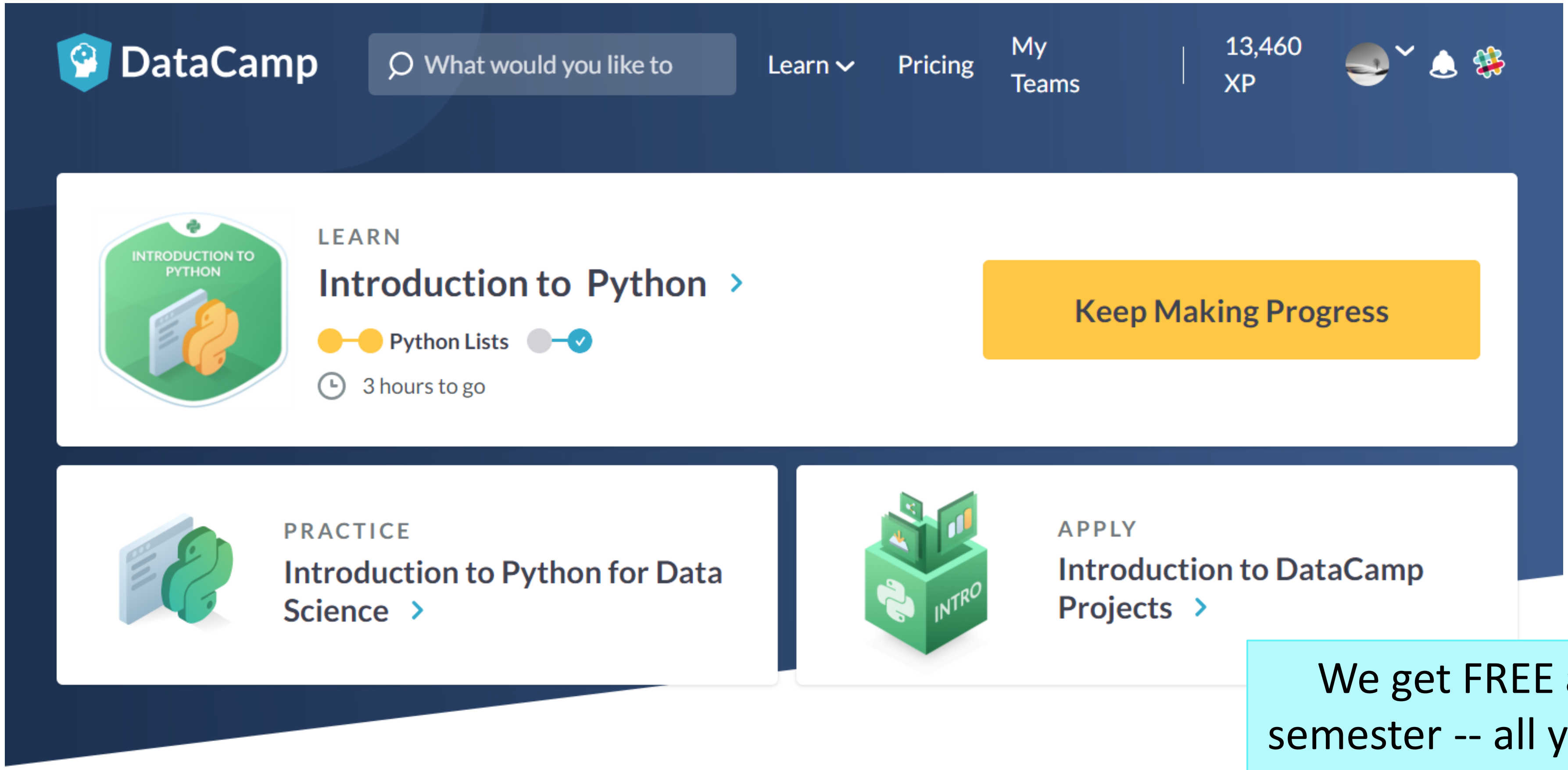
- We will follow some ground rules.
- DO NOT EDIT A REPOSITORY'S CONTENT THROUGH GITHUB.



- 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`.

Course Group on DataCamp

- Video-based, interactive tutorials



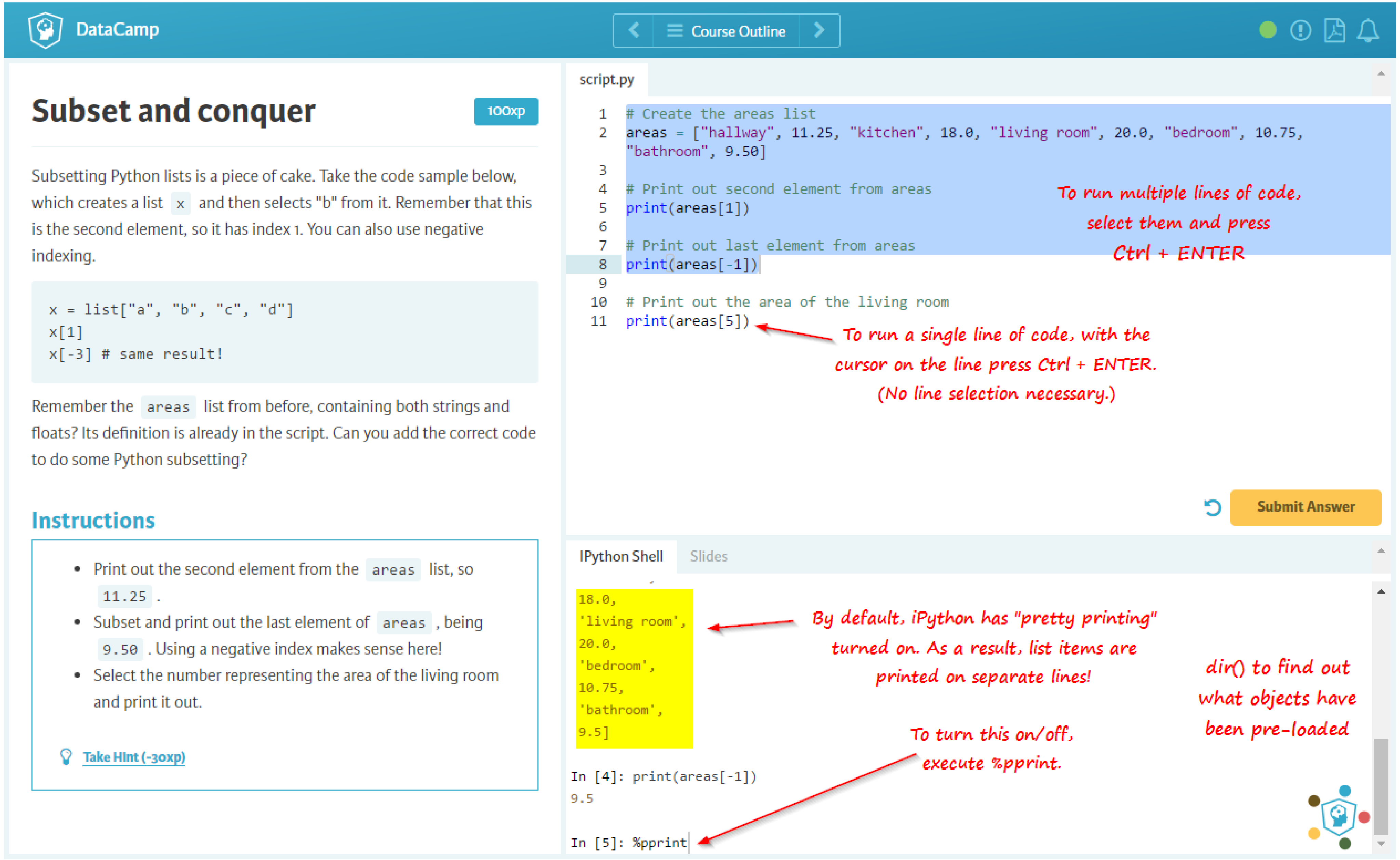
We get FREE access this semester -- all you can learn! Use **Pitt email address** to sign up.

How to use DataCamp

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



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 = 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.

```
# 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.)

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

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

In [5]: %pprint
```

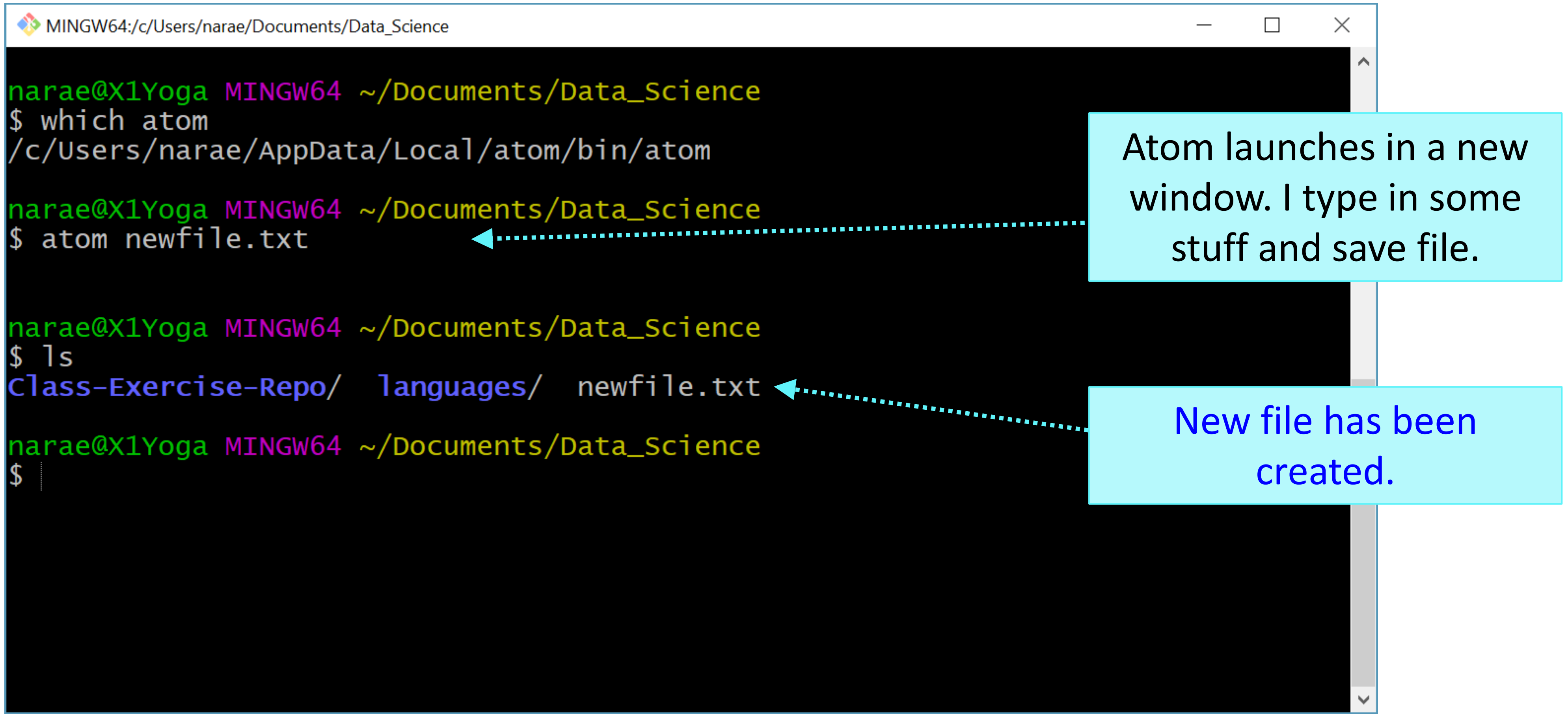
By default, iPython has "pretty printing" turned on. As a result, list items are printed on separate lines!

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.



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

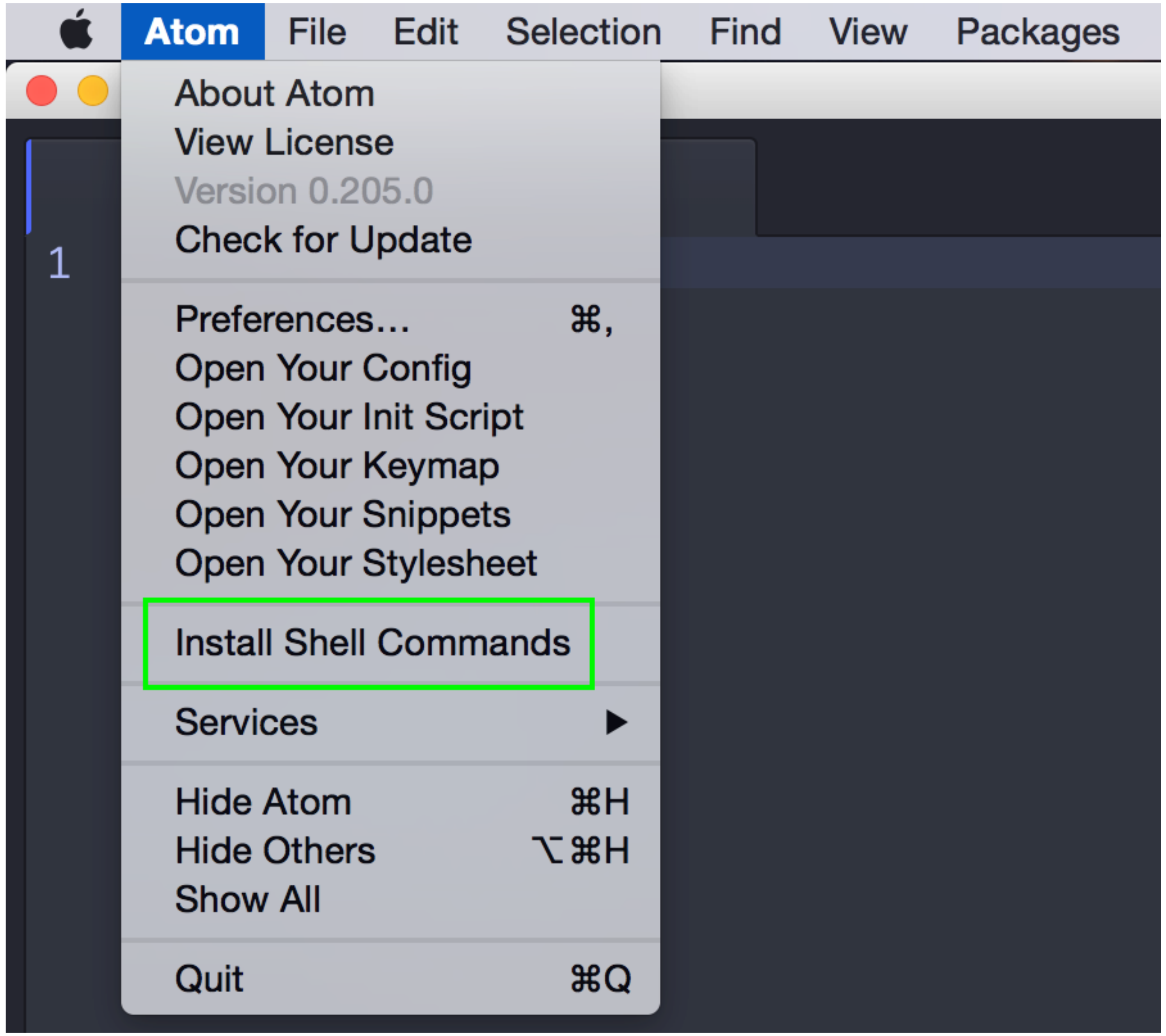
```
narae@X1Yoga MINGW64 ~/Documents/Data_Science
$ which atom
/c/Users/narae/AppData/Local/atom/bin/atom

narae@X1Yoga MINGW64 ~/Documents/Data_Science
$ atom newfile.txt


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

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

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

New file has been created.

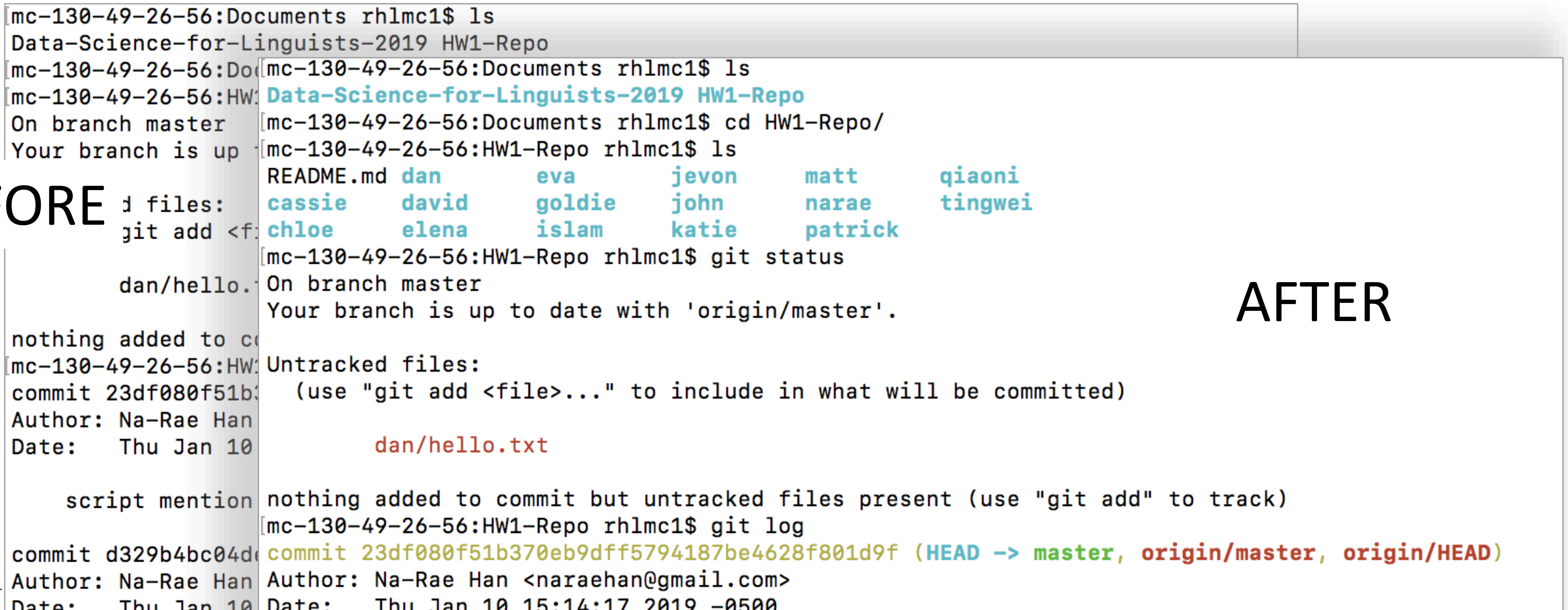
Mac users: configure Atom for shell

- 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.
  - Dan will demonstrate:

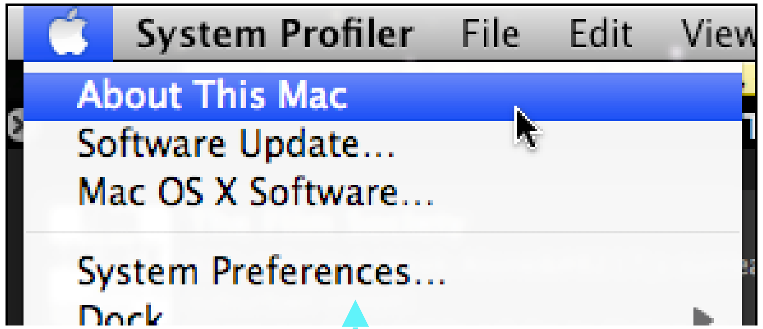


```
mc-130-49-26-56:Documents rhlmc1$ ls
Data-Science-for-Linguists-2019 HW1-Repo
mc-130-49-26-56:Do
mc-130-49-26-56:HW
On branch master
Your branch is up

                d files:
                git add <f

        dan/hello.

nothing added to c
mc-130-49-26-56:HW
commit 23df080f51b
Author: Na-Rae Han
Date:   Thu Jan 10

    script mention

commit d329b4bc04d
Author: Na-Rae Han
Date:   Thu Jan 10
```

```
mc-130-49-26-56:Documents rhlmc1$ ls
Data-Science-for-Linguists-2019 HW1-Repo
mc-130-49-26-56:Documents rhlmc1$ cd HW1-Repo/
mc-130-49-26-56:HW1-Repo rhlmc1$ ls
README.md dan        eva        jevon      matt       qiaoni
cassie    david      goldie     john       narae      tingwei
chloe     elena      islam      katie      patrick
mc-130-49-26-56:HW1-Repo rhlmc1$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        dan/hello.txt

nothing added to commit but untracked files present (use "git add" to track)
mc-130-49-26-56:HW1-Repo rhlmc1$ git log
commit 23df080f51b370eb9dff5794187be4628f801d9f (HEAD -> master, origin/master, origin/HEAD)
Author: Na-Rae Han <naraehan@gmail.com>
Date:   Thu Jan 10 15:14:17 2019 -0500
```

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