Covenant University

Raising a new Generation of Leaders

PET328

COMPUTER APPLICATIONS IN PETROLEUM ENGINEERING

PET328: COMPUTER APPLICATIONS IN PETROLEUM ENGINEERING

(With Python Programming)

python Programming

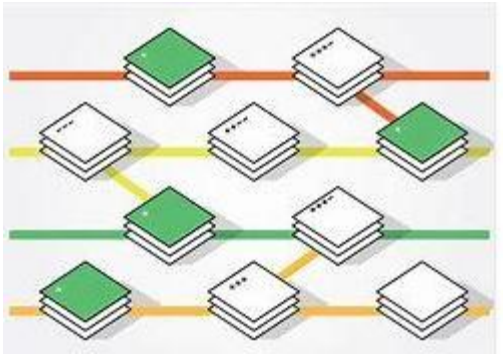
Olatunde O. Mosobalaje (PhD)

Department of Petroleum Engineering,
Covenant University, Ota
Nigeria

PREAMBLES

The Toolbox

- Git and GitHub



Git is an open source version control software.

What is Version Control?

Version control (VC) is a system used for keeping track of changes made to a file over time. As the changes are made, the system records and save the state of the file at instances indicated by the user. Such user can revert back to a previous version of the file when necessary.

Essentially, the VC system keeps the latest version of the file but also keeps a record of all changes between all versions.

PREAMBLES

The Toolbox

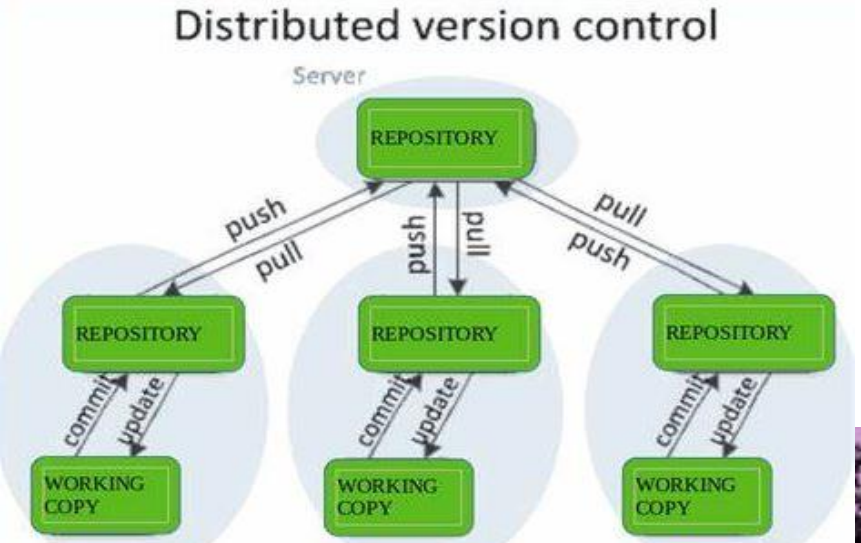
- Git and GitHub

And, there is something called Distributed Version Control (DVC)

What is Distributed Version Control?

Typically, real life projects (including oilfield digital projects) are done by teams whose members need to collaborate – work together on same files. Individual members of the team can make changes to such shared files. There is therefore a need to make such file available on a central server and to keep track of the following:

- who made what change?
- When was the change made?
- Why was the change made?



The Toolbox

Git and GitHub

And, there is something called Distributed Version Control (DVC)

What is Distributed Version Control?

A version control system that also comes with the capabilities for collaboration among several people is known as Distributed Version Control system.

Git is a version control system – locally hosted on your system.

GitHub is an online platform that interfaces with Git, hosting your files on remote servers thereby making them available for collaboration with others.

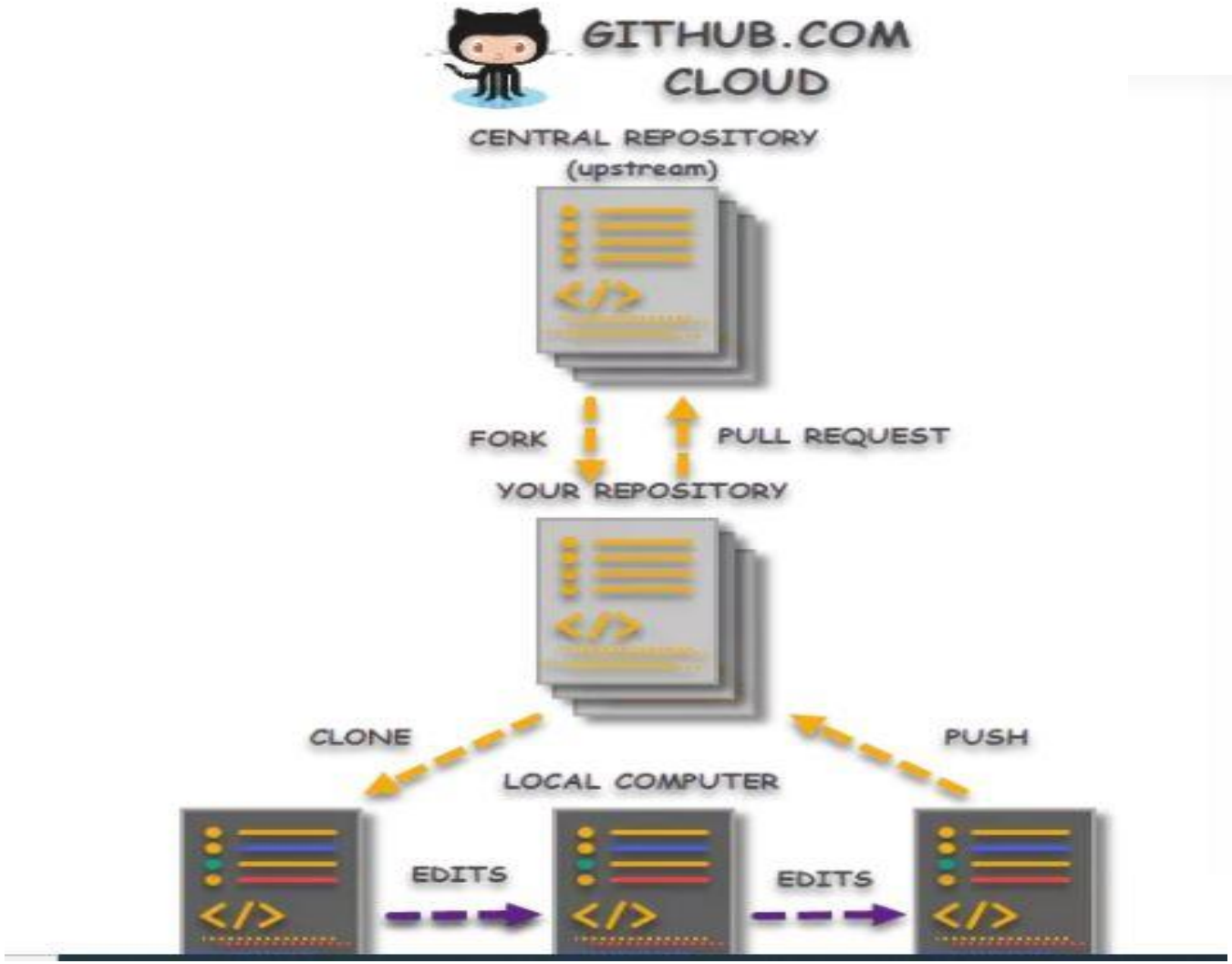
PREAMBLES

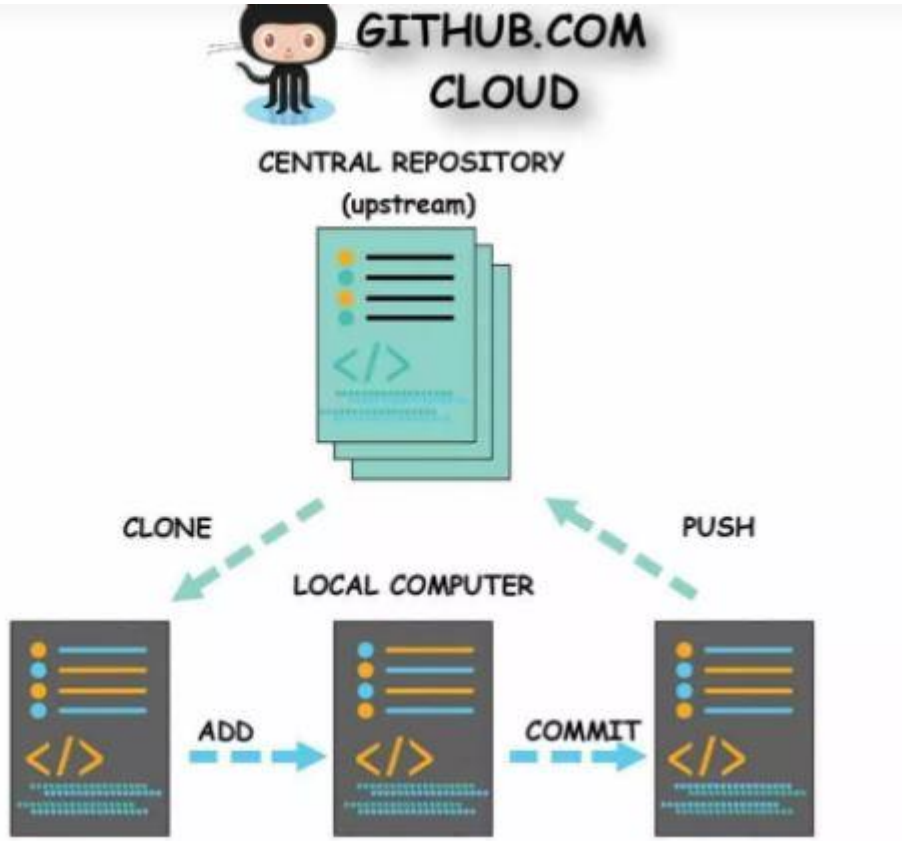
The Toolbox

Git and GitHub

In this course, we shall be working as a team, therefore, both Git and GitHub are part of tools we shall be using. Essentially, submissions to some assignments shall be in the form of code file editing and sharing between students and the Course Instructor.

COLABORATION WORKFLOW

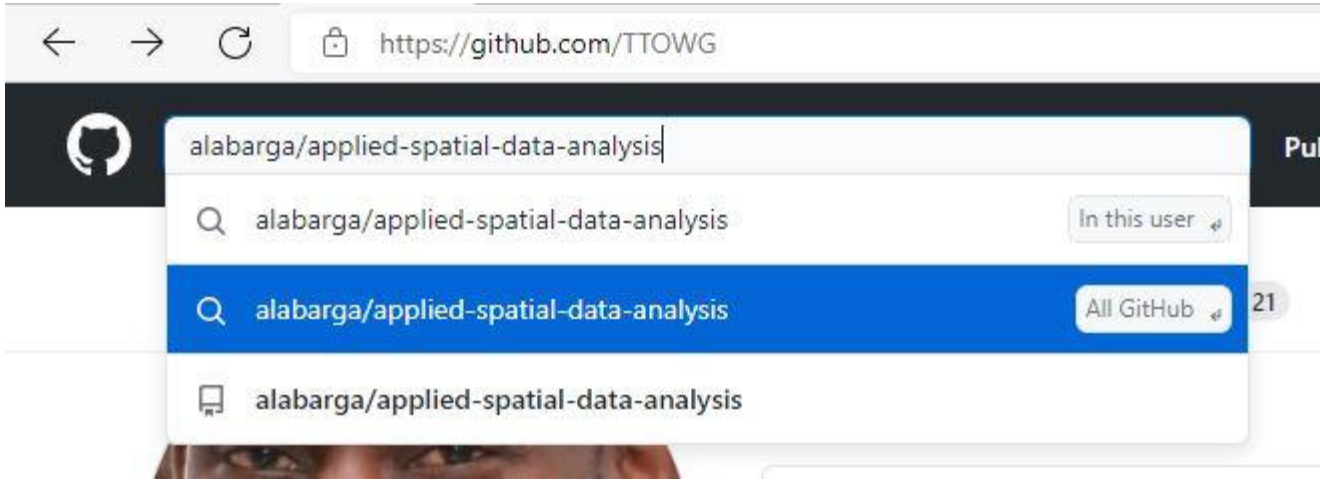


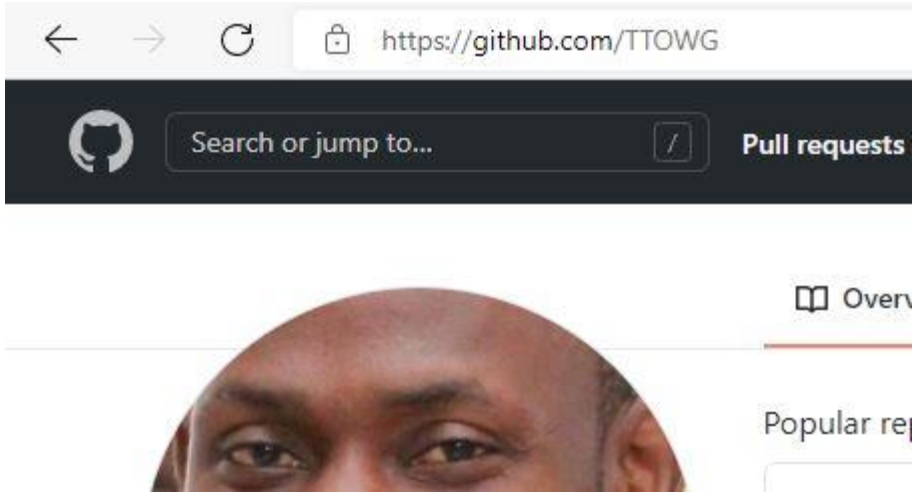


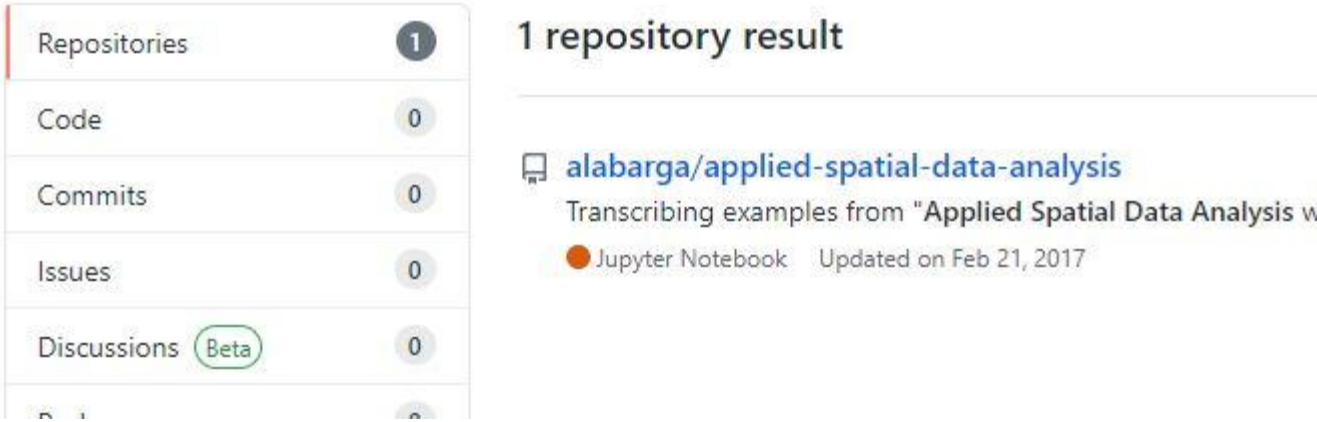
COLABORATION WORKFLOW

Searching for the repo

- Log in to your GitHub account in a browser window.
- Type in the name of the desired repository in the search box e. g, *TTOWG/PET328_2021_Class*.
- Choose to search All GitHub
- Click on the repository among the search results.



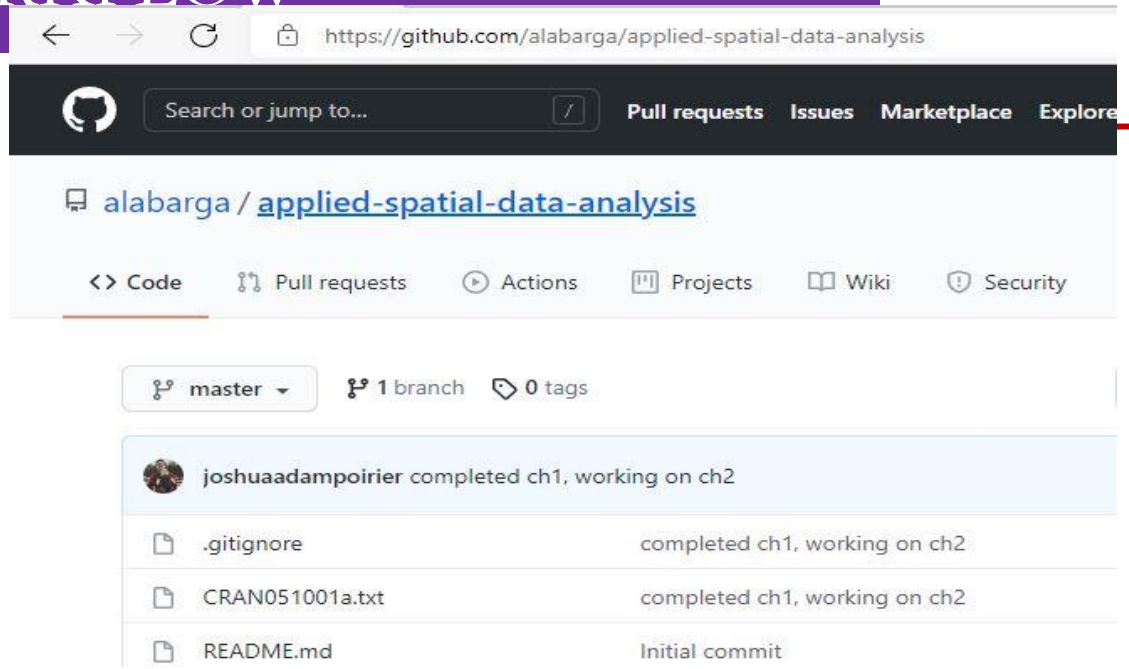


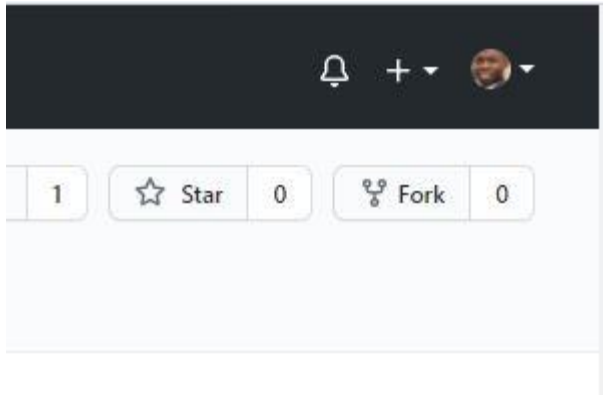


COLABORATION WORKFLOW

Forking the repo

- You will be directed to the GitHub page of the clicked repository (repo).
- This is the original repository; it is currently hosted on the author's GitHub account.
- You want to collaborate on this project, so you need to copy (fork) the repo to your GitHub account.
- Fork the repo by clicking on the 'Fork' icon at the top-right corner of the repo page.

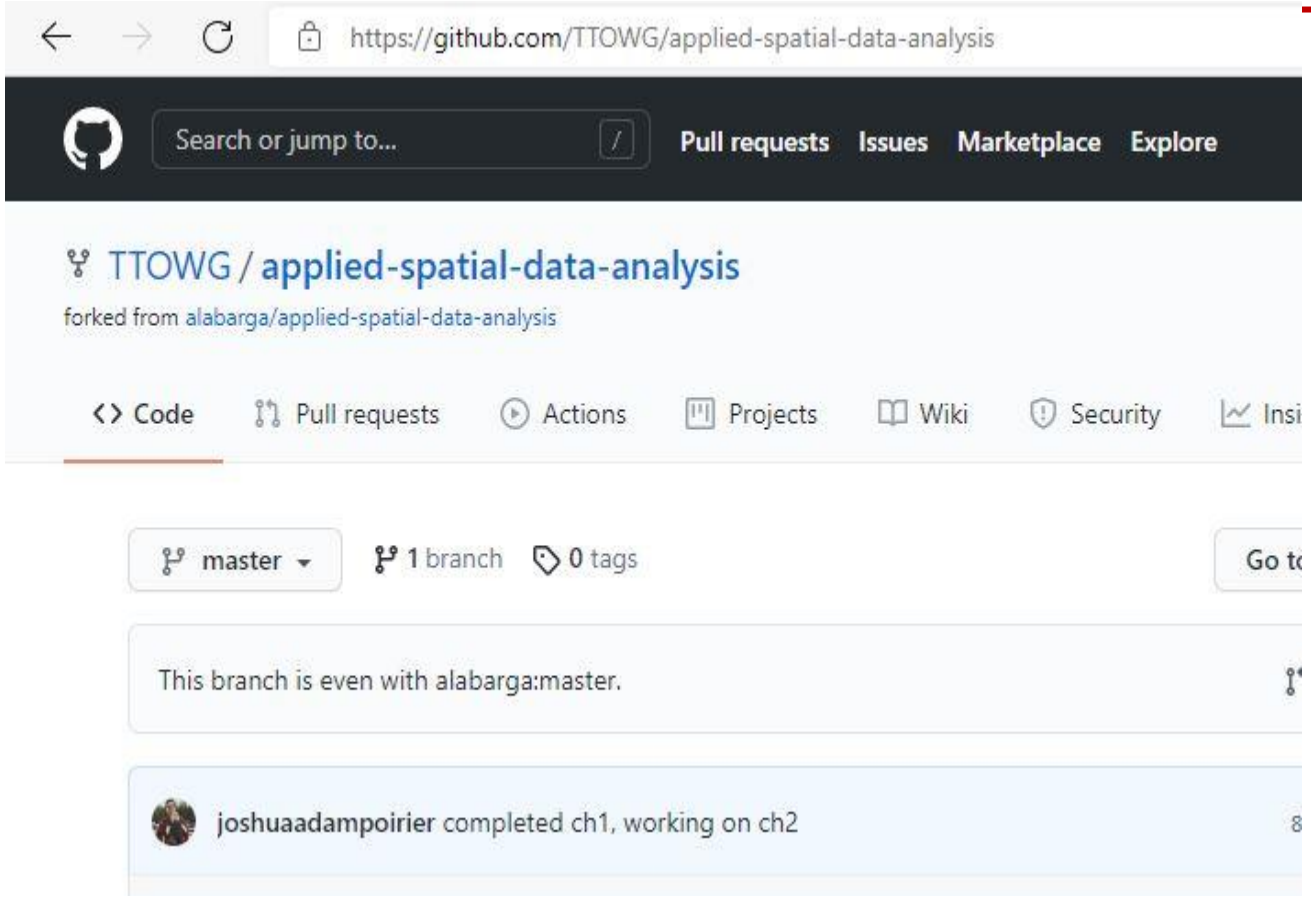




COLABORATION WORKFLOW

Forking the repo

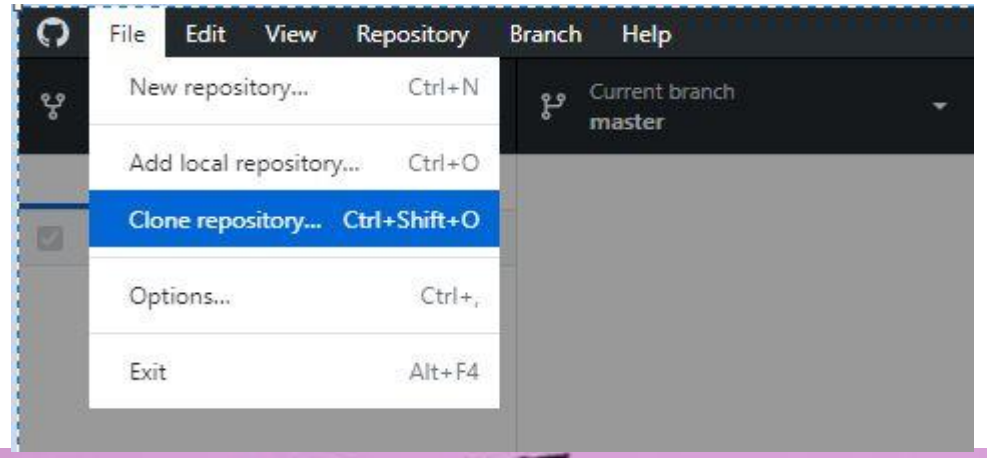
- Once the repo is thus forked, it is copied to your GitHub account; it becomes your repo, although you are not the authour.
- Notice that the name of the forked now has your username; a proof that this copy is now yours.
- However, this copy is only on your account on GitHub's remote server.
- You need to download (clone) it into your PC as a folder so you can work with it offline.

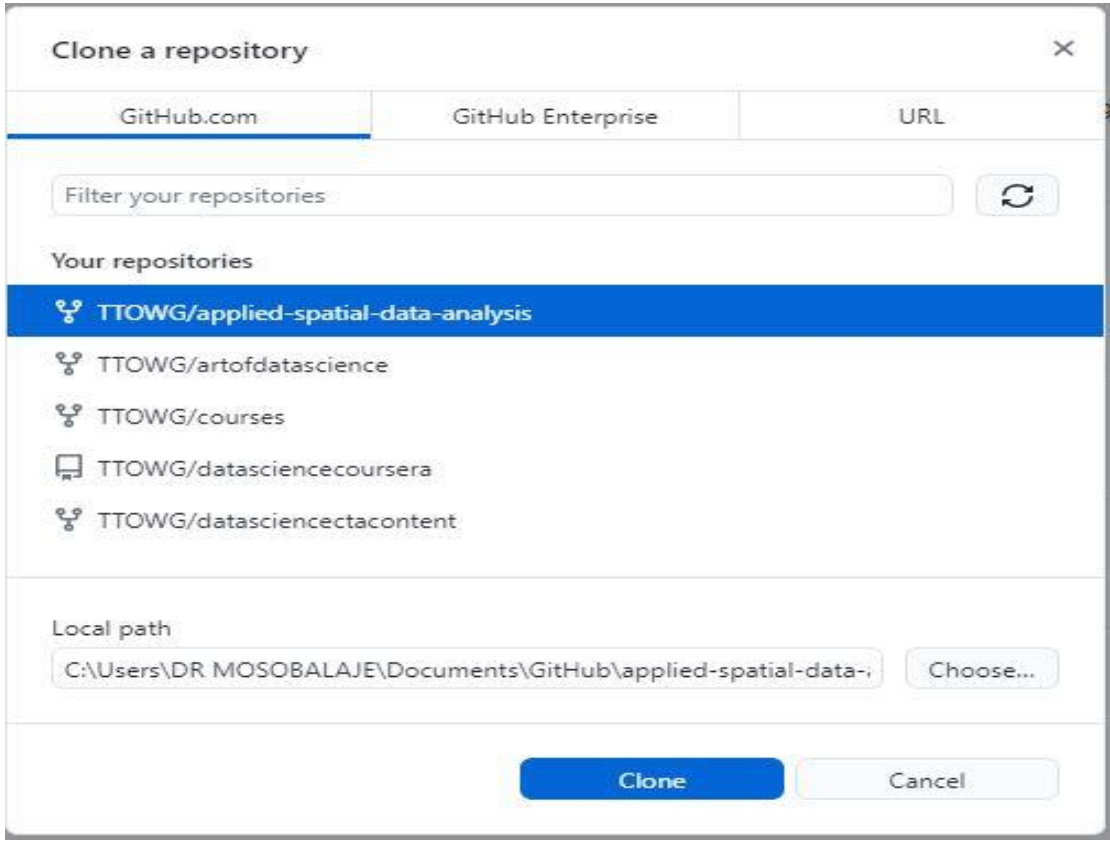


COLABORATION WORKFLOW

Cloning the repo

- At this point, you need to open the GitHub Desktop app on your PC.
- From the file menu, choose *Clone repository*.
- Choose the repo from the list in the dialog box.
- Take note of the location (Local path) where the download folder (repo) would be located.





COLABORATION WORKFLOW

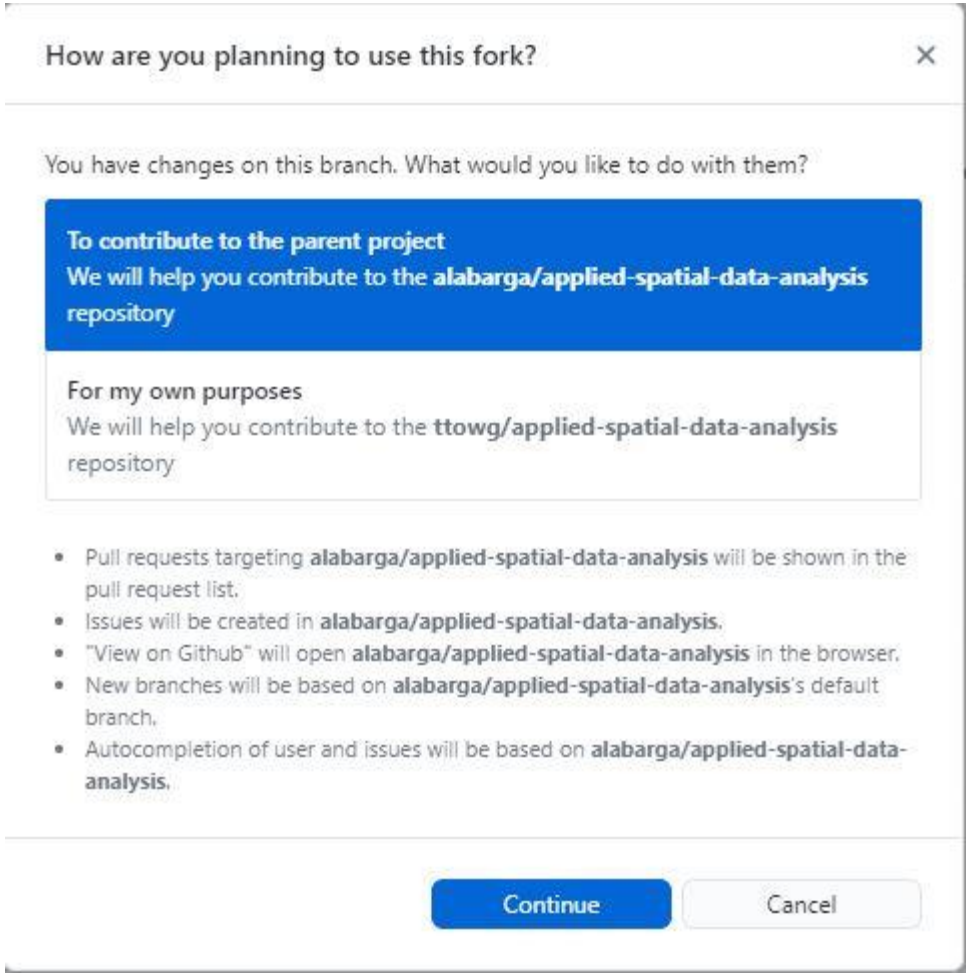
Cloning the repo

ted version of GitHub Desktop is available and will be installed at the next launch. See what's new or restart GitHub Desktop.

COLABORATION WORKFLOW

Cloning the repo

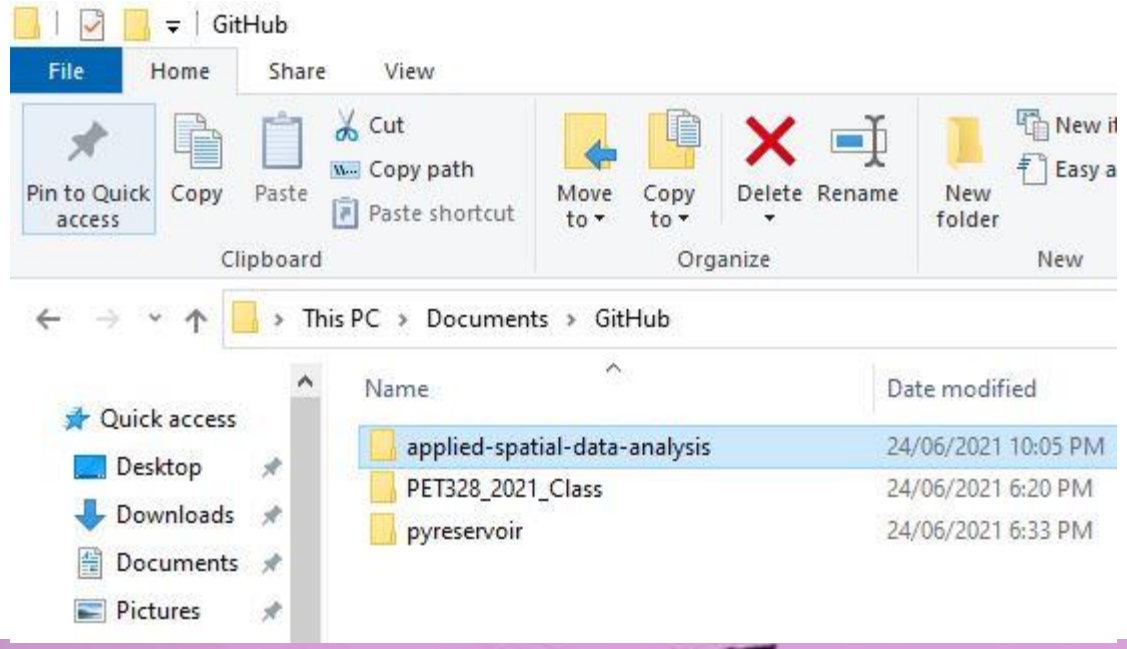
- Choose *To contribute to the parent project*. This would enable you to send your changes to the original repo.
- Click *Continue*.

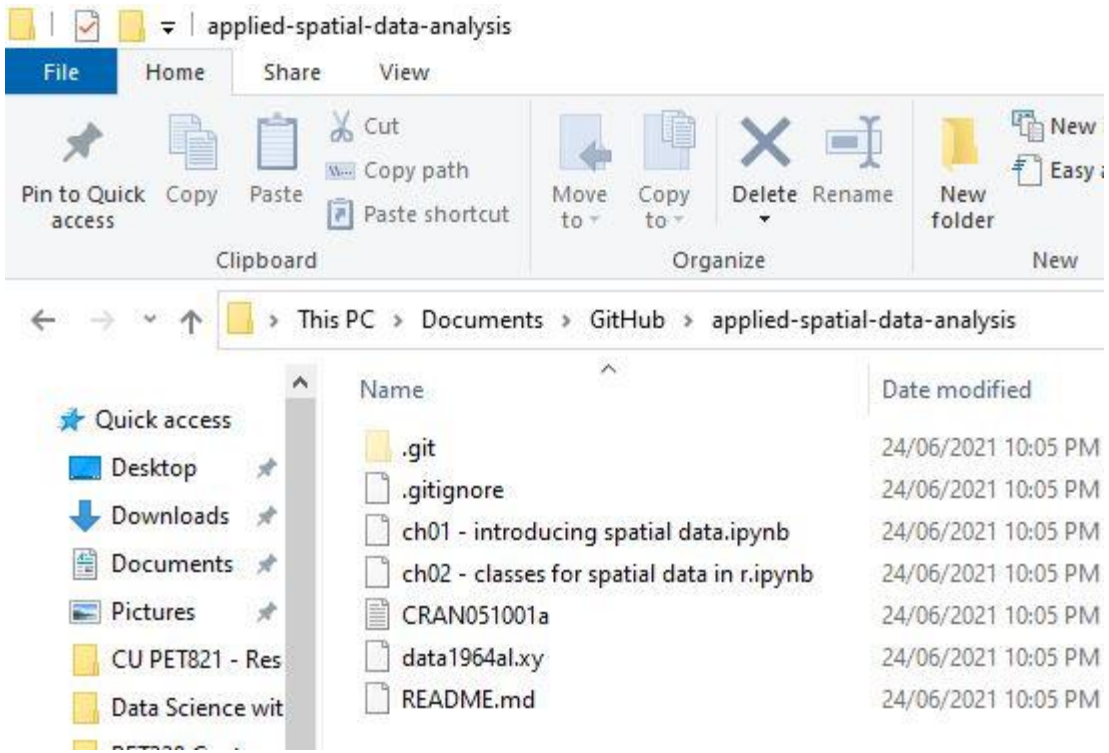


COLABORATION WORKFLOW

Cloning the repo

- Check the Local path (folder) earlier indicated to confirm that the repo is downloaded there.
- Open the cloned repo to see its contents.

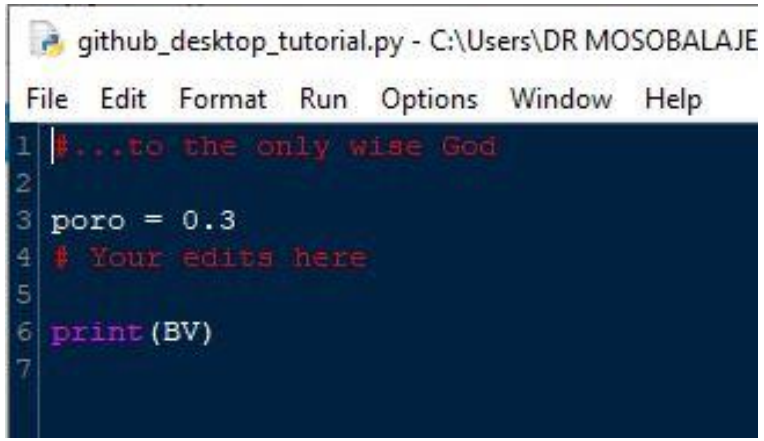


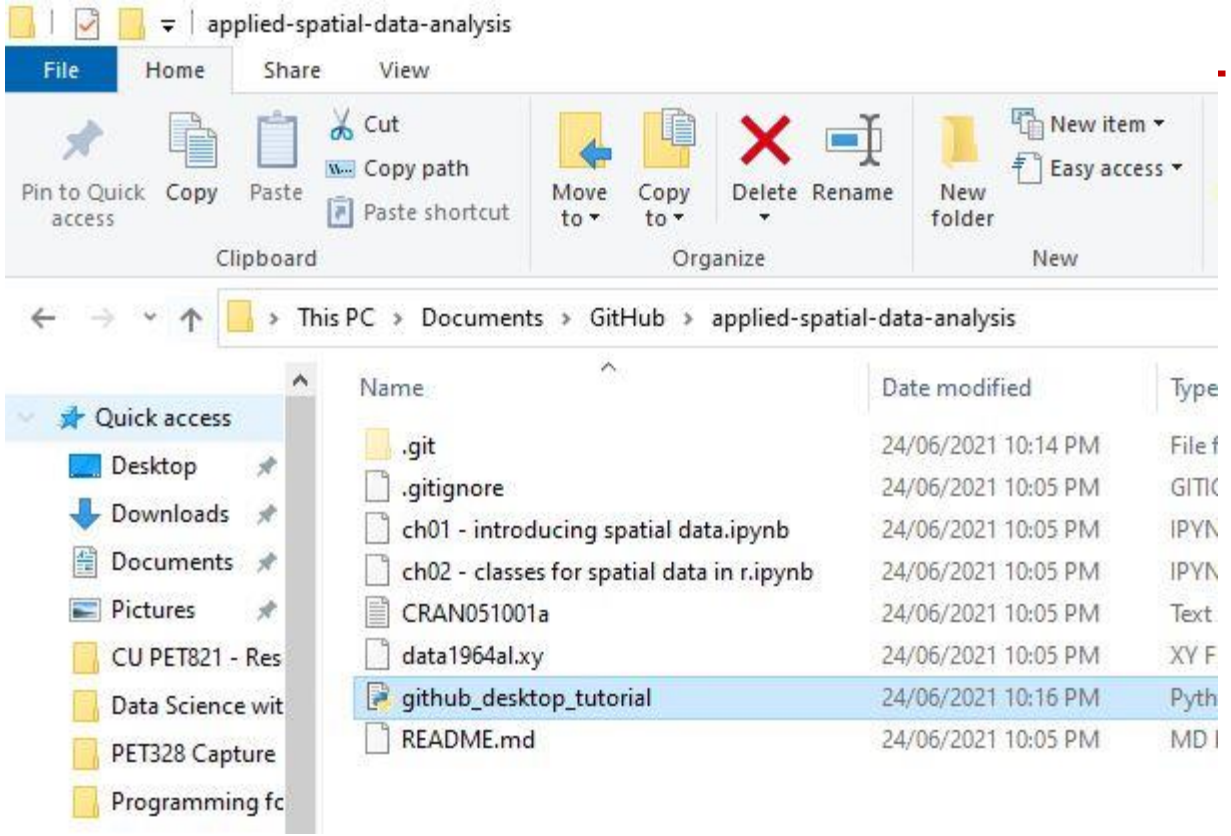


COLABORATION WORKFLOW

Making changes to the repo

- You could add a new file (say a Python code editor, i.e. a .py file) to the cloned repo.
- Here a new file named *github_desktop_tutorial.py* is created with Python IDLE and saved into the cloned repo folder.

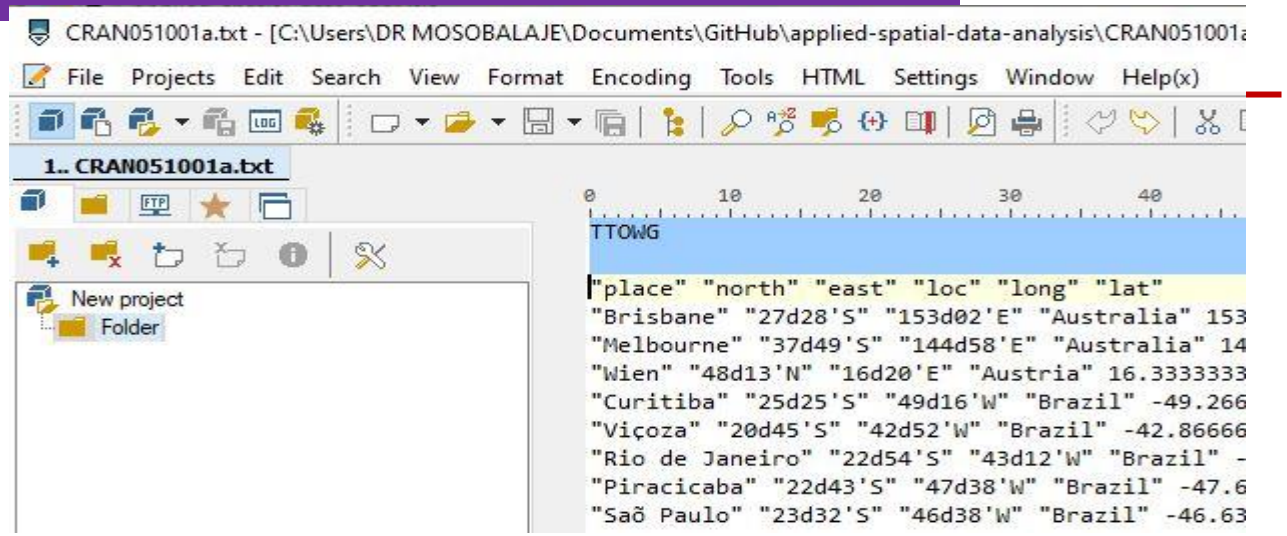


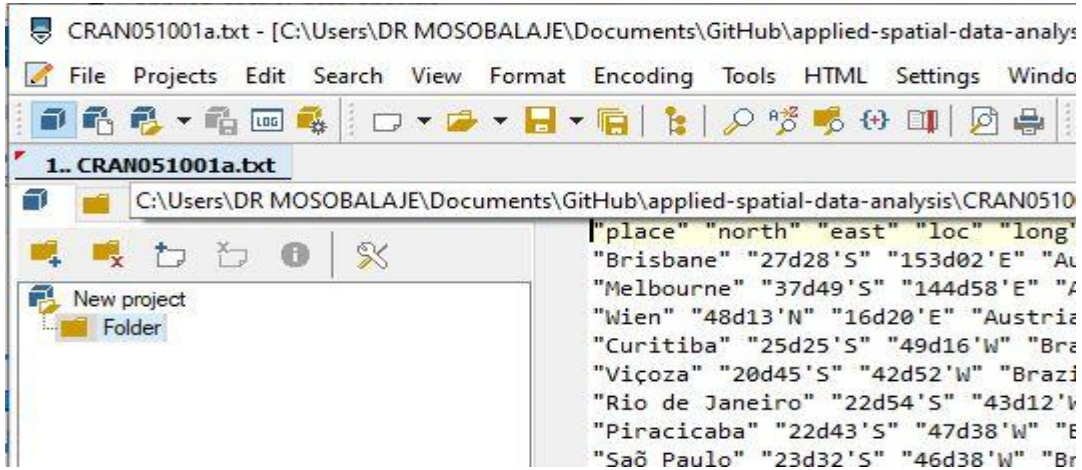


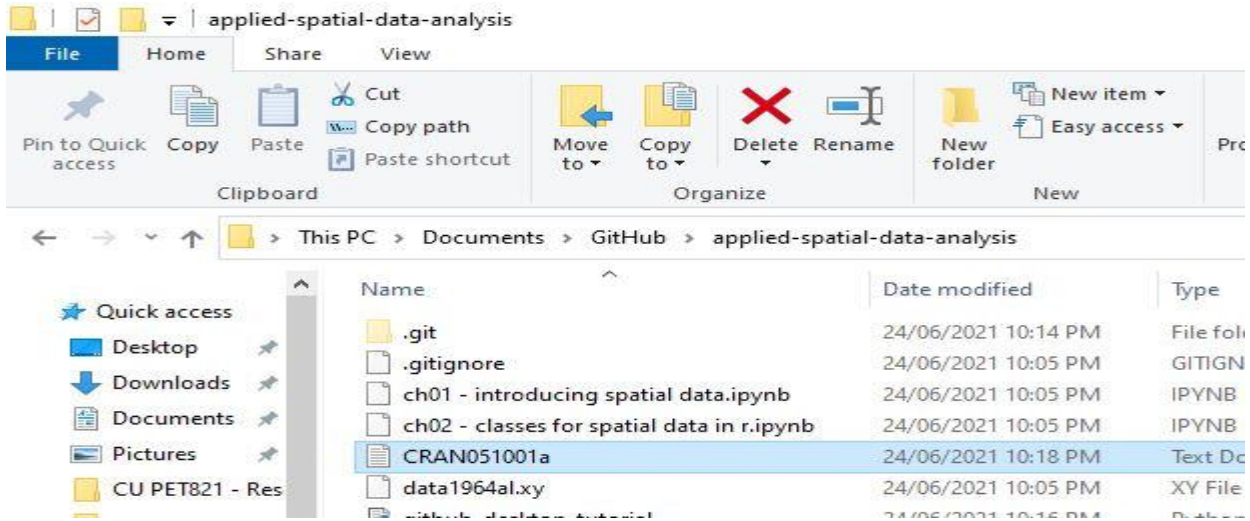
COLABORATION WORKFLOW

Making changes to the repo

- You could also edit an existing file in cloned repo.
- Here, the file *CRAN05100a.txt* existing in the repo is edited.
- A line with the text 'TTOWG' is added.
- A blank line is also added.



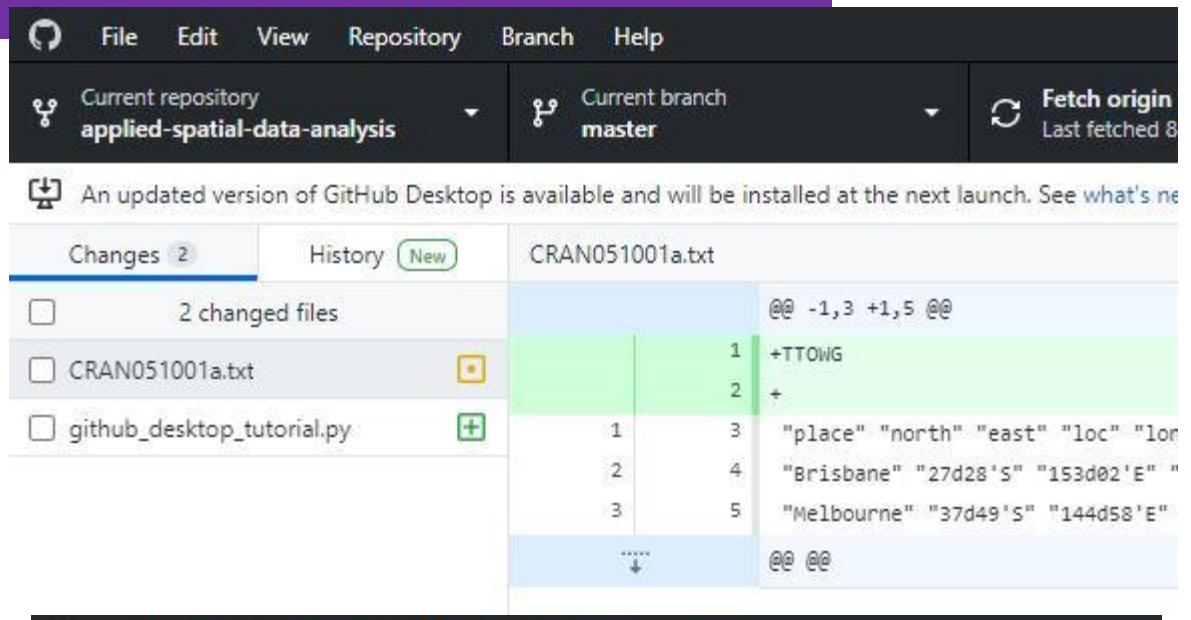


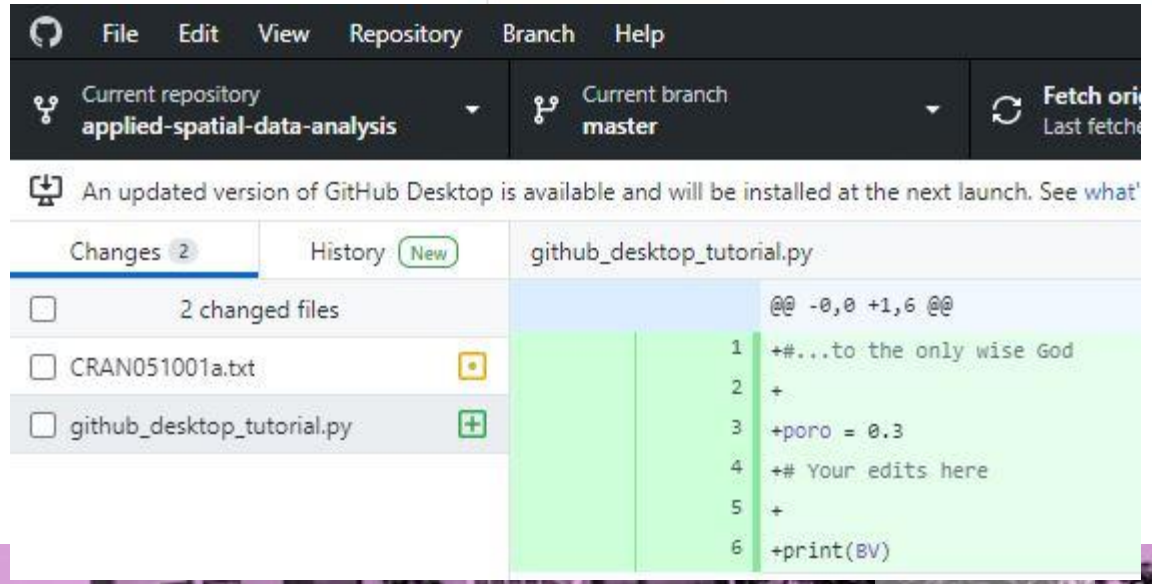


COLABORATION WORKFLOW

Making changes to the repo

- At this point, you can return to the GitHub Desktop app to view the changes you made.
- The changed/added files are listed at the left pane and the specific changes are highlighted in green colour in the right plane.





COLABORATION WORKFLOW

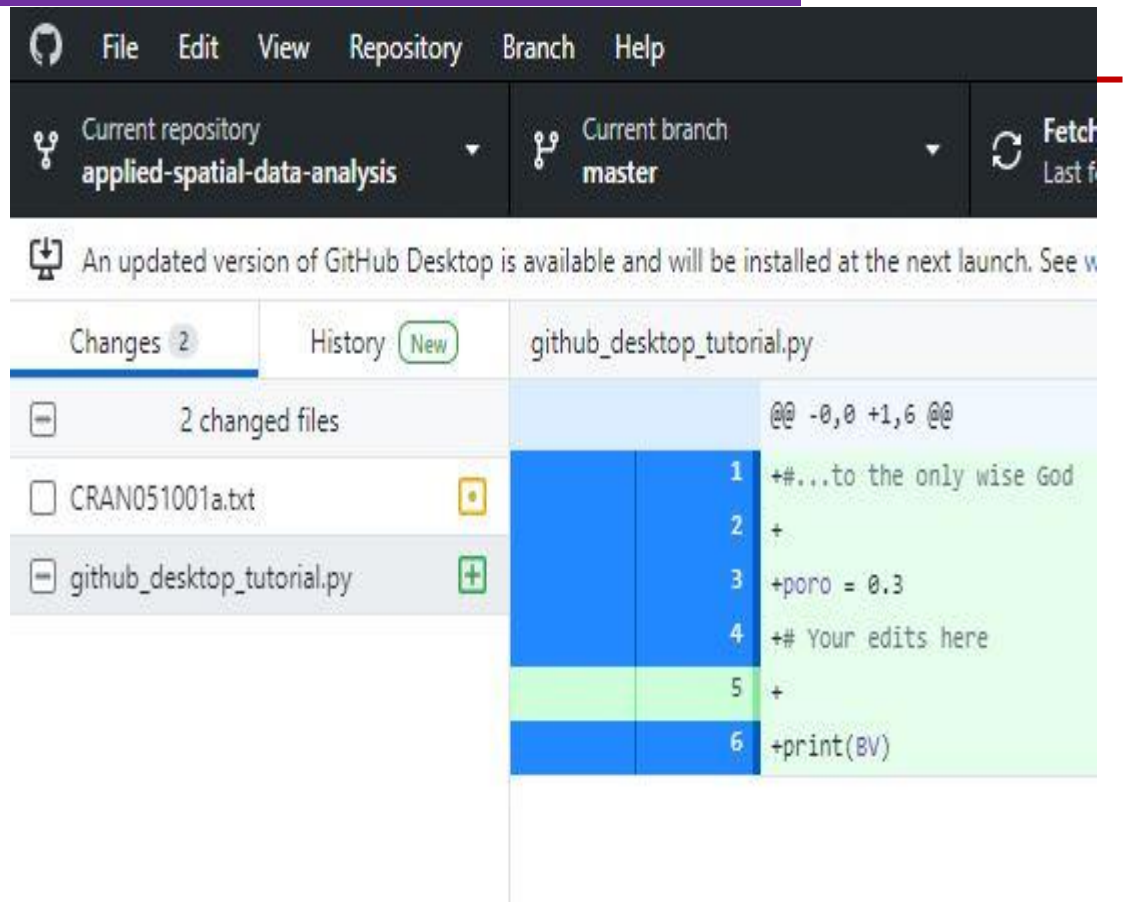
Making changes to the repo

- These changes are made only your local copy, i.e. on the cloned repo.
- The changes are not yet reflected on Git's copy of the repo; of course, not yet reflected on your GitHub account
- So, the changes need to be propagated to Git and then to your GitHub account.
- A process known as *commit* is needed for the first propagation while *push* is needed for the second propagation.

COLABORATION WORKFLOW

Committing a change

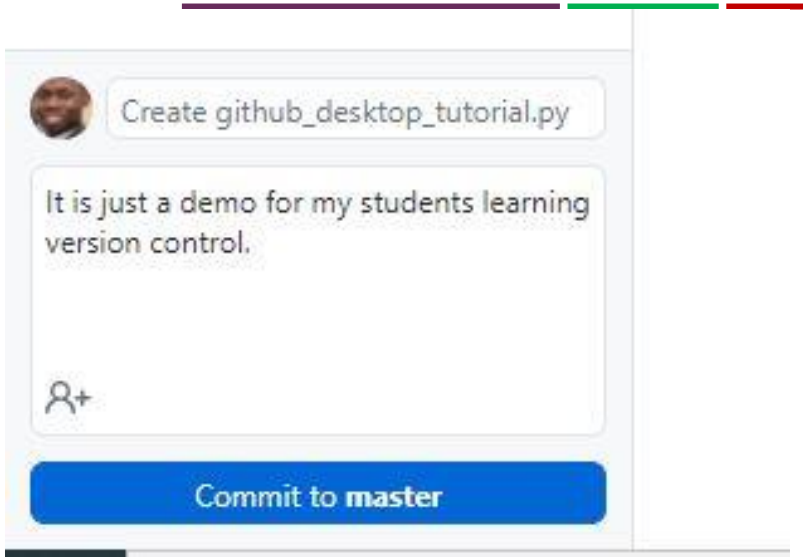
- You don't have to commit all listed changes; you may commit some and leave some uncommitted. However, only committed changes would be propagated to Git and would be later pushed to GitHub.
- Check the box beside the changed file that you would like to commit.
- Specifically, the changes highlighted in blue in the right pane shall be committed; you may deselect lines you don't wish to commit yet.
- Deselected lines are highlighted in green.



COLABORATION WORKFLOW

Committing a change

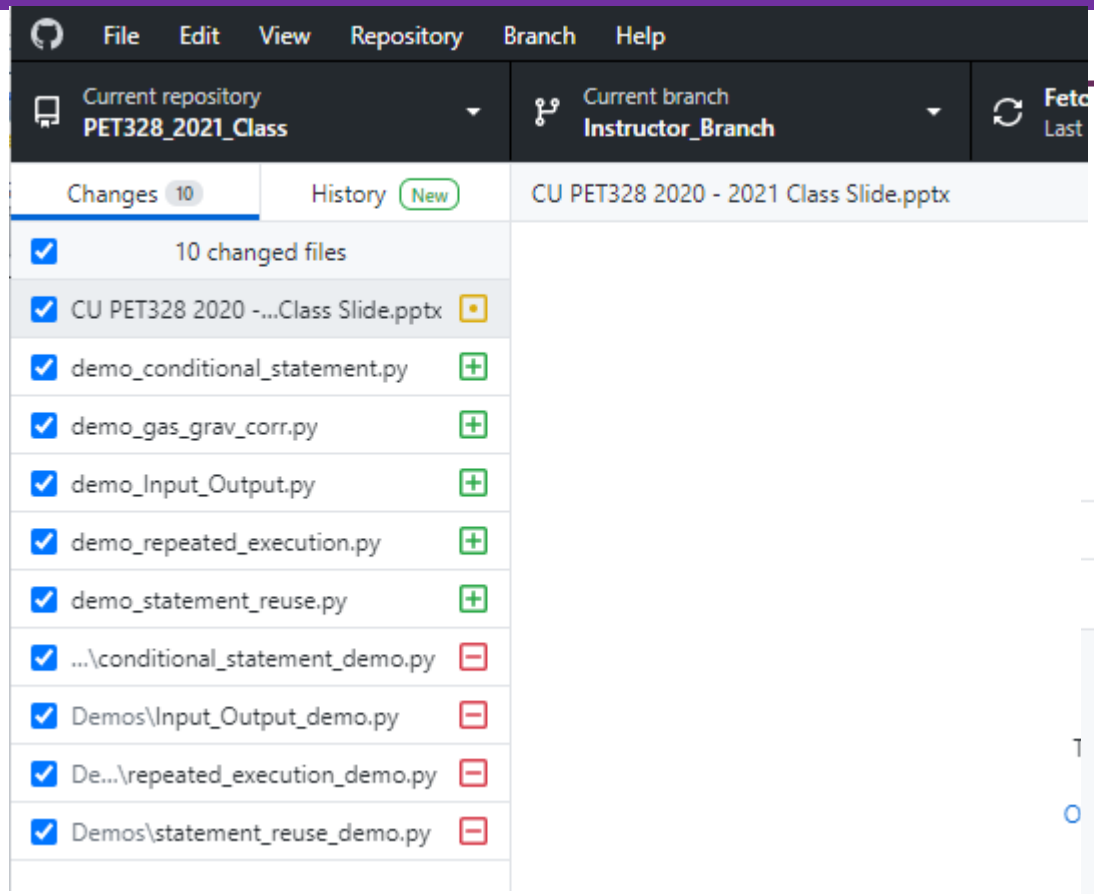
- Once changes to be committed are selected, you should enter a short commit message and commit description in text boxes located at the lower left of the app window.
- The commit message would be listed on GitHub repo to inform you and your collaborators of the purpose of the change you made.
- Click *Commit*.

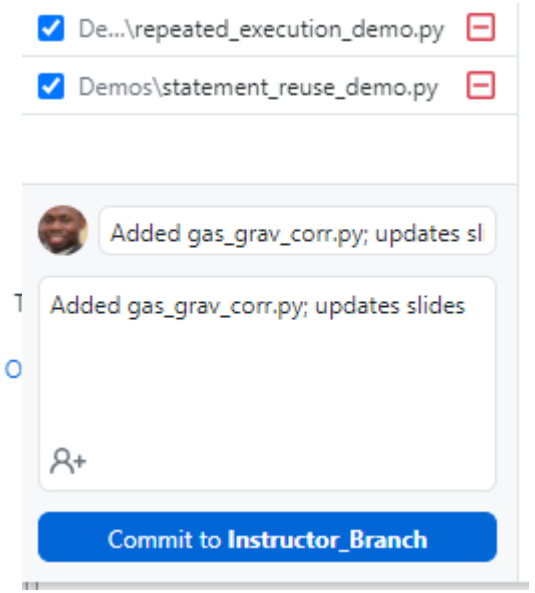


COLABORATION WORKFLOW

Committing a change

- Another example!

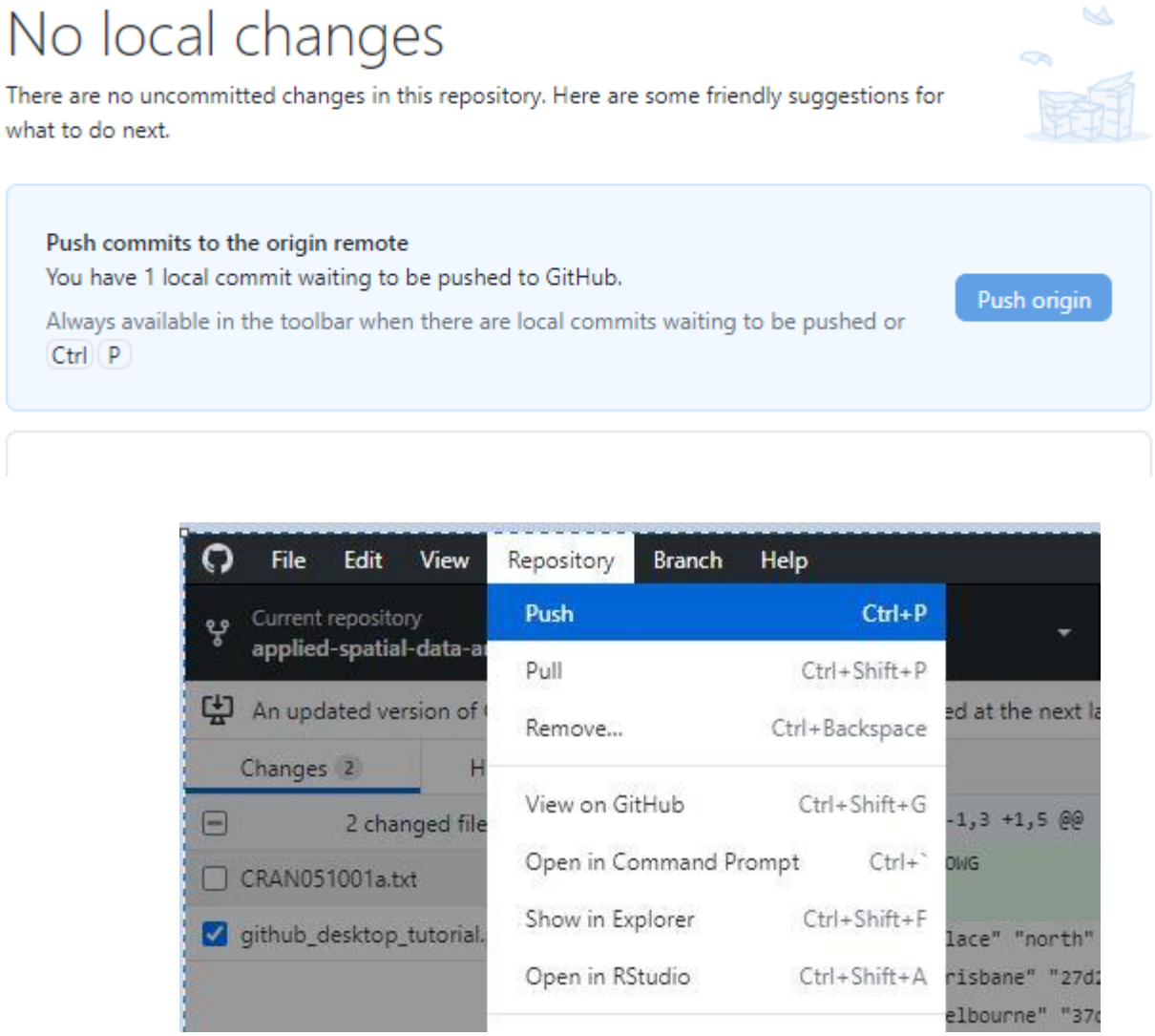




COLABORATION WORKFLOW

Pushing a commit

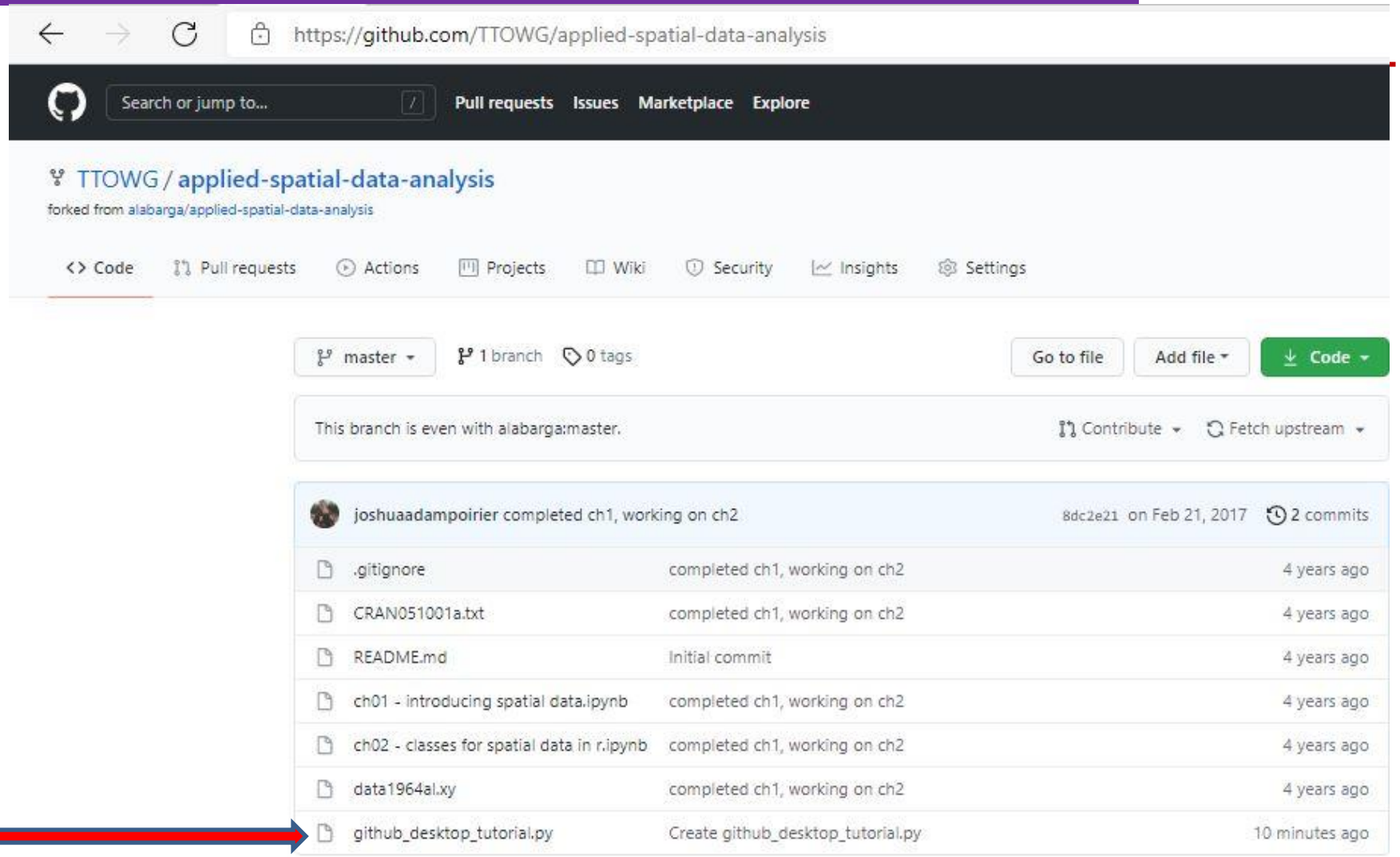
- Once changes are committed, they are referred to as commits.
- As soon as you make a commit, the GitHub Desktop app displays a small dialog box offering you a chance to push the commit to your GitHub account.
- Alternatively, you can choose the *Push* command from the *Repository* menu.



COLABORATION WORKFLOW

Pushing a commit

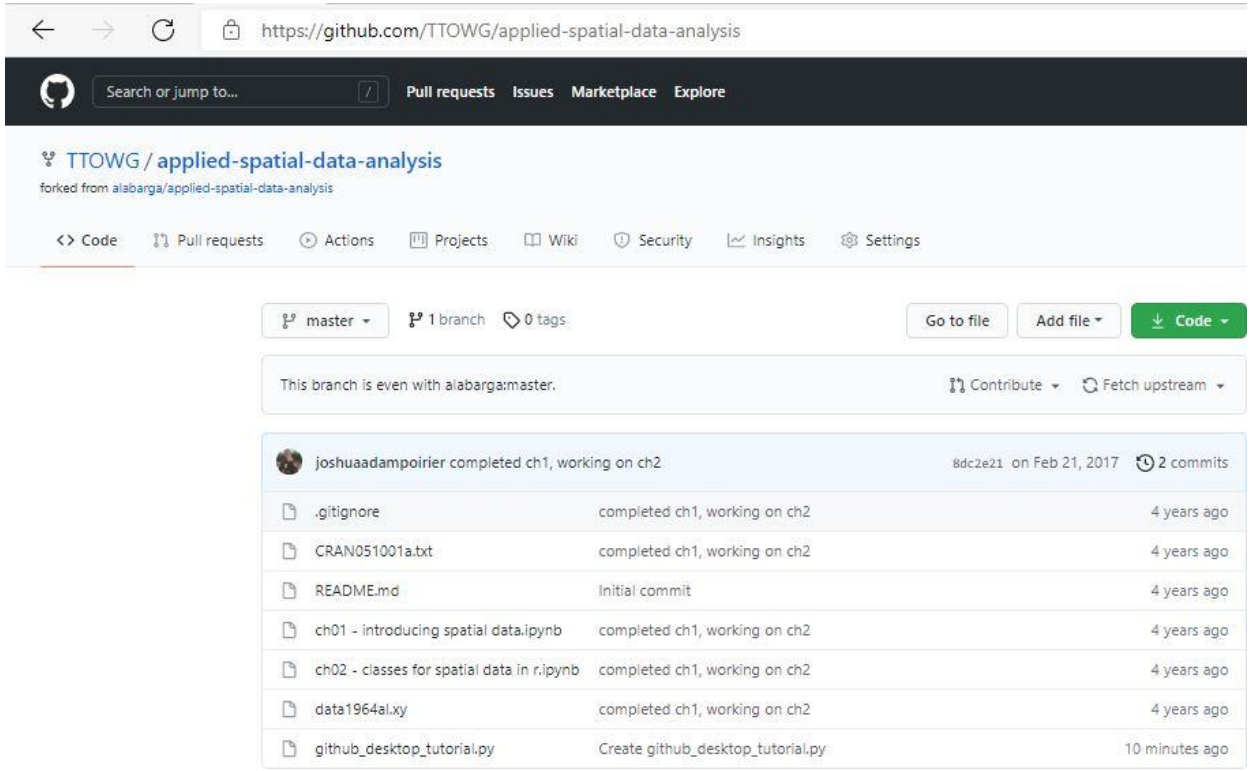
- Once commits are pushed, the repo page on your GitHub account reflects the pushed change.
- Notice that the newly added file (*github_desktop_tutorial.py*) is now listed in the repo page.



COLABORATION WORKFLOW

Pushing a commit

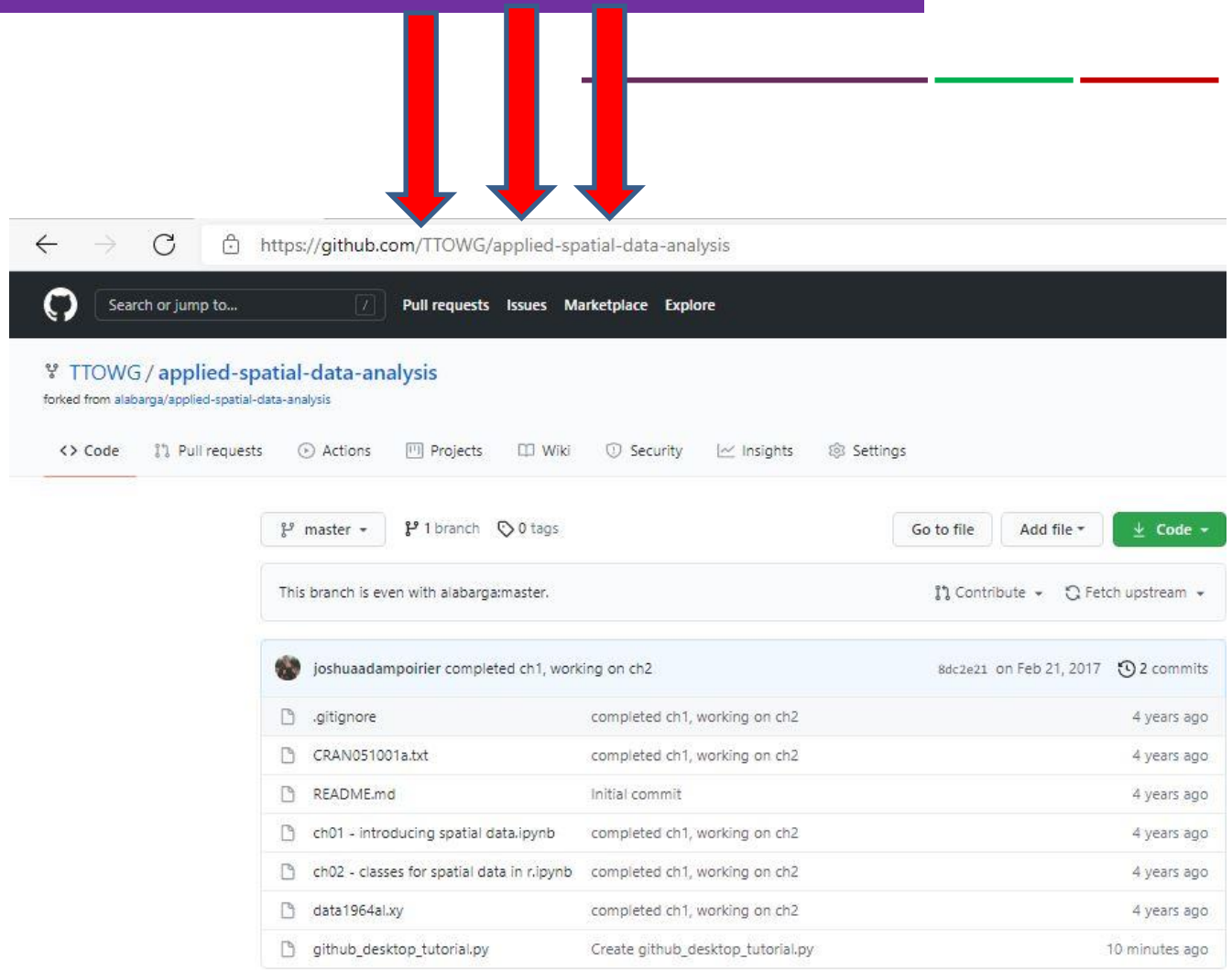
- Note that changes are only pushed to the copy of the repo you forked to your GitHub account.
- The original repo in the author's account is not yet updated to include the changes.
- You would need to notify the author of the changes in a form of a proposal – you would need to create a pull request.



COLABORATION WORKFLOW

Submitting repo URL for grading

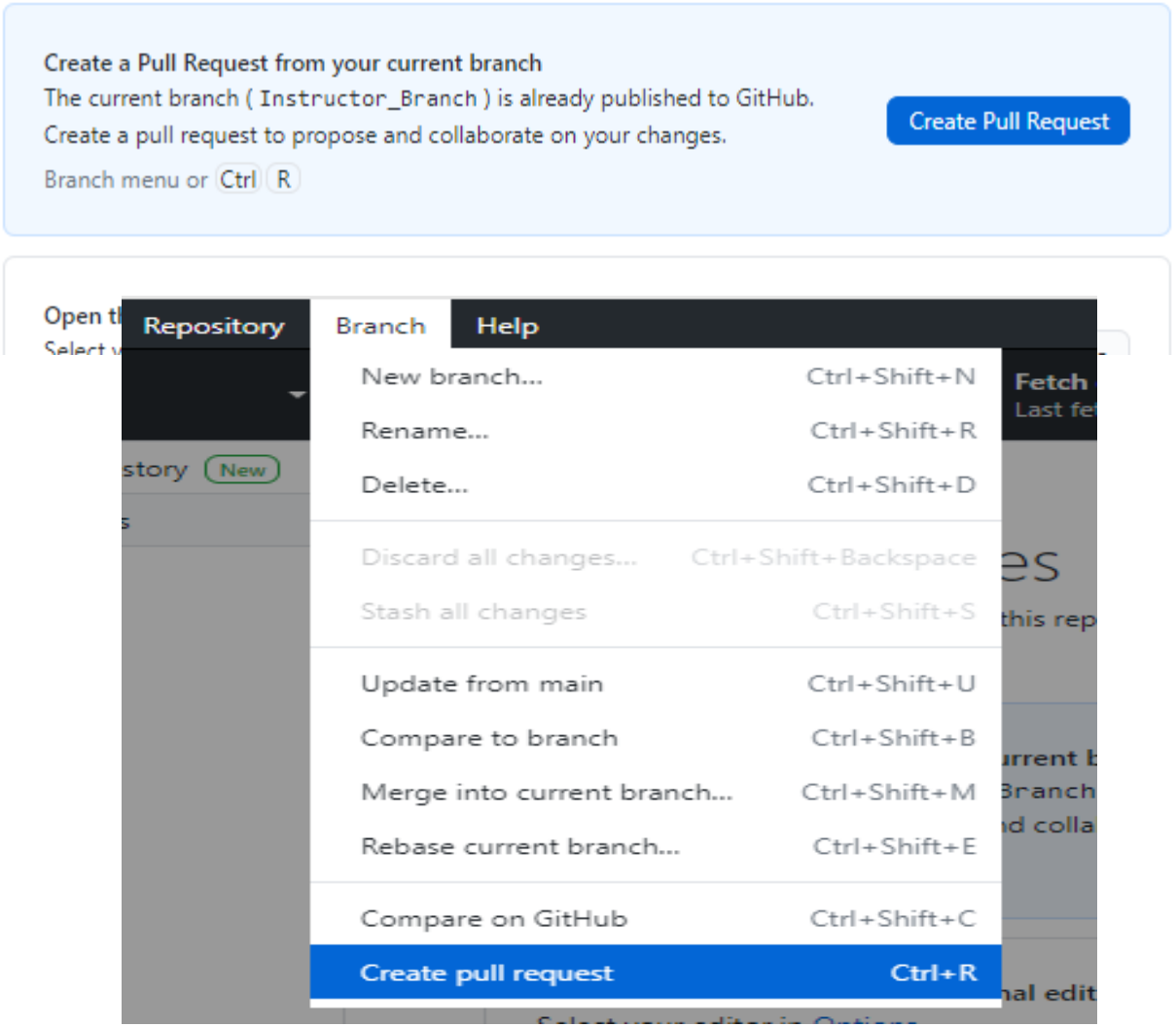
- In completing programming assignments in this course, you would be required to submit the URL of your forked copy of the repo to the submission portal on Moodle.



COLABORATION WORKFLOW

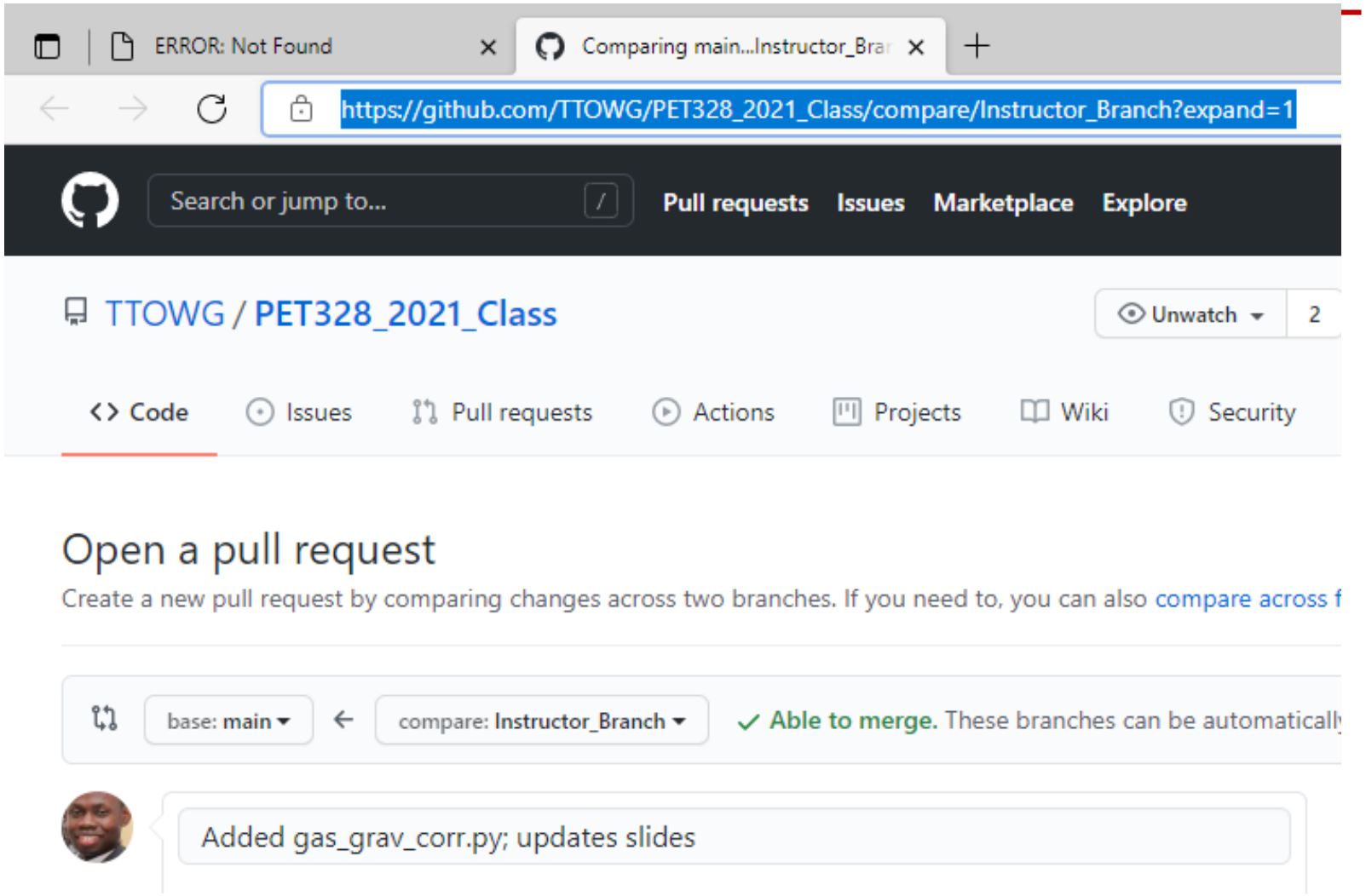
Creating a pull request

- As soon as you push a commit to your account, the GitHub Desktop app displays a small dialog box offering you a create a pull request to the original GitHub repo that your forked.
- Alternatively, you can choose the *Create pull request* command from the *Branch* menu.



COLABORATION WORKFLOW

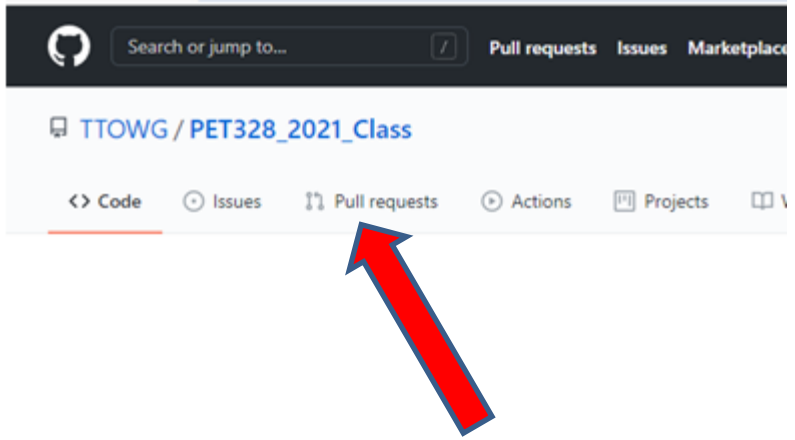
Creating a pull request



COLABORATION WORKFLOW

Accepting pull requests

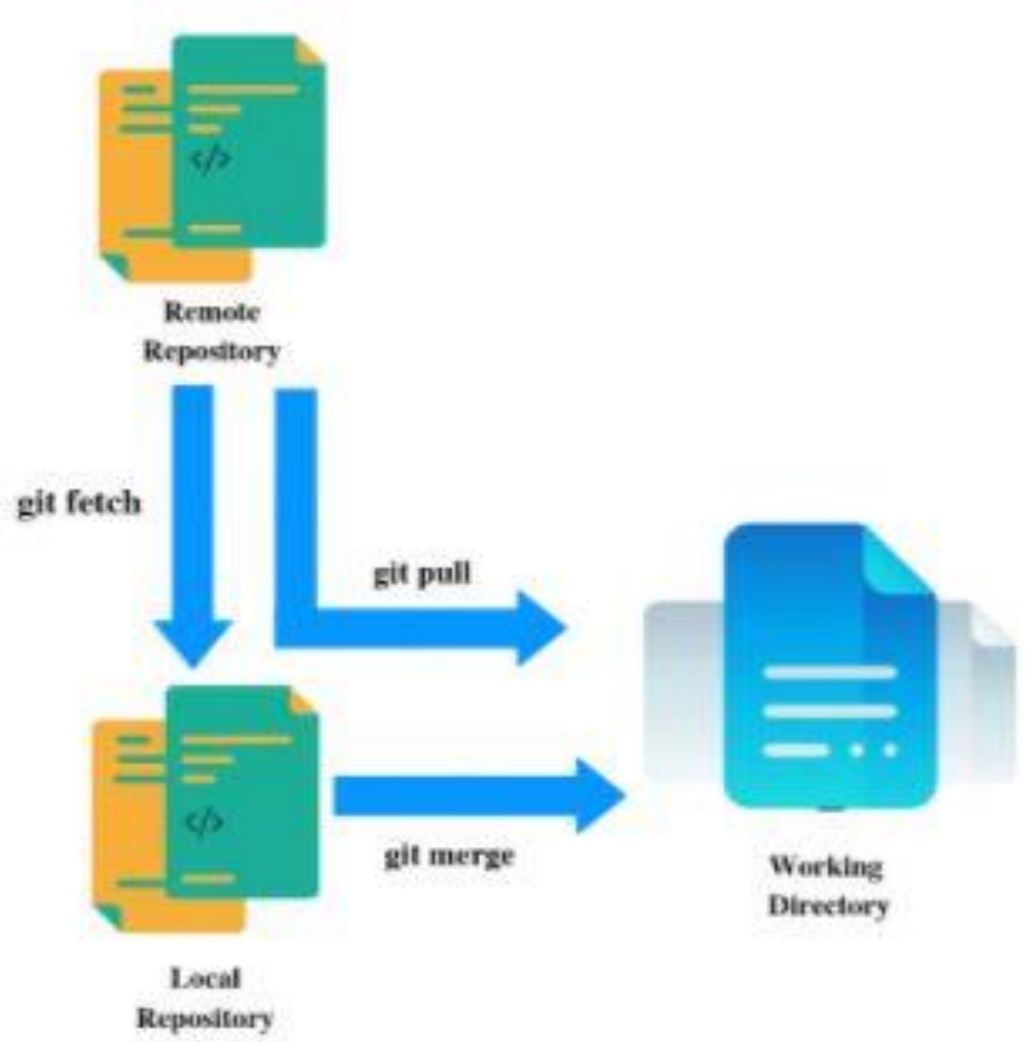
- Now, if someone (say the author of the original repo) made changes to their copy of the same repo; how may you update your own GitHub copy of the repo?
- When pull requests from your collaborators are available, they can be accessed via the Pull request button on your GitHub account.



COLABORATION WORKFLOW

Fetching changes

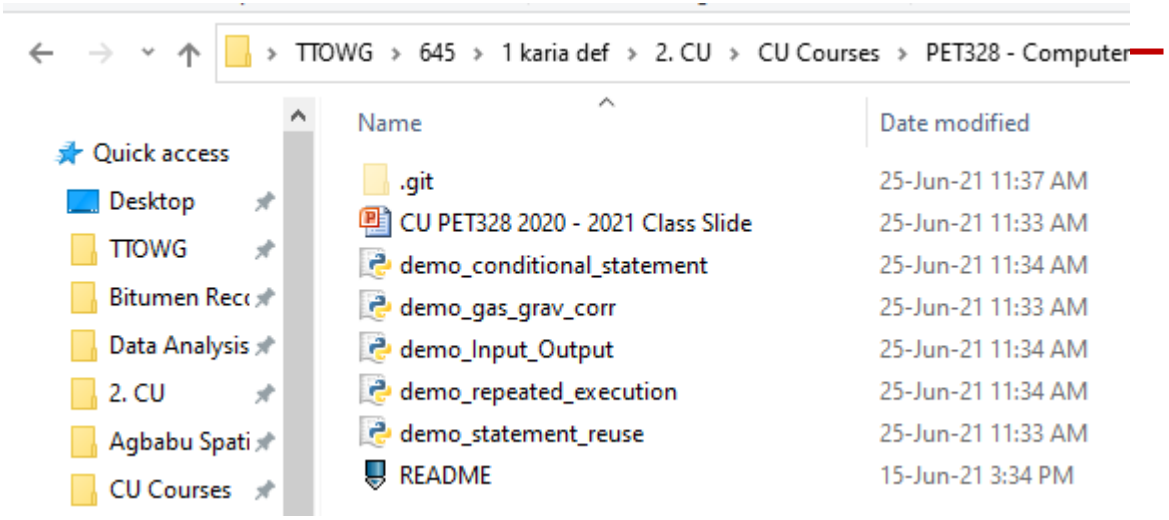
- The acceptance of a pull request only updates the copy of repo on your GitHub account.
- You need to bring in the changes to your Git copy and then to the copy cloned on your PC.
- To bring in changes to your Git copy, you simply fetch the changes. The process called Fetch is like the opposite of Push.

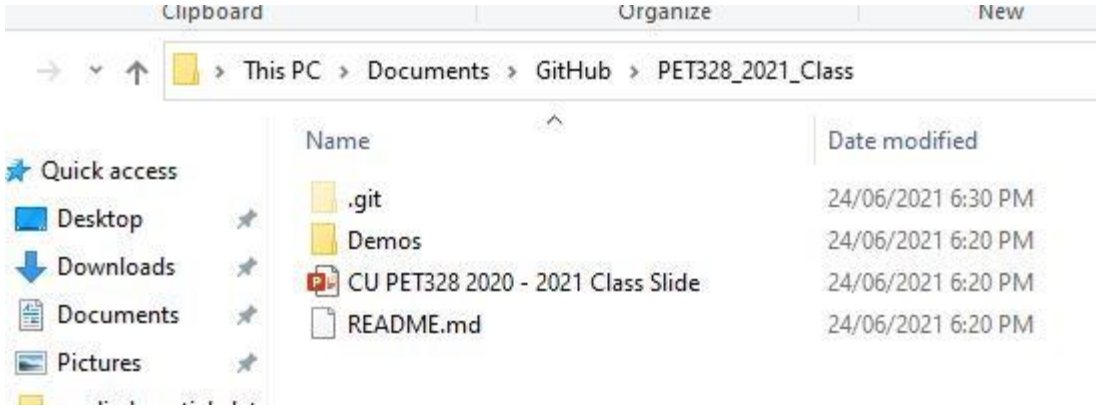


COLABORATION WORKFLOW

Fetching changes

- These are snapshots of the *PET328_2021_Class* repository cloned on different PCs.
- Changes were made locally, committed, and pushed from the PC depicted in the top image.
- A pull request was made and accepted.
- However, the repo on the second PC (bottom image) is yet to be updated of the changes.
- The Git copy on the second PC can only be updated with the Fetch process.

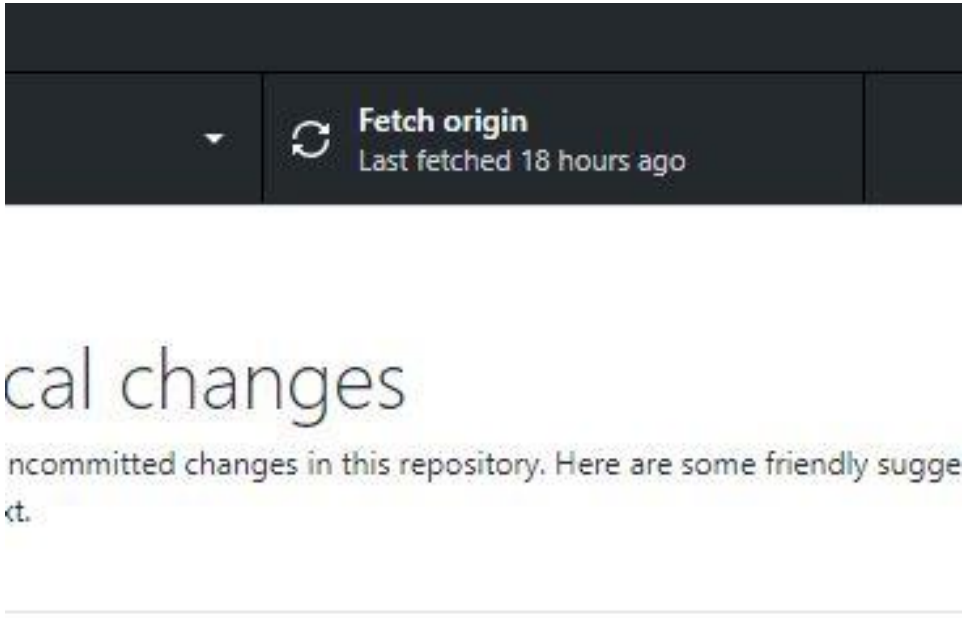




COLABORATION WORKFLOW

Fetching changes

- To fetch changes already accepted on your GitHub account to the Git's copy on your machine, simply click the Fetch button in the GitHub desktop account.



COLABORATION WORKFLOW

Merging changes

- Once changes are fetched to the Git's copy on your PC, they could be incorporated finally into the cloned repo with a process known as *merge*.
- To merge changes, choose *Merge into current branch* in the Branch menu of the GitHub Desktop app.
- Note that the cloned copy on the second PC is now duly updated.
- This accept-fetch-merge workflow is the means by which you keep your forked-cloned repo updated of changes from collaborators.