Basics

jueves, 6 de octubre de 2022

4:28 p. m.



Terms & Concepts

Links: [Git and GitHub for Beginners - Crash Course](https://www.youtube.com/watch?v=RGOj5yH7evk)

Additional resources:

* [Git & GitHub Crash Course For Beginners](https://www.youtube.com/watch?v=SWYqp7iY_Tc)
* <https://learngitbranching.js.org/?locale=es_ES>

* Directory: Folder.
* Terminal or Command Line: Interface for Text Commands.
* CLI: Command Line Interface.
* cd: Change Directory / To move around the terminal.
* Code Editor: Word processor for Writing Code
* Repository: Project, or the folder/place where your project is kept
* Git / GitHub: Git is the tool that tracks the changes / Github is the website that serves as fronting for projects

Git Commands

* clone: Bring a repository that is host somewhere like Github into a folder on your local machine.
* add: Track your file and changes in Git.
* commit: Save your file and changes in Git.
* push: Upload Git commits to a remote repo, like Github.
* pull: Download changes from remote repo to your local machine, the opposite of push.
* branch: tells you in which branch you're located in.
* init: creates a Git repository.

Notes

* Whenever commiting files, the file must have a message, with the command git commit -m "*Message of the commit explaining the 'What' and the 'Why'* ".

* If I wanted to create a Git Repository, being inside of the directory the command 'init' with the preceding 'git' word is to be executed and a hidden .git folder will be created and the machine will understand that that is a repository. In the prompt command git init

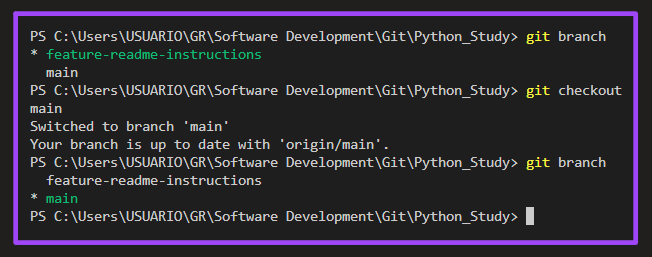
* If I wanted to create a new branch, being inside of the directory the command 'checkout' with the preceding 'git' and succeded by a "-b", plus "the name of the brach".

The practical example will be: git checkout -b 'feature-readme-instructions'

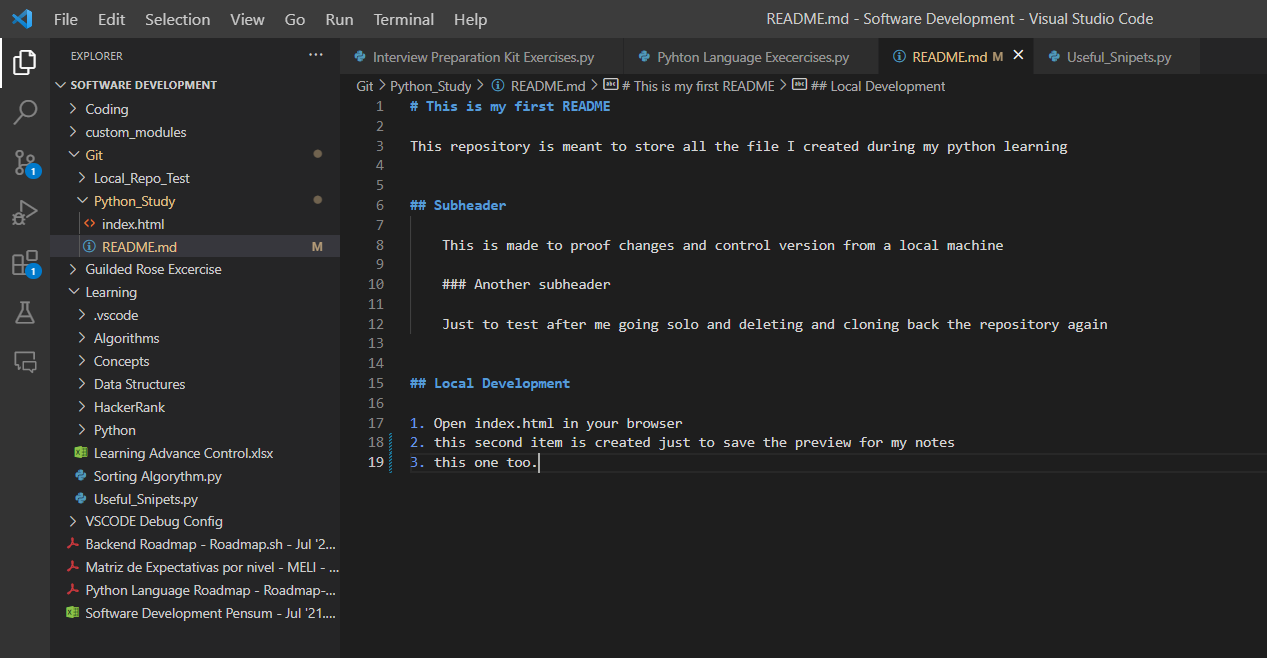
Texto

Descripción generada automáticamente

* Now, If I wanted to change from branch to branch, the command 'git' + 'checkout' + "the name of the branch" will switch branches.



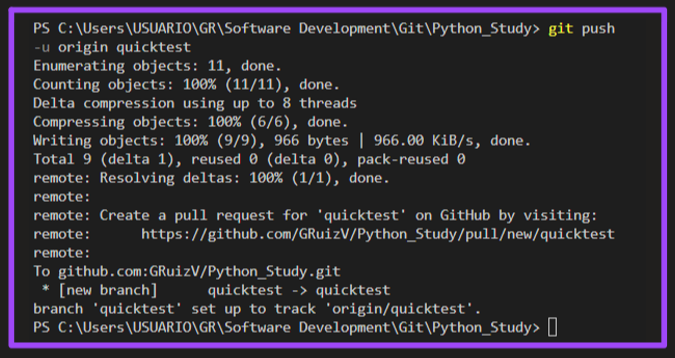
* While working on a branch, to see the difference between the file on the branch and vs the main, the git command 'git' + 'diff' will show up a preview of how the changes look.



Texto

Descripción generada automáticamente

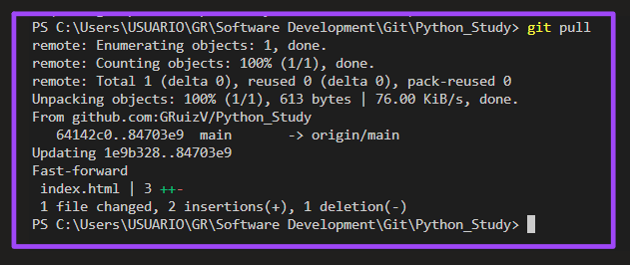
* When pushing a new branch, since it is not know to which branch the changes will be pushed to the remote repo, for setting this upstream the command is git push -u origin 'the name of the branch'.



* Pull requests: for a Pull request, which is basically saying 'I'm ready to merge with the main version', is the way to close the work after pushing the commits and changes on a branch. It is done just after executing the git push command and the pull request is expected to trigger the discussion with the team before closing. It can be done locally but remotely on Github, the tutorial from FreeCodeCamp showed how.

After the merger, commonly the branch is deleted.

After accepting the merge on GitHub, to get the latest version of the code in the local main branch, the command git pull will do the rest and update locally.



Undoing in Git

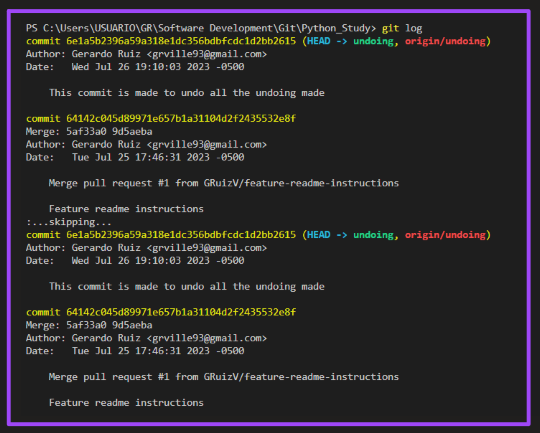
Say we have made a couple of changes in a branch and we added the file(s) and commit is what's left. With the command git reset or git reset file-name will take the file(s) of the staging area.

Now, lets say that a commit was done and it has an error in it and is required to undo that last commit, here instead of using the "file name" the word HEAD in Uppercase is used, then Git will understand HEAD as the last commit made, and with the tilde operator (~) plus a number, is the same to say 'undo back to the number stated of commits'.

Since there is no direct way to know where the HEAD pointer is directed at, the git log command with display a log, this to know how many commits back is needed to undo. The importance of having good and clear commit messages is shown here, since all the information available is the time of the commit and the message, usually when needed to go back to a certain point, the commit messages are the markers to do it.

Another thing to mention is that the commit 'hash' or the id (e.g.: 6e1a5b2396a59a318e1dc356bdbfcdc1d2bb2615) could also be used to go back to that point particularly with the command git reset 6e1a5b2396a59a318e1dc356bdbfcdc1d2bb2615 And all the files will be unstaged, uncommited and the HEAD pointer is now directing to that particular commit.

Finally, the additional command git reset --hard 6e1a5b2396a59a318e1dc356bdbfcdc1d2bb2615 will actually erase all the changes up to that point.



Git Forking

Forking is basically cloning a repository to have it under my user.