Git and GitHub useful commands.

->To configure your user email & username.

$git config - - global user.email " your email"

$git config - - global user.name "your username"

->Make a new directory.

$mkdir <name>

->Change the working directory.

$cd <name>

->Make a git repository in directory.

$git init

->Check if the directory exists.

$ls -la

->Look inside directory.

$git -l .git

->Add file to directory staging area.

$git add <file name> **OR** $git add *

->Get info about current working tree and pending changes.

$git status

->Commit your changes.

$git commit (Then write the commit msg in the editor and save)

OR $git commit -m "commit msg"

->Check current configuration of a directory.

$git config -l

->When adding a new python file in directory we need to make it executable first by:

$chmod +x <file name>

->To get the history of your commit msg.

$git log

->A shortcut to stage any changes to tracked files and commit them in one step (Doesn't work on new files).

$git commit -a

OR $git commit -a -m "commit msg"

➔ Git shows the head alias to represent the current checkout snapshot of your project.

->To look at the actual lines that changed in each commit.

$git log -p

->If you want to see a specific commit details by commit ID.

$git show <commit ID>

->To show stats about the changes in the commit "How many lines changed?".

$git log - - stat

->To keep track of everything you change before staged them.

$git diff

->To show the changes being added and ask you if you want to stage them.

$git add -p

->To see the changes that are staged but not committed

$git diff - - staged

->To delete files from a directory.

$git rm <file name>

➔ After we delete the file, it goes to the stage area and is ready to be committed.

-> To rename a file in a directory.

$git mv <old name> <new name>

➔ After we rename a file, it goes to staging area and is ready to be committed.

->To ignore file that you don't want. (First create a .gitignore file)

$echo file name > .gitignore

$git add .gitignore

$git commit -m "commit msg"

->To discard changes in the working tree. "Before staging them"

$git checkout <file name>

OR $git checkout -p <file name> "To checkout individual changes instead of the whole file"

->To unstage our changes that don't want to commit.

$git reset HEAD <file name>

->To overwrite the previous commit or add the other file in the staging area with the previous commit.

$git commit - - amend

➔ Don't use this command in public repository because it rewrites the git history, removing the previous commit and replacing it with the new one.

->To create a new file.

$touch <filename>

->To roll back a commit you made.

$git revert HEAD "The previous commit"

OR $git revert commit id "The commit you want to roll back"

->To show a list of branches we have in our repository.

$git branch

->To create a new branch.

$git branch <Branch name>

->To switch to a new branch.

$git checkout <Branch name>

->To create a new branch and switch to it in a single line.

$git checkout -b <Branch name>

->To delete a branch.

$git branch -d <Branch name>

->To merge branches together. "The master/main with other one"

$git merge <Branch name>

->To better understand the history of your merge occurred.

$git log - - graph - - oneline - - all "If you need all branches"

->To stop the merge process.

$git merge - - abort

➔ It will stop the merge process and reset the files in your working tree back to the previous commit before the merge.

GitHub

->After making a repository on github and want to work with it on your local computer, you can clone the repo on your computer.

1. Copy the URL of the repo. "From clone or download bottom"
2. Open git, then type.

   $git clone <URL>
3. Enter your username and password.
4. After making the changes to your repo on your computer, commit them.

   ->Then push them to github.

   $git push
5. Enter your username and password.

->To retrieve new changes from the remote repository "GitHub".

$git pull

->To avoid entering your github username and password.

$git config - - global credential.helper cache

➔ This will cache your credential for 15 min.

->To know the configuration of the remote repository.

- Go to the directory of the repository and run

  $git remote -v

➔ Contains URLS associated with remote repo “Fetch & pull” data

->To get more information about our remote repo.

$git remote show origin

->To look at the remote branches that our git repo currently tracking.

$git branch -r

->To modify the content of the repository.

1. We must pull any new changes to our local branch.
2. Merge them with our changes.
3. Push our changes to the repository.

->To see what others committed in the remote repo and to our remote branches.

$git fetch

➔ The changes are not automatically mirrored to our local branches.
it only reviews changes that happened in the remote repo; if you are happy with the changes then merge them into your local branches

->To see current commits in the remote repository.

$git log origin/master **or** main

->To merge the changes made by others that we got by fetch from origin/master branch to our local master branch.

$git merge origin/master **or** main

->To fetch the remote copies and current branches and automatically try to merge them into the current branch.

$git pull

->To look at the changes after merging by "$git pull".

$git log -p -1

->IF you run "$git remote show origin" and found that our colleague is working on a new branch, to create a local branch for it.

$git checkout <new branch name>

➔ Git automatically copy the content of the remote branch into the local branch.

->IF you want to get the content of the remote branch without automatically merge any content of our local branch.

$git remote update

->The first time you push a branch to a remote repository; we need to add a few parameters to git push command.

$git push -u <Branch name>

➔ It's another way to refer to remote repo and we also have to say we are pushing this to the origin repo.

->IF you want to merge a new branch to the main branch; if someone update the main branch and need a (three-way-merge).

$git rebase master **OR** main (run on the new branch)

➔ This will make the two branches have the same base, and now we can merge them by (fast-forwarding-merge).

->After you finish merging the two branches, you need to delete the other branch from remote and local repo.

REMOTE: $git push - - delete origin <Branch name>

LOCAL: $git branch -d <Branch name>

->When we try to rebase, and someone makes changes, and it comes up with a conflict.

➔ Solve the conflict, then add the changes you solved to the stage area, then run

$git rebase - - continue

Tips for best practices for collaborating with others.

1) **Always synchronize your branches before starting any work on your own.**
2) **Avoid having very large changes that modify a lot of different things.**
3) **When working on a big change, it makes sense to have a separate feature branch.**
4) **To make the final merge of the feature branch easier it makes sense to regularly merge changes made on the master branch back onto the feature branch to avoid conflicts.**
5) **Have the latest version of the project in the master branch and the stable version in a separate branch.**
6) **You shouldn't rebase changes that have been pushed to the remote repository.**
7) **Good commit messages.**