GIT/Github

A version control system, or VCS

Basic Git commands

1. `git init`: initializes a brand new Git repository and begins tracking an existing directory. It adds a hidden subfolder within the existing directory that houses the internal data structure required for version control.

2. `git clone`: creates a local copy of a project that already exists remotely. The clone includes all the project's files, history, and branches.

3. **git add**: stages a change. Git tracks changes to a developer's codebase, but it's necessary to stage and take a snapshot of the changes to include them in the project's history. This command performs staging, the first part of that two-step process. Any changes that are staged will become a part of the next snapshot and a part of the project's history. Staging and committing separately gives developers complete control over the history of their project without changing how they code and work.

4. **git commit**: saves the snapshot to the project history and completes the change-tracking process. In short, a commit functions like taking a photo. Anything that's been staged with git add will become a part of the snapshot with git commit.

git commit -m "message ki kyu change kiya tha"

5. `git branch`: shows the current branches being worked on locally.

6. `git pull`: updates the local line of development with updates from its remote counterpart. Developers use this command if a teammate has made commits to a

branch on a remote, and they would like to reflect those changes in their local environment.

7. **git push**: updates the remote repository with any commits made locally to a branch.

8. `git merge` `:` merges lines of development together. This command is typically used to combine changes made on two distinct branches. For example, a developer would merge when they want to combine changes from a feature branch into the main branch for deployment.

9. **`git status`**: shows the status of changes as untracked, modified, or staged.

10. `git checkout branch_name` :to move to another branch, `git checkout -b new_branch` if the branch doesn't already exist or `git branch branch_name`

koi branch delete krne k liye 1st go the another branch using `git checkout branch_name` and then use `git branch -d branch_name`

11. `git log branch_name`: to show all commit changes till now

12. `git diff <filename>` - Show changes between commits, commit and working tree, etc or a particular file

before doing git restore we have to do `git -–staged git restore`

13. `git restore <filename>` - Restore working tree files, jo bhi change kiya sb restored

14. `git show <commit id of the change>` to view what changed in that commit

15. `git reset soft <change id>` to delete the commit but leave the code change untouched

16. `git commit –ammend` amendment in the same commit rather than new commit after changing something again

creating conda environment create `conda env create --file` `environment.yml`

`conda env export --file environment.yml` creating environment.yml from existing conda environment

activating conda environment conda `cond activate` `nlp-reer-env`