- git
- dir / ls.
- mkdir projeit
- cd projeit.
- git init (initialized empty git repository where version, history is stored and is stored in .git folder)
- dir -a (will display all hidden files).
- dir .git (will display all files in .git folder)
- notepad names.txt (will create new file named as names)
- git status (To see what ~~make which~~ changes made). (green / red)
- untracked files (red color).
- git add names.txt (Now it will maintain its history).
- git commit -m " names.txt file added".
- type names.txt → will display content of names (Pranav Kardo)
- git status (as we modified names it will show in red color).
- git add names.txt
- git status (in green color)
- git restore ~~staged~~ names.txt (will ~~untaged it (red color)~~ ~~brow~~ delete change made is name.txt).
- git log (to see the commit made with history)
- git rm names.txt (will delete name file)

& now will do git add & as commit we and after git log which will show all the commits. If u we want just 1st commit then we have to do.

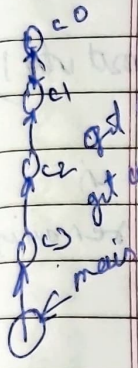
- git reset Index Id (mention the index id whose commit we to be key eg 3 2 1 if we want only 1 then add Prev index id so 2, 3 commits will not shows)
- ~~get~~ notepad heroes.txt.
- git add.
- git stash (will send names.txt & heroes.txt to backlog and will show nothing to commit, working tree clean)
- git stash pop (brings them again to unstage area)
- git stash clear (deletes them)

origin - (All repo + folder in your account is origin)
any folder that starts with our name is origin.

Date
Page

URL ~~origin~~, adding url → name of URL going to be ~~add~~

- git remote add origin URL. (will ..
- git remote -v (will display URL attach to the folder / fetch / push)
- git push origin master.
- ~~one~~ commit it looks like branch, by default its name is master, main
- use : when we work on new feature create new branch.



new main ~~per~~ previously known as master ..

origin is a shortend name for the remote repository ~~for~~ that a project was originally cloned from ..

- never commit on main branch. (create new branch and commit on it : as no if error occur it must not affect other user)
-
- git branch feature → create new branch name feature
- git checkout feature → ~~switches~~ Head is pointing to feature.
(Head is just a pointer that says all the new commits that you will be made will be added on the head, currently the head is pointing feature, hence all commits will be added on feature brand

- git merge feature (merge with main branch).
- git push origin master

- ~~fork~~ creates a copy of repo in which you can ~~add~~ any do anything you want (fork is done on github) and clone it done on git.
- git clone URL.
from where - you forked is upstream ~~and~~ url.
- git remote . add upstream url link .

If you are fork a repo and you make changes into it and want to show in main original repo then you will send a pull request to make it visible in original repo and then he will review the code or add any suggestion to it (Reviewers)

- Create multiple pull request for diff branch. (one branch one pull request)
- git fetch --all --prune --(will fetch all the changes into main)
- git reset --hard upstream/main
- git push origin main.
- git pull upstream main.
- git rebase -i Index (Squash :- Use to merge multiple commit into 1)
  ↳ interactive environment.

  pick 1 squash commit is it replace pick with s (Keep pick to only in which where have to merge the commit) & replace remaining pick with s).

- Merge conflict - Same change is done in file by two users.