1. git --version
2. git config user.name -------- to check if you have any username
3. git config --global user.name “Eklavya Bishnoi” ---------- this is to define the username in global to show your name if you made any edits.
4. git config user.name
5. git config --global user.email [eklavya.bishnoi@gmail.com](mailto:eklavya.bishnoi@gmail.com) ----- use this to assign the email
6. Terminal Comands
   1. LS – list the contents of the folder in the current directory
      1. ls folder\_name --- it will show what is in this folder
      2. ls -a --- it lists all file such as hidden files too
   2. Open . – if I type “ open . “ then it will open the current folder in finder
      1. Open folder\_name – it open the folder in the finder
   3. Cd – change directory is used to enter into a folder
      1. Cd .. ---- cd with two dots makes us move out of the folder by one level
   4. Mkdir – command to make a directory
      1. Mkdir plants
   5. Pwd ---- this command prints the current working directory
   6. Clear --- clears the terminal screen
   7. Touch ---- touch filename and it will create a file with that name in the folder where we are in the terminal
      1. You can create multiple files with different extensions you just have to use touch and then filename\_1, filename\_2, filename\_3
         1. Touch filename\_1.ipynb filename\_2.py filename\_3.txt filename4.pdf
      2. Use Touch folder\_name/file\_name.extension to create that file in a folder
   8. Rm ---- to remove any file use the rm command – this deletes the file permanently
      1. Rm file\_name
      2. Rm -rf -- deletes a folder -🡪 rm -rf folder\_name
7. Git Commands
   1. Git status --- it gives information on the current status of a git repository and its contents
   2. Git init --- used to create a new git repository. Before we can do anything git-related we must initialize a repo first. This is something you do once per project initialize the repo in the top-level folder containing your project
   3. Always use Gitstatus first to confirm that you are not in a respository. If you use Git init in an already existing repository then you would confuse git down the line. There should be only one git init per repository/repo.
8. Committing
   1. It is the most important Git feature
   2. Working Directory – it is the directory that I am working in.
   3. Staging Area --- you stage the changes that you want to make using the git add command and then once you have staged them you will make the changes to the repository using the git commit.
      1. Adding ---- “ git add filename1 filename2” ---- separate files with spaces to add multiple at once.
      2. Committing – use “git commit -m “Message here in quotes” “ – always use git commit -m “My message” ----- while you are committing new changes.

Stage using “git add” and then put it in the repository using “git commit”

* 1. Repository --- actual git repository the .git folder where you are making the changes or commits
  2. Git Commit -m “my message’
     1. We use the “git commit” command to actually commit changes from the staging area
     2. When making a commit, we need to provide a commit message that summarized the changes and work snapshotted in the commit
  3. Git Log : It retrieves the information for the logs of the commits for a repository
  4. Git add . ---- it stages all the files with the changes for the commit. If there are multiple changes to the files tracked by git then it will add all the files for the commit.

1. Code . --- this command opens up the current folder in VS Code
2. The Git Docs--- talks about git documentation
   1. Keep your commits atomic
      1. When possible , a commit should encompass a single geatuere, change or fix. In other words, try to keep each commit focused on. Asingle thing. This makes it much easier to undo or rollback changes later on. It also makes your code or project easier to review.
   2. Use Present-Tense Imperative Style ---- Describe your changes in imperative mood, e.g. “make xyzzy do frotz” instead of [This patch] makes xyzzy do frotzz or “I changed xxyzzy to do frotz”, as if you are giving order to the codebase to change its behavior
3. Closer look at the Git Log Command
   1. Git log --abbrev-commit
   2. Git log – -oneline
4. GitKraken – GUI to commit using a graphic user interface.
5. AMMEND COMMITS
   1. Suppose you just made a commit and then realized you forgot to include a file! Or, maybe you made a typo in the commit message that you want to correct.
      1. Rather than making a brand new separate commit you can “redo” the previous commit using the –amend option
         1. Git commit -m “some commit”
         2. Git add forgotten\_file
         3. Git commit --amend
      2. NOTE – The Git Commit amend only amends the latest commit not the previous commits
6. IGNORING FILES
   1. We can tell git which files and directories to ignore in a given repostiroy , using a .gitignore file. This is useful for files you NEVER want to commit including:
      1. Secrtes, API Keys, Credentials, etc
      2. Operationg System Files (.DS\_Store on Mac)
      3. Log Files
      4. Dependencies and Packages
   2. .gitignore – Create a file called .gitignore in the root of a repository. Inside the file, we can write patterns to tell Git which files and folders to ignore:
      1. .DS\_Store will ignore files name .DS\_Store
      2. folderName / will ignore an entire directory
      3. \*.log will ignore any files with the .log extension
7. Gitignore.io file that needs to be added
   1. .gitignore should be added to the base folder
   2. Use .gitignore.io to copy paste general files that needs to be out of io section.