1) Linux Commands
    a) git rm <fileNameWithExtension>
        i) deletes file
        ii) git stages the condition automatically
    b) git mv hello.txt hello_renamed.txt
        i) renames and automatically stages
    c) touch error.log
        i) creates a txt file named error.log

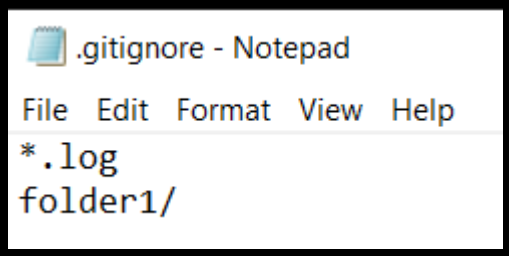
2) Installation, SetUp
    a) git config --global user.name “Himanshu”
        i) setting up user name
    b) git config --global user.email “agarkarhimanshu456@gmail.com”
    c) git config --list
        i) check username, mail
    d) git config --global core.editor vim
        i) choose editor, eg: vim > emacs
    e) git config user.name
    f) git config user.email

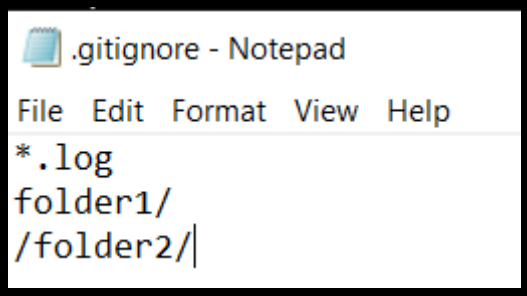
3) Tracking Files
    a) git status
        i) check if it is git repo
    b) git init
        i) make git repo
        ii) also creates a .git folder which is hidden
    c) git add --a
        i) add all files to staging area
        ii) after this when we commit there will be a snapshot of all files
        iii) make untracked files to unmodified
    d) git add first.txt
        i) add only ‘first.txt’ file to staging area
    e) git commit -m “Initial Commit”
        i) commit all files
    f) git log
        i) know which commit we did
        ii) type ‘Q’ to escape from continuously showing logs
    g) git log -p
        i) shows what change (git diff) is there along with commit logs
    h) git log -p -3
        i) Shows only 3 commits along with diff
    i) git log --stat
        i) git diff + no of lines added/removed
    j) git log --pretty=oneline
        i) shows all commits, commits take only 1 line
    k) git log --pretty=short
    l) git log --pretty=full
        i) shows author and committer name unlike short
    m) git log --since=2.days
        i) can replace days with weeks, months
    n) git log --pretty=formal:"%h -- %an”
        i) for customised output
        ii) https://git-scm.com/docs/git-log find for ‘placeholder’
    o) git commit --amend
        i) merge current stage area condition and last commit into 1 single commit

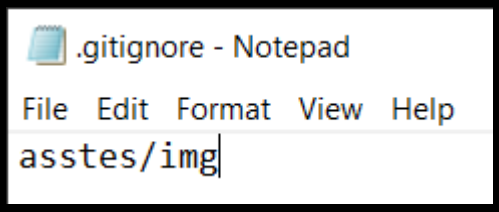
ii) press ‘i’ to start editing in editor
iii) press esc and type ‘:wq’ to save and quit the editor

p) rm -rf .git
   i) stop tracking folder
   ii) deletes .git file
q) git diff
   i) compares current status of files to that of files in staging area
r) git diff --staged
   i) shows difference between currently staged (and unmodified) files and last commit
s) git restore --staged <file>
   i) to unstage files after git add
t) git checkout -- hello_renamed.txt
   i) works when file are in working directory (unstaged)
   ii) rolls back the file to previous version of git add or commit whatever is latest
u) git checkout -f
   i) roll back to previous commit when dir is now unstaged
v) git commit -a -m "direct commit”
   i) committing files directly from tracked
   ii) does not commit untracked files (files which have not been added even 1ce in staging area)
w) git rm --cached <fileNameWithExt>
   i) stops tracking the file whose name is under .gitignore

4) Cloning Remote Repo
   a) git clone https://github.com/tensorflow/tensorflow.git
      i) contents are pulled from url to local folder
   b) git clone https://github.com/tensorflow/tensorflow.git <folder_name>
      i) customised folder_name

5) Other Points
   a) tells Git which files to ignore when committing your project to the GitHub repository
   b) add any file name in .gitignore file to ignore that file
   c) 

ignores tracking all log files, and folder1 anywhere present
   d) 

ignores only outer folder2
   e) 

ignore img folder in assets folder
   f) git ignores blank folder by default

6) Working with Branches
   a) git remote add origin <gitRemoteUrl>
   b) git remote -v

c) git branch
    i) check which branches are there
d) git checkout master
e) git checkout -b <branchName>
f) git push origin branch_b1
    i) push contents from main to b1
g) git push origin bb1:bb2
    i) copy branch bb1 is created named as bb2 in remote
h) git merge bb1
    i) merges bb1 to master
i) git branch -d bb1
    i) delete bb1 branch