\***cd git-repos** % creating a repository named git-repos

\***ls -la** % list all the files in the git repository

\* **git init** %initialize git repository

* **Rm -rf .git** %remove the directory we want to make change
* **Git status** % % shows untracked files that are not part of our preferences
* **Touch .gitignore** % will create a git ignore file to ignore file which is a simple text file which we add for git to ignore. For example let’s make a file that will be ignored by git.
* \* Go to the current directory and open the .gitignore text file and add the files to be ignored by git. For example, you can add the following file extensions. We can also use wildcards when referring to files.

\* In the text editor type the files you want git to ignore:

.txt

.project

.R

\*pyc

Then save your text editor.

* **Git add .gitignore**
* If you type “**git status**” now, you will no longer see the ‘.project’ file in staging area, so we will be able to commit the file without the ‘.project’ file
* **Git add -A** % for adding all the files in the files in the directory

Or also we can add files individually: ‘git add -filename

* Git reset calc.py % remove the calc.py file from the staging area
* **Git commit -m ‘Initial commit’** % shows the files to be commited
* **Git status** %to check our work of tracked and untracked files
* **Git log** % for getting information about the committed files

**To Track an Existing Remote Repo**

* **Git clone <url> <where to clone>**
* **Git clone ../remote\_rep.git .** % the ‘.’ means current directory

**Viewing Information about the remote repo**

* **Git remote -v** % It lists the information to the local repo
* **Git branch -a** % list all of the branches in our repo (local/remote branches)

**Pushing Changes**

* **Git diff** % will show me the changes that I have mades to the code, ‘+’ sign meaning the lines added and ‘-’ sign means the code deleted/removed.
* **Git add -A** % to add all files to the staging area
* **Git commit -m ‘multiply function’**
* **Git pull origin master** % it will pull all the changes since our last pull from the remote repo
* **Git push origin master** % push our changes to the remote repo

**Common Workflow**

**Create Branch for desired feature**

* **Git branch calc-divide** % calc-divide is the name of a new branch
* **Git branch** % will list the branches that currently exist
* **Git checkout calc-divide** % working in the calc-divide branch
* **Git branch** % will name the current branch you will be working
* **Git status**
* **Git add -A**
* **Git commit -m ’divide function’**

This has no effect on our local master branch or remote branch, and we have to push it remote repo for our changes

* **Git push -u origin calc-divide** % associate our current branch associate with our remote repo so next time we will only have to do git push/pull without -u
* **Git branch -a**

**Merge a branch**

* **Git checkout master** %checkout our local master
* **Git pull origin master** %
* **Git branch --merged** % the branches that we have merged so far
* **Git merge calc-divide** % merges the changes to the master branch (local)
* **Git push origin master** % changes pushed to the master branch to the remote

To check if the newly merged branch was successful

* **Git branch --merged**

**Deleting branch**

* **Git branch -d cacl-divide** % now it has been deleted localy
* **Git branch -a** % check changes on the branch
* **Git push origin -- delete calc-divide**