Git commands

Git – version gives the version of Git

Git – help gives the help menu commands

Git – help command name displays the format for any given git command

Eg: Git – help init

Git config – global user.name “meera-ramesh19” to assign a user name to git

Git config – global user.email “[meeraramesh19@outlook.com](mailto:meeraramesh19@outlook.com)” - to create a git account if you already have one give the email you registered with

Git config –global core.editor”Notepad”

If you create a file in notepad

Touch test.txt

To edit in notepad

Notepad test.txt takes you to the file and opens notepad

Git config –global core.editor”atom”

Atom test.txt

Favorite editor options to use when coding are

Atom or Notepad

Vim or nano

Git config – list : displays all the setup in the git

Make a dir and change to it

Mkdir shop , cd shop

To track this directory

Git init - we have initialized an empty file in this directory. This is the master branch

Ls -lists the files

Ls -A shows the list of all files in the directory

Cd .git to move into directory

Touch lists.txt

Rm lists.txt

Ls

To track a file we need to use git add”filename”

Git add “list.txt” – on branch master no commits but we can see the file in the directory. This step is called staging. Now we can commit this file

Git commit -m m stands for message

Eg : git commit -m “create a shopping list”

Git status shows the commit-

Git log- shows that we created a shopping list file with a unoque id number for the commit and tracks all the changes we did to the file.

To make changes to this list now

Notepad List.txt opens it

Add some columns and save it

Git status doesn’t save the changes

Use clear command to clear the screen

Git diff now shows the changes we added to the file

There is an a and b version on the git diff to the file list.txt

Git add list.txt- shows the list with changes

Git status shows the changes

Git commit -m”Add fruits and vegetables”

Git log shows all the info about the file

If we change in the notepad and save then goto git status it shows that we have changed the file.

Git diff lists out all the differences form the previous commit to the new changed file that we haven’t committed yet.

Now we can commit the following way

Git add list.txt to stage the file

Git commit -m”Restructure list to include new headings for different categories of purchases”

We need to explain the message of the file committed in great details

Git status

Git log shows all three commits that occurred now

Git log -1 shows the most recent commit

Add yogurt for the diary section in list.txt

Git add list.txt to stage the file

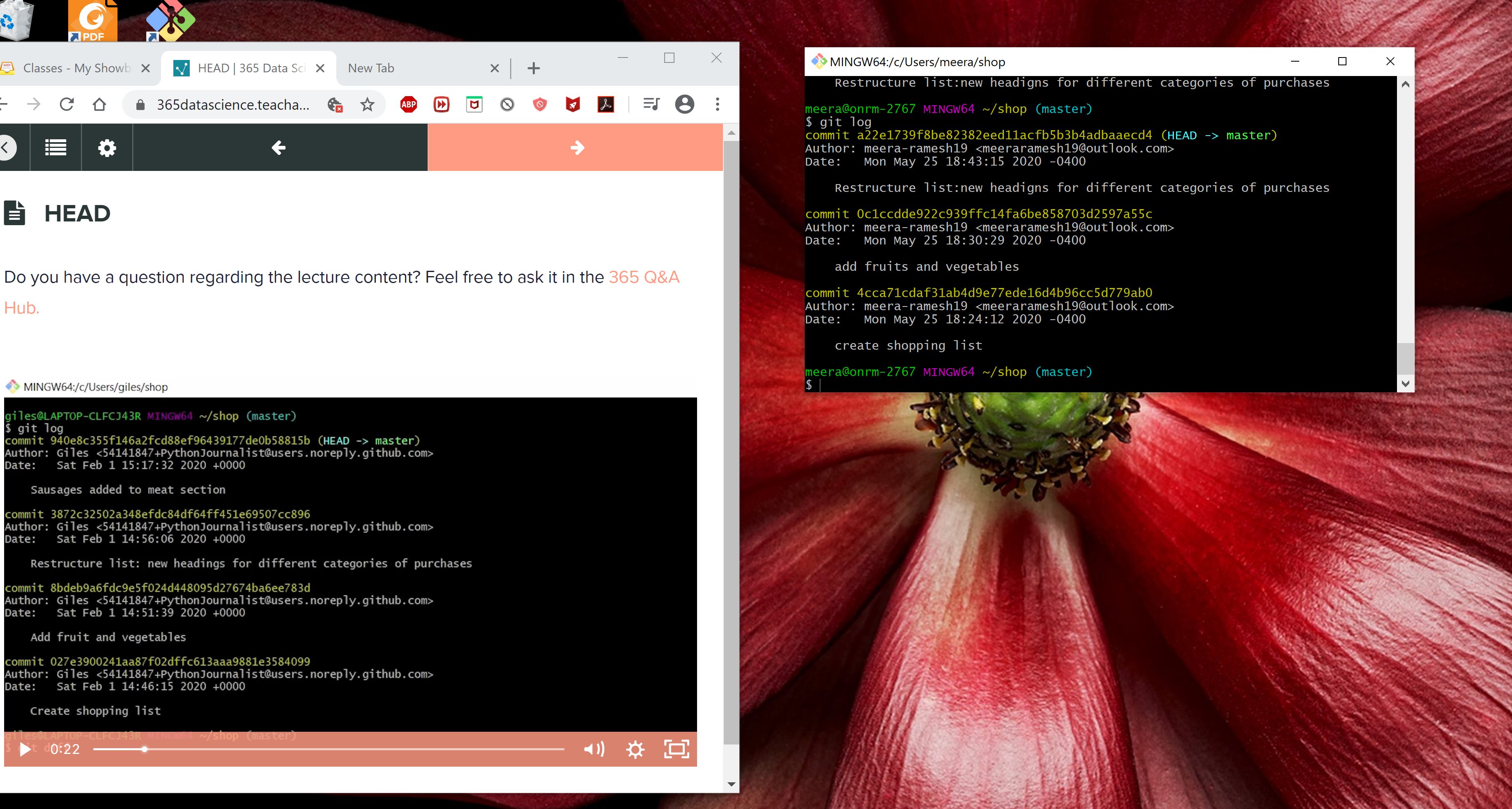
Get diff doesn’t show any changes

Git diff –staged will show the changes made in the list.txt (what we staged and what is in our working copy)

Git commit -m “Yogurt added to diary section” then commits to the file

Git log -shows the new commit

Git diff



When you type in the following number from fruits and vegetables

git diff 0c1ccdde9 list.txt

**diff --git a/list.txt b/list.txt**

**index af11c07..1f077c2 100644**

**--- a/list.txt**

**+++ b/list.txt**

@@ -1,4 +1,16 @@

+

+Fruit and vegetables

+

bananas

strawberry

grapes

-apple

\ No newline at end of file

+apple

+

+HOUSEHOLD

+

+

+

+DIARY

+

+

+PETS

:

HEAD command will enable us to go back to all the versions of the file

Git checkout HEAD-3 list.txt - will display the original list file without any commits

Git checkout HEAD-2 list.txt – file when it had first commit

Git checkout HEAD-1 list.txt – file when we did the second commit

Git checkout HEAD list.txt – file when we did the last commit

**Making corrections git reset**

**When you make a mistake in the file**

**Notepad List.txt made a mistake accidentally in this file**

**Git add list.txt -ready for staging**

**Git status**

**Git diff –staged now we have staged but to unstage it**

**Git reset list.txt – we have unstaged those changes**

**What instead of staging we committed the changes then,**

**Git commit -m “add content to Diary”**

**Git log**

**Notepad list.txt**

**One options to use is Reset with HEADoption**

**Git reset HEAD-1 – soft**

**(soft ,mixed, or hard options)**

**Soft options removes the commit that we did but it doesn’t change what happened in the staging area and the file hasn’t been changed.**

**Git reset HEAD-1 –mixed**

**Mixed removes the commit and also makes the changes unstaged but it still doesn’t change the file.**

**Git status now shows that it has not added changes to the commit.**

**Git reset HEAD-1 – hard**

**Most recent commit is not there and when we look at the file it has changed the file that we had added**

**Reset can also be used in a staged area to do unstage.**

**BRANCHING**

**We could make different branches for diary and fruits branch**

**Git branch dairy**

**Git branch fruits**

**Git branch – displays all the branches including the master**

**Git checkout dairy will switch you to the dairy branch**

**Goto notepad list.txt and make some changes in the diary section by adding butter, cheese, milk , eggs.**

**Git add list.txt**

**Git commit -m “update dairy”**

**Git log**

**Notepad list.txt does not have the updated diary items in the master branch.**

**Git checkout fruits**

**Notepad list.txt**

**Git add list.txt**

**Git commit -m “Further development of fruits section**

**Git checkout master still does not have the updated file in dairy and fruits branch but the other respective branches have been updated.**

**Notepad list.txt**

**Git checkout dairy**

**Notepad list .txt**

**Git checkout fruits**

**Notepad list.txt**

**If more than one person have been working on different parts of the file this will be really helpful to have different repositories where they can work alone on it.**

**To merge all of it , we need to go to master branch:**

**Git checkout master**

**Git merge fruits**

**Notepad list.txt now we can see the changes in master branch**

**Git merge dairy**

**Notepad list.txt**

**Create a New repository**

**Make it private**

**Copy the https url to the clipboard and now go back in to the command line. We want to synchronize our local repository with the new repository on Github.So we use Git remote to do that.**

**You can see the commands with**

**Git help remote**

**Git remote**

**git remote add origin** [**https://github.com/meera-ramesh19/365\_practice.git**](https://github.com/meera-ramesh19/365_practice.git)

**git remote -v – now is associated with the git account to this repository**

**We can now push our repository to the remote repository on Github by using push command**

**We want to push our local repository to the remote repository on Github**

**Git push origin master you will be asked your username and password**

**GITCLONE**

**Downloading a repository to your computer that is on github**

**Create a new dir**

**Mkdir imaging\_data**

**Cd imaging\_data/**

**Goto clone or download on that repository**

**On CMD type in**

**Git clone and the url of the repository you need to clone/download**

**These are the commands you use when you are collaborating with other people**

**Git fetch**

**Git pull**