Ls: current folder

Pwd: current work directory

Cd /c: change directory to c disk

cd ..: go to the previous work directory

When you want to access a folder, first use command, then tell where you want to go: cd data-shell, then you went to data-shell

Check url for more help <https://swcarpentry.github.io/shell-novice/02-filedir/index.html>

Then you can us mkdir: make directory, create a new folder

Typing “nano draft.txt” to create a new file. Ctrl+o to write out

ls –l: to check if this exist or not.

rm -i +filename.txt can remove the file

rm -R +foldername (remove a folder)

mv thesis/draft.txt thesis/quotes.txt (moves a file to a new file with a different name, which is how you rename a file. Can also change the path to move a file somewhere else.)

lls cu\* (will search for everything that starts with cu)

ls \*e\* (will search for everything that has an e anywhere in the file name)

ls \*.pdb (will match everything with the .pdb extension)

ls \*t?ne.\* (will match with octane.pdb and pentane.pdb)

wc cubane.pdb (WordCount counts the words in the file and gives more information about the file)

wc -l (print new lines) cubane.pbd (list only word count)

nano to open the file

cat to

wc -l \*.pdb

wc -l (get out by using ctrl+c or ctrl+d)

wc -l \*.pdb > lenghts.txt

cat lengths.txt (Concatenate FILE(s) to standard output.)

nano lengths.txt

sort lengths.txt (sorts everything in the lengths file)

sort lengths.txt > sorted\_lengths.txt

Question: What is the difference between 'sort' and 'sort -n'

sort lengths.txt (Sorts on the first character. E.g. 107 will come before 20 because 1 is before 2 in the alphanumerical alphabet)

sort -n lenghts.txt (Sorts on the first whole number. E.g. 20 will come before 107 because 20 is a smaller number than 107)

echo test

echo The Echo command prints things

echo The Echo command prints things > file\_echol.txt (it will create a new file)

cat file\_echol.txt

echo and is a great tool >> file\_echol.txt (It will append everything to the input)

Link to exercises (Pipes and Filters):

https://swcarpentry.github.io/shell-novice/04-pipefilter/index.html (what does >> mean)

head –n filename: print first n lines in file.

sort -n lengths.txt | head -n 2 (| is called the pipe symbol)

sort -n lengths.txt | head -n 2 > sorted\_head.txt

cat sorted\_head.txt

Link to exercises (Pipes and Filters):

https://swcarpentry.github.io/shell-novice/04-pipefilter/index.html (Piping Commands Together)

Link to exercises (Pipes and Filters):

https://swcarpentry.github.io/shell-novice/04-pipefilter/index.html (Piping Commands Together)

head -n 15 $1(which refers to the first file! The first input) | tail -n 5(write in text editor)

(when you forget to input the yellow line, the computer will take more time to search for the file, using ctrl+C to exit the operation.)

Question: What happens if you replace "\*.pdf" with $1 in the first line of the script?

Answer: "\*.pdf" gets replaced with the actual list of files matching \*.pdf, then $1 refers to the first element of this list

Link to exercises (Loops):

https://swcarpentry.github.io/shell-novice/05-loop/index.html

(Limiting Sets of Files)

(Saving to a File in a Loop - Part One)

grep the haiku.txt (find the word in this file) also there is many included commands like, –n –v –w, using help to find the function of these commands.

Question: Why is - treated as an option inside quotes?

Answer: The quotes are lost the moment the command is executed, so "-" is still treated as a flag.

in the terminal:

git

git config --list #tell you all the options to configure Git in your computer

git config --global user.name "Your Name" #Git requries you to have a username and an email

git config --global core.autocrlf true #set up Git in Windows. For Mac use "input" instead of "true" after core.autocrlf

git config --list #to check that the setting has changed

git init #initialize a repository

ls -a #to see hidden files in your directory. You will see Git creates a .git/ folder, which contains all git configuration.

cd .git #just to see what there is in there. Usually do won't have to do anything here

ls

cd .. #going back to planets directory

git status #current status of the git repository

mkdir moons #create a new folder within planets directory

cd moons

git init #git does not complain if you initialize a repo within a directory where a repo had already been initialized (planets)...let's see what is happening

touch Io.txt #creating a file called Io.txt...let's see if Git is going to be tracking the changes of this file in the repository created in the moons directory.

#You will probably find a problem at some point...Be aware of that..and do not create repos within repos

ls -a

rm -r .git #removing the .git directory within the moons directory

rm Io.txt #removing the text file we just created

cd ..

nano mars.txt #creating a file in the planets directory and editing it

git status #Git noticed you created a file but is not tracking the changes of it...for that you have to explicitly ask Git to track changes on this file. We do that by using 'add' :

git add mars.txt

git commit -m "Started notes on mars plans" #commit a comment explaining what changes you have made to the file

git status

git log #to see the log git has on what you have done to the file

nano mars.txt #edit the file by adding a new sentence

git status

git add mars.txt #this is to add the changes. Whenever you make changes to the file or create a file or whatever you want to save, then you have to do the 'add'

git diff

Parenthesis:

"git add --all" would add all the changes you have made at once. But be careful with what you save. Sometimes you want to save only some changes. That is why people add chanegs at once and not with '--all'. Let's go further into the difference between commit and add:

--'add' takes the changes to the staging area

--'commit' sort-of officially submits the changes to the repository

When git presents the changes, remember 'a' refers to the previous version and 'b' with the new version

###################################################################

> nano mars.txt #edit the file

> git status

> git commit -m "mention of gloves"

> git log

> git diff

> git status

> git add mars.txt

> git log

> git show 0329ajhfdj723487x37 # to see the specific changes of a commit. The string there is fake. You have to use the identifier number of the change you want to see instead. That identifier you can see when you commit changes. Every commit has it's own identifier.

> git log

> git status

> git commit -m "need wool"

> git log

> git log --online

> git log --oneline

> mkdir spaceships

> git status #It won't show the fact you created a folder...let's try to add it anyways to see what happens

> git add spaceships

> git status #It doesn't show anything because Git does not track folders

> nano spaceships/arianne.txt

> ls spaceships

> git status

> git add spaceships

> git status #Here Git mentions that there is a file in there

> touch spaceships/sputnik-1 spaceships/apollo-11

> ls spaceships

> git status #git tells you there are two files it is not tracking..so let's add them

> git add spaceships

> git status #now you can see Git is tracking them

> git commit -m "Add initial information avout spaceships"

###################################################################

Exercise time! see Exercise at the end of the lesson 04: http://swcarpentry.github.io/git-novice/04-changes/index.html

###################################################################

> git log

> git checkout 13249 mars.txt #the number be replace with the first few characters of the change identifier. You do not need to write the enitre number

> cat mars.txt

Here we go back to take a look at the history and then we come back to the most recent status:

> git log

> git checkout 13249 mars.txt #the number be replace with the first few characters of the change identifier. You do not need to write the enitre number

> cat mars.txt

> git checkout HEAD #HEAD is the identifier of the most recent commit

> cat mars.txt

> git status

> git chekout master

> git status

> cat mars.txt

> git status

> git log #let's see the identifier of the commit where we want to start from...going back in time again to a specific version of the file

> git checkout "identifier" mars.txt #change "identifier" by the real string of the commit of the version we want to go back to.

> cat mars.txt

> git status

> git add mars.txt

> git commit -m "removing poor changes"

> cat mars.txt

> git log

###################################################################

Exercise time! see Exercise at the end of the lesson 05: http://swcarpentry.github.io/git-novice/05-history/index.html

###################################################################

> git blame mars.txt #every line and dates of when it was last modified and the person who did that (the one to blame!)

> git status

> ls -a

> nano .gitignore #edit the .gitignore file and write \*.dat (all files ending in .dat)

\*.dat

> cat .gitignore

> git status

> git add .gitignore

> git commit -m "Added git ignore file for .dat files"

> nano data.dat

> git status #Git does not see the changes in data.dat because in .gitignore we specified "hey Git just ignore all files ending in .dat"

> ls

> git status --ignored #Git still sees the files that will ignore

> mkdir results

> cd results

> touch a.txt b.txt

> ls

> cd ..

> nano .gitignore # add 'results/', so that .gitignore file looks like this:

\*.dat

results/

> git status

> git add .gitignore

> git commit -m "ignoring resutls folder"

> git status

> nano results/a.txt #let's change something in this file

> git status #Git is ignoring all the changes in the files within the results folder

Connect to your github account! We are going to start pushing and pulling changes from your local repo to the Github repo in the web.

From your planets directory:

> git remote add origin https://github.com/NZR/SWC\_Planets.git #origin is the local name you will give to the link

> git status

> git push -u origin master #all the files commited in the local repo in planets will now be pushed to the web repo in Github

When collaborating with others and working on the same files --> clone or download a repo in Github

In your local computer:

> mkdir example

> cd example

> git clone https://github.com/NZR/SWV\_Planets.git

> ls -a

> cd SWC\_Planets

> ls

> git log #we can see now all the history of changes made to that repo

cd ..

ls

cd planets

ls

nano mars.txt #edit the file

git status

git add mars.txt

git commit -m "added notes on mars being too cold" #this is in your local repo (if you go the web repo, you won't see these last changes)

git push #we have to push whatever we have done in our local repo to the web repo in github. Then we will be able to see the lates changes in the web repo

When creating code and sharing code with others in public repositories like Github, be aware of copyright! If you are using part of someone else'

chooselicense.com

git pull (to download the file contain in the folder)

quite vim mode: (:q!)