*Lesson 1: The Unix Shell

Files you'll need for the lesson: http://swcarpentry.github.io/shell-novice/data/data-shell.zip
Download this link and unzip to your Desktop

*Lesson 2: Version Control with git

Download git from https://git-scm.com/download
Create a user account on github.com if you do not already have one.

```
# Configure some git settings
git config --global user.name "My Name"
git config --global user.email "myaddress@example.com"      # use whatever address you user when you signed up on github.com
git config --global core.editor "nano -w"

# Go to the desktop directory
cd ~/Desktop

# Create an empty folder (directory)
mkdir planets

# Go into the planets directory
cd planets

# Make the planets directory into a git repository
git init

ls
ls -a

git status

# Create a new text file
nano mars.txt
# Write some text
# Hit Ctrl-X to exit; you will be prompted to save the file;

# View the contents of mars.txt
cat mars.txt

git status

git add mars.txt

git status

git commit -m "started notes on Mars"

git status

# Look at the commit history of your git repository
git log
```

```
# Make a change to mars.txt (add a line)
nano mars.txt

git status

# See the difference between the new changes and the last time changes were committed
git diff

git commit -m "fact about Total Recall"
# There should be an error because we forgot a step

git add mars.txt
git commit -m "fact about Total Recall"

git status

git log

nano mars.txt
# Add some text

git diff

git add mars.txt

git diff
# There shouldn't be any differences

# Check the difference between the staged files and the last commit
git diff --staged

git commit -m "third line"

git status

git log

# Look at the most recent commit message
git log -1

# Look at condensed (one-line) commit messages
git log --oneline

git log --oneline -1

git log --oneline -2

git log --graph

mkdir spaceships
ls
git status
git add spaceships
git status
# There shouldn't be any changes her because git doesn't care about empty directories
cd spaceships
```

```
touch .gitkeep
ls -a
cd ..
git status
# Now git will care about the spaceships directory
# But we don't actually want to keep it
rm -r spaceships/

nano venus.txt
# Add some text

git status

nano mars.txt
# Add some text

git status

git diff
# It will only show the changes to mars.txt

git add mars.txt
git add venus.txt
git status

git diff --staged
# It should show the changes from both mars.txt and venus.txt

git commit -m "Create venus.txt and mention Mars being cold"

git log

cat mars.txt

Note: "HEAD" refers to the most recent commit.

git diff HEAD mars.txt
# Should be no output

nano mars.txt
# Add some text

git diff HEAD mars.txt
git diff
# The output from thest commands should look the same


git diff HEAD~1 mars.txt
# You can think of "~" here as being like a minus sign

git diff HEAD~2 mars.txt

# Show the changes that were made to mars.txt in the commit "HEAD~1"
git show HEAD~1 mars.txt

git log
```

Copy the long text on the line that begins with "commit"
git diff {paste the long text here} mars.txt

Shorter versions of the commit hash will work as long as there's enough so that it only matches one commit (6 or 7 characters is usually enough)
git diff 6bfce

Restore mars.txt from the most recent commit
git restore mars.txt

This will also do the same as the restore command
git checkout HEAD mars.txt

git checkout 6e22f1a
This sets the contents of your git repository to where it was at the specified commit and puts you into a "detached head" state
Get out of the detached head state
git checkout master

Note: The "git revert" command creates a new commit that undoes a specified previous commit.

nano venus.txt
Add some text

git add venus.txt

git status

git restore --staged venus.txt

git status

mkdir results
touch a.dat b.dat c.dat results/a.out results/b.out
ls

git status

nano .gitignore

- *.dat
- results/

git status

nano .gitignore

- *.dat
- results/
- !a.dat

git status

git add .gitignore

```
git add a.dat
git commit -m "create gitignore and add data file exception"
```

Log into github.com
Click + in upper right corner of screen to create new repository
Create a repository named "planets" and set it as public
Copy your repository address (should be something like https://github.com/{your_username}/planets.git)

```
git remote add origin {paste your repository address}

git remote

git remote -v

git push origin master
# You should get prompted to log into github
```

Now, in your browser refresh your repository on github. You should see all files and commit history from earlier.

```
cd ~/Desktop
mkdir home-planets
cd home-planets
git clone {paste your repository address}

nano pluto.txt
# Add some text

git status
git add pluto.txt
git commit -m "create pluto.txt"
git status
# You should see a message that your branch is ahead of origin/master
git push origin master

#
#
#

# Create a conflict
cd ~/Desktop/home-planets
nano mars.txt
# Make some change
git add mars.txt
git commit -m "home planets mars addition"
git push origin master
cd ~/Desktop/planets
nano mars.txt
# Make some change
git add mars.txt
git commit -m "planets version mars addition"
git push origin master
# There should be a message because the remote repository has work that you do not have locally
git pull origin master
# There should be a message about a conflict
```

nano mars.txt
There should be some lines between "<<<<<<<<" and ">>>>>>>>" that show the conflicting changes between the different versions. You'll have to resolve the conflict manually; it could be choosing one change or the other or some combination of the two.

git add mars.txt
git status
git commit -m "fix mars.txt conflict"
git push origin master

cd ~/Desktop/home-planets
git pull origin master

*Lesson 3: Python (Part 1)

Question: Is there a way to print/reference separate parts of a list within a single indexed notation? E.g., list[1,3] to pull the second & 4th value from list - rather than doing: list[1] list[3]
Answer: We don't think so.

Question regarding setting numbers to a specified length in Python (decimals):
print("0.4f" % (3))=3.0000
print("0.4f" % (3.14))=3.1400

- [Courtesy of Heather Orr - thank you!]
-

Question: What's a good resource to get a list of all possible libraries to import into Python?

Question: Why are some functions/commands structured as function_name(argument) while others are variable.function(argument)?

- [I believe this involves a better definition/explanation of "function" vs. "method"]

*Lesson 3: Python (Part 2)

*To install, go to:

- HTTPS://osu-carpentry.github.io/2019-11-01-okstate/

For today's lesson, please make sure that you download the data files from:

https://swcarpentry.github.io/python-novice-gapminder/files/python-novice-gapminder-data.zip

We will be going through the "Plotting and Programming in Python" lesson from Software Carpentry, in the event that you want to repeat the lessons later:

https://swcarpentry.github.io/python-novice-gapminder/

Xlabel web page

https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.xlabel.html

conda create --name workshop
conda install -n workshop jupyterlab numpy
conda activate workshop