Week 1 Assignment - W200 Python Fundamentals for Data Science, UC Berkeley MIDS

Command line and bash scripting

Objectives:

- Build a file tree using the command line
- Write a short shell script to make a file tree
- Demonstrate uploading files to Github
- Edit a shared file on Github which might include fixing merge conflicts

1.1. Using the command line to build a file tree

In your main classroom Github repository on your local computer make a folder called **SUBMISSIONS**. Under this folder make a folder named **week_01**. This week_01 folder is where all of the submissions for this week will go.

Please use the command line to build a file tree within your week_01 folder as shown below.

```
s1
|---s3
|   |---conf.txt
|---s2
    |---text_chunk1.txt
    |---Advanced
         |---text_chunk2.txt
```

- Record your commands in a text file `answers.txt` and put this file in your week_01 folder
- **s1, s2, s3, Advanced** are directory (or folder) names (for example: under the week_01 folder will be the s1 folder and then the rest of the directory & file structure as shown above)
- Make **conf.txt** contain the sentence (without the quotes):

  ```
  "I love bash scripting."
  ```

- Make **text_chunk1.txt** be the following two lines (without the quotes):

  ```
  "A whole new world"
  "A new fantastic point of view"
  ```

- **text_chunk2.txt** is a ***copy*** of text_chunk1.txt with one extra text line appended (without the quotes)

```
"A whole new world"
"A new fantastic point of view"
"Do you want to build a snowman?"
```

1.2. Writing a Shell script

Shell scripts are files with lines of shell commands. They are designed to be run at the command line terminal and are a full featured language (although no one programs completely in shell scripts). Shell scripts are an automated way of programming tedious tasks like repeated sets of commands that need to be performed. For example, you might write one to run a series of python programs every morning or when you deploy a website. You might write one to automatically create a file structure for a project (like a project template).

- Since you are on different systems choose either the Mac/UNIX instructions **or** the Windows instructions below

Mac/Unix

Only do this if you are on Mac or Unix - if you are using Windows skip to that section!

Write a bash script that generates the above file tree and save it as **make_tree.sh**. To do so, you can simply save the commands you use to generate the file tree in a .sh file. Note: Do NOT use the "cd" command. For folder and file paths, please use "relative paths" instead of "absolute paths". If you are not sure what that means, this website might help: *https://www.computerhope.com/issues/ch001708.htm*

Hint: You should try running your script before you submit it to make sure it works before we grade it. There are several ways of executing bash files. Most simply, you can execute them via a command line call like sh make_tree.sh.

In order to make this file executable, you should:

- Add this to the top of your file: `#!/bin/bash`

  This is called a "shebang" and it points to where your bash scripting is located (in this case bash is in /bin/bash which I can see by calling: **which bash** or **type -a bash**)

- make the file executable (do this on the command line NOT in the file) : `chmod +x make_tree.sh`

  This is modifying the permissions to make the file executable (+x)

Windows

Write a bash script that generates the above file tree and save it as **make_tree.bat**. To do so, you can simply save the commands you use to generate the file tree in a .bat file. Note: Do NOT use the "cd" command. For folder and file paths, please use "relative paths" instead of "absolute paths". If you are not sure what that means, this website might help: *https://www.computerhope.com/issues/ch001708.htm*

Hint: You should try running your bat file before you submit it to make sure it works before we grade it. There are several ways of executing bat files. Firstly, you can execute them via a command line call by simply typing make_tree.bat. You can also just double-click on the file from a Windows folder.

In order to print text to a file from a .bat script, you must type the command slightly differently than we did in the command line.

Instead of: `"this is some text to output" >> output_file.txt`

You will need to use: `@echo this is some text to output >> output_file.txt`

Other commands like cp, mv and touch are different in windows command line syntax also. For example, cp is copy, mv is move and touch is not used in windows. Hint: You can google something like: "mv command in windows command line" or "windows mv equivalent" to find out what these commands are translated to the windows environment.

1.3. Github commits

Please make at least two commits to the github-playground repository. They must be separated by at least one full day and can be small but they must edit a single file called `edit.txt`. This is to show you how to deal with merge conflicts and multiple editors [i.e. people editing the repository]. *If this file does not exist, please go ahead and be the first to create it.*

A few notes:

- To clone into the github-playground, you do not need to set up a "three way" repository like you have for your homework repository. You can instead use the default git command, "git clone" to clone the repository. You can then use just "git pull" and "git push" to pull from and push to the repository.
- You may find these instructions helpful to clone into the repository (note, that the repository name will not have your username but instead will have the course group name): https://help.github.com/articles/cloning-a-repository/
- You may also run into a "merge conflict" if you try submitting your changes to `edit.txt` at the same moment in time as somebody else. These conflicts occur when two people edited the same lines of code at the same time. Git does not know how to resolve the conflict, so it asks the user to help.
- Some merges are easy and will be done automatically however some are more complicated and have to be done manually. This resource can help you resolve difficult merge conflicts: *https://help.github.com/articles/resolving-a-merge-conflict-from-the-command-line/*
- Remember - when in doubt, use "git status". Good luck!

Turn-In Checklist

Please be sure you submit the following:

1. Your folder structure, as discussed in 1.1.
2. The answers.txt file, as discussed in part 1.1.
3. Your shell script, as discussed in part 1.2.
4. Two github-playground commits, separated by at least a day, as discussed in part 1.3.

Items 1, 2 & 3 will be uploaded to your personal github repository under the SUBMISSIONS/week_01 folder. It's a good idea to check using the web interface that the files are uploaded properly!