CS 251: [Code Warrior] git and python: Outlab

- Handed out: 9/18 Due: 9/21 11:00 PM
- Please write (only if true) the honor code. If you used any source (person or thing) explicitly state it. You can find the honor code on the web page.

Overview

The goal of this lab is to introduce version management using **git** which is a powerful version management tool. The days of individually writing useful software programs are over. It's all to be done in a team.
Also this lab will introduce you to **python** as a file manipulation and scripting language.

Pretasks

1. Go through the following link to learn about basic git commands: http://rogerdudler.github.io/git-guide/. Also spend 15 minutes at try.github.com.

   The following is mainly for the inlab.

2. Make sure you link your previous assignment on your `public_html` directory so that the instructor can find it during viva hours.

3. Precursors to `git` involves `patch` and `diff`. Read the man pages of `patch` and `diff`.

4. You must have started to work on your Box2D project. You haven't? Collect all the changes you have made in your Box2D implementation (e.g. the changes in Lab 03) in an area which is owned by the middle roll number in your group. If there are only two members in your group, the middle roll number is that of the lowest roll number.

5. Collect all the documentation related to your updates and place them in an area owned by the lowest roll number in your group.

Tasks

1. **git** Lets go through a simple example of merging with a workflow that gives you a glimpse of what you might use in the real world.

   (a) Change to your home directory and create a folder `remoterepo`. This is where you are going to keep your stuff, a repository. It's called remote because in the real world, it will be present remotely, perhaps on github.

   (b) Initialize a bare repository here. (In real life, this would almost always have been done by the lead architect since most likely you are just modifying some code that already exists).

   (c) Clone `remoterepo` to a location we refer to as `first`.

   (d) Create a file `observeConflict.txt` and on the first line type "For a high jump, you need a long run".

   (e) Commit and push this to `remoterepo` with a suitable commit message.

   (f) Now clone the `remoterepo` again to a new location and name it `second`.

   (g) Within `second`, open `observeConflict.txt` and edit the line to "For a higher jump you need a longer run". Commit with a suitable message.

(h) Now switch back to your first version and edit the line to "For the highest jump you need the longest run". Commit with a suitable message.

(i) Now you have both the repositories ahead of the `remoterepo` (check this using `git status -sb`)

(j) Change to `second` and push to remote.

(k) Change to first and pull the changes. (What happens if you try to push before pulling?) Document what you observe.

(l) Resolve issues if any and perform steps to ensure that `remoterepo` and `first` are in sync.

2. **[Python]** VISA was one of the companies which came for placement to IITB. Their selection process included five written tests followed by an interview. The written tests had up to 5 marks each. Students could decide not to attempt a written test (If they got the written test totally wrong, they may get negative marks.)

   The file `marks.csv` gives the spreadsheet of marks obtained by each of the candidates who appeared for selection. There are some header rows in the spreadsheet. The first row before the actual data in the spreadsheet gives the column names which stand for the various written tests. In the cells, a "NA" means that the candidate did not attempt that particular written test (for fear of getting negative marks). The column named "Sel" tells whether the candidate was finally selected after the interview: this is a binary value (1=selected, 0=not selected). [4 marks each]

   (a) Write a python script `fetch_data.py` to read the file `marks.csv` and print the total number of candidates who appeared for selection. You must look for a number as the first entry in a row, and count such rows to arrive at the answer. Your program should take exactly one command-line argument, the input csv file. If the wrong number of arguments are given, the program should print the correct usage and exit.

   What you are expected to learn: command-line arguments, functions, if-then-else, loops, the csv module, regular expression matching.

   (b) Modify the above program to store the main data (i.e. excluding the header rows) as a list of lists (2D array). And then print the number of candidates by using "len".

   What you are expected to learn: lists, list-of-lists, the len function.

   (c) Modify the above program to make another 2D matrix. This should consist of the rows corresponding to only the selected candidates. Use the filter function to achieve this. Then print the total number of selected candidates by using "len" on this 2D matrix.

   What you are expected to learn: using filter on a list.

   (d) Now add to the above program to make a "dictionary" or a "hashtable" which maps the column name to the column index. To construct this index, you have to use the header row, which you can identify in the program as having "SNo" as the first column value. The program should then loop through the dictionary to print the column index of each column name.

   What you are expected to learn: constructing and looping through dictionary or hashtable structures in python.

   (e) Modify the above program to store an extra column in each row, which is the `total_marks` obtained by the candidate (count "NA" as 0 marks). Then sort the list-of-lists by the total marks obtained by each candidate. Print this sorted list onto a new file `marks_sorted.csv`

   What you are expected to learn: sorting a list using the "sort" command, and using a lambda function to specify the key.

3. **[python]** [Bonus question, 25% extra credit] Central Board For Secondary Education (CBSE) provides 10th marks formatted for a "dot matrix printer". A sample of such a data is provided in the data folder.

View this file in a text editor. You will find that the first few lines contains the school name and a header that explains the fields. Also notice on scrolling, you can find the same information (i.e., school name and header) multiple times in the file. This is great if you actually want to print the file on the dot matrix printer.

The task here is to process the raw file and convert into a csv file such that all the data for one student comes in a single line. Other than the student's record (possibly spanning multiple lines), everything can be considered as irrelevant, and needs to be ignored. The csv file should be importable in standard spreadsheets such as Google Sheets, or LibreOffice. Your code will be tested for the given sample input file.

For better understanding of the fields in the record, refer to the sample marksheet image given to you.

The basic task (15%) is to get the student data. The extra credit task (10%) is to get the headers also.

Note: Student where the final result is EIOP (eligible for improvement of performance), the CGPA field will not be available. The blank field should be replaced by NA(not available). For understanding this case, view the first student's record in both sample input and output file. For any other special cases, refer to the given sample files.

Your output file should be named as `cbseProcessed.csv`.

Submission Guidelines:

Submit the following documents:

1. **Task 1:** Submit the `git log` output displaying all the changes that you had performed in remote repository as `log.txt` with a `readme.txt` file.
2. **Task 2:** `fetch_data.py` and `marks_sorted.csv` files.
3. **Task 3:** `text_to_csv.py`.

Do not forget to put `readme.txt` file in a folder. The folder and its compressed version should both be named `lab07_groupXY_final`. Hence, you submit a tar.gz named `lab07_group07_final.tar.gz` if your group number is 7.

How We will Grade You

- Honor Code and package complete in all respects +2. **Incorrect or incomplete -2**.
- For Question 2, whenever stdout is required, print it with a appropriate message that indicates the **part no**. In some parts, same data is asked to print as output, so grading will be based on your code.
- In Question 3 bonus point will only be given only if the headers are extracted from the input file.Hardcoding the header will not fetch bonus points.