Computer Science I for majors by James Tam Return to the course web page

CPSC 231: Assignment 1

Due Friday May 26 at 5 PM

New Concepts to be applied for the assignment

- Learning how to complete common tasks using UNIX and creating a complete Python program from the ground up

Part I: Introduction to UNIX, using common commands (Each of the graded steps in this part is worth 0.1 grade point (GPA) x 12 steps = 1.2 GPA)

On the UNIX workstations in the Computer Science lab complete the following steps. (JT: creating a typescript file literally records everything: output shown onscreen as well as every character that you type into a text file. Pressing the backspace or delete key will not delete what's recorded into the file, instead that keyboard character will be translated to a gibberish character. If you make a typographical error then DON'T try to delete it during the typescript. Instead just hit enter once or twice - so there will be a few blank lines in the script file - and then re-enter the command properly. Also you will create a directory called '`231`' under your home directory, this is not the same as the official course directory which is located under `/home/231`. Don't mix the two up!). In to make marking reasonable for your TA make sure that you **complete each step exactly as-is and in same order as specified below**. If you deviate from either requirement then you may end up with few (if any) marks.

1. Login to your UNIX account*
2. Start a script session *
3. In your current location make a new directory called '`231`'
4. Enter the 231 directory.
5. Show your location (path) in the UNIX file system.
6. Go to the official course directory for CPSC 231 (`/home/231`)
7. Go to the directory for the CPSC 231 Assignment 1 (`/home/231/assignments/assignment1`)
8. Display the contents of the file called '`index.txt`' onscreen (*JT's hint for this step: don't try to view it with an editor such as 'NEdit' or 'vi'*).
9. Try to delete the '`index.txt`' file *[JT's Hint: you aren't supposed to be able to complete this step successfully but instead you should get an error message. Take away learning: you only have limited file permissions in the course directory]*
10. Copy the '`index.txt`' file to the newly created '`231`' directory **that you created in your home directory**
11. Go to the '`231`' directory that is located in your home directory
12. Show the contents of the '`231`' directory. (It should include the file '`index.txt`' now.)

13. Delete the '`index.txt`' file from your `231` directory [*JT's comment: it should work now because it's a directory that you created rather than the course directory*]

14. Show the contents of the '`231`' directory. (The file should be gone now).

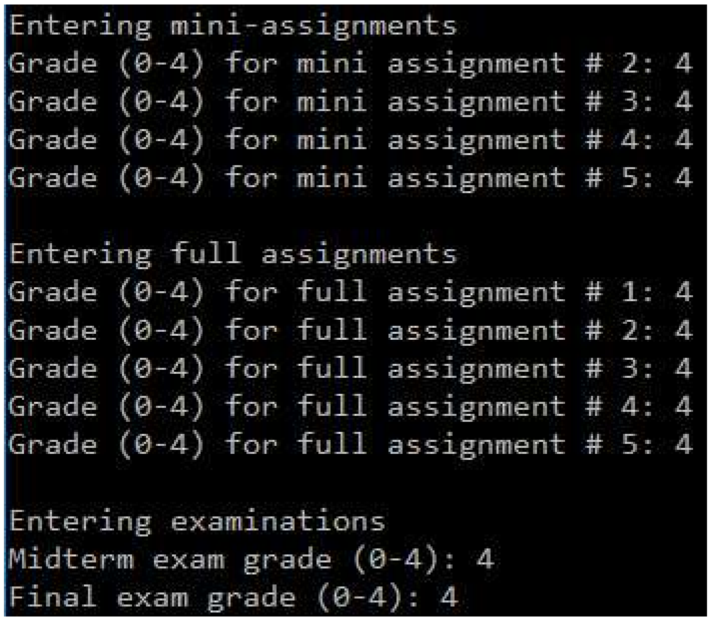
15. End the script session*

16. Submit your assignment (the file called "`typescript`" using D2L as described on the course web page [D2L link]).*

The lines marked with a star '*' don't directly yield any marks but must still be completed in order for Part 1 of the assignment to be marked. You will either receive full marks on a step in Part I or no credit. You don't need to include contact information in the typescript (partly because some of it will be displayed in Step #4 but mostly because you will be including this information in Part II).

Part II: Creating your first Python program (*worth 2.8 GPA*)

Write a Python program that will calculate your term grade point based on the weights specified in the course admin notes, summary in the following PDF. The program will prompt the user for the grade point for each component and calculate the weighted grade point for that component. Note: just like with your actual term grade each component will be awarded a value from 0 - 4.0 GPA (some will allow for 'bonus' marks of up to 4.3 GPA). However your program does not have to error check that the user actually entered a value within this range because you will not yet have yet learned how to do this yet. Just assume that the user enters a value within the correct range. Also your program doesn't have to check for invalid types of information being entered (e.g. the user enters a string instead of numbers for a grade point). You should however document these things as current program limitations.



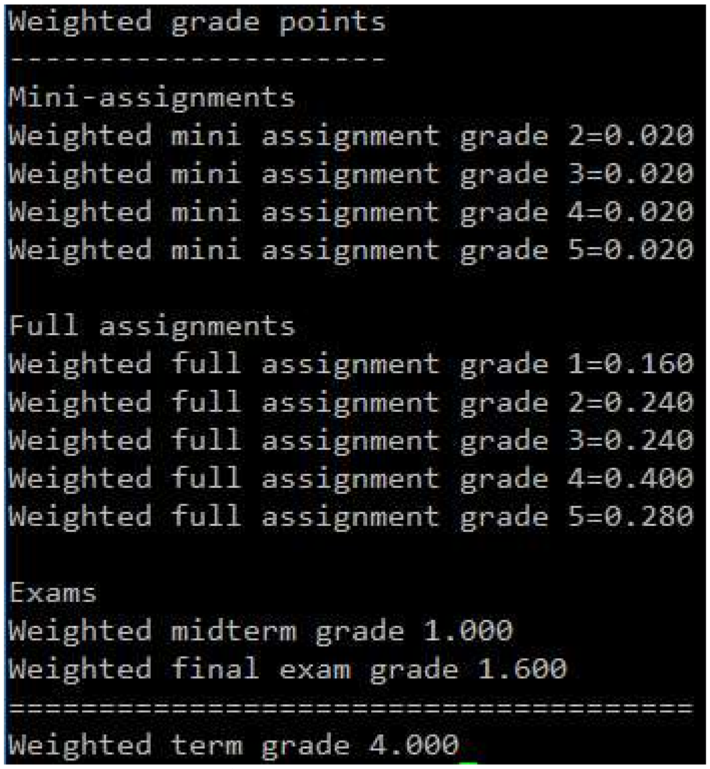
```
Entering mini-assignments
Grade (0-4) for mini assignment # 2: 4
Grade (0-4) for mini assignment # 3: 4
Grade (0-4) for mini assignment # 4: 4
Grade (0-4) for mini assignment # 5: 4

Entering full assignments
Grade (0-4) for full assignment # 1: 4
Grade (0-4) for full assignment # 2: 4
Grade (0-4) for full assignment # 3: 4
Grade (0-4) for full assignment # 4: 4
Grade (0-4) for full assignment # 5: 4

Entering examinations
Midterm exam grade (0-4): 4
Final exam grade (0-4): 4
```

Figure 1: Data entry assumes numeric grade points (range of 0-4)

Finally the program will display the weighted grade points for: each of the mini assignments, each of the regular full assignments, the midterm and the final exam. This output must appear at the very end of your program with each weighed grade on it's own line (see image). Weighted grade points should only display three places of precision for the rational component (although all rational digits can be used when performing calculations - no rounding is needed this is just for the formatting of output). All output should be neat and presentable. You should employ good naming conventions for variable names and be sure to include the required contact information (your name, student identification number and tutorial number) in the header of the program. J*T's final hint: don't try to type in the whole program all at once, especially when calculating and* *displaying all the assignment grades. Try it for just one component, make sure it works before adding the information for the other components.*



```
Weighted grade points
---------------------
Mini-assignments
Weighted mini assignment grade 2=0.020
Weighted mini assignment grade 3=0.020
Weighted mini assignment grade 4=0.020
Weighted mini assignment grade 5=0.020

Full assignments
Weighted full assignment grade 1=0.160
Weighted full assignment grade 2=0.240
Weighted full assignment grade 3=0.240
Weighted full assignment grade 4=0.400
Weighted full assignment grade 5=0.280

Exams
Weighted midterm grade 1.000
Weighted final exam grade 1.600
======================================
Weighted term grade 4.000
```

Figure 2: Display of weighted grade points

D2L configuration:

- Multiple submissions are possible for each assignment: You can and should submit many times before the due date. D2L will simply overwrite previous submissions with newer ones.
- **Important**! Multiple files can be submitted for each assignment. I am allowing you to submit multiple files for each assignment so **don't archive/compress multiple files using a utility such as zip**. However, this means that **TAs will only mark the latest versions** of each file submitted via D2L. Even if the version of a document that you want marked has been uploaded into D2L if it isn't the latest version then you will only get marks for the latest version. (It's unfair to have the TAs check versions or to remark assignments because marking is enough work as-is).

Marking

- Assignments will be marked by your tutorial instructor (the "Teaching Assistant" or "TA"). When you have questions about marking this is the first person that you should be directing your questions towards. If you still have question after you have talked to your TA, then you can talk to your course (lecture) instructor.
- As well as being marked on whether "your program works" you will also be marked on non-functional requirements such as style and documentation. Consequently this assignment will include a separate [marking checklist]. Besides seeing your grade point in D2L you can also see the detailed feedback that your TA will enter for each student. You can access the grading sheet in D2L under `Assessments->Dropbox` and then clicking on the appropriate assignment link. If you still cannot find the grading sheet then here is a [help link]

Points to keep in mind:

1. **Due time:** All assignments are due at 5 **PM** on the due dates listed on the course web page. Late assignments or components of assignments will not be accepted for marking without approval for an extension beforehand. Alternate submission mechanisms (non exhaustive list of examples: email, uploads to cloud-based systems such as Google drive, time-stamps, TA memories) cannot be used as alternatives if you have forgotten to submit work or otherwise have not properly submitted into D2L. **Only files submitted into D2L by the due date is what will be marked**, everything else will be awarded no credit.

2. **Extensions** may be granted for reasonable cases by the course instructor with the receipt of the appropriate documentation (e.g., a doctor's note). Typical examples of reasonable cases for an extension include: illness or a death in the family. Example cases where extensions will not be granted include situations that are typical of student life: having multiple due dates, work commitments etc. Tutorial instructors (TA's) will not be able to provide extension on their own and must receive permission from the course instructor first.

3. **Method of submission:** You are to submit your assignment using D2L [help link]. Make sure that you [check the contents of your submitted files] (e.g., is the file okay or was it corrupted, is it the correct version etc.). It's your responsibility to do this! (Make sure that you submit your assignment with enough time before it comes due for you to do a check).

4. **Identifying information**: All assignments should include contact information (full name, student ID number and tutorial section) at the very top of your program in the class where the '`main()`' method resides (starting execution point). (Note other documentation is also required for most assignments).

5. **Collaboration**: Assignments must reflect individual work; group work is not allowed in this class nor can you copy the work of others. For more detailed information as to what constitutes academic misconduct (i.e., cheating) for this course please read the following [link].

6. **Execution**: programs must run on the computer science network running Python 3.x. If you write you code in the lab and work remotely using a remote login program such as Putty or SSH. If you choose to install Python on your own computer then it is your responsibility to ensure that your program will run properly here. It's not recommended that you use an IDE for writing your programs but if you use one then make sure that you submit your program in the form of text "`.py`" file or files.

7. **Use of pre-created Python libraries**: unless otherwise told you are to write the code yourself and not use any pre-created functions. For this assignment the usual acceptable functions include: `print()`, `input()` and the 'conversion' functions such as `int()`, `float()`, `str()`. Look at the particular assignment description for a list of other classes that you are allowed to use and still get credit in an assignment submission.