PSU CS 161 - Introduction to Programming and Problem Solving (Winter 2013)

Homework 5: Payroll System

Due Wednesday, March 20 at 12:30AM(Tuesday Night)

You are going to program a very simple payroll system. You can use any example code given during lectures (see the zip files for the lectures on Functions and Files).

Input File

You will have a file called "payroll.txt" in the same directory as your program files. The format of the payroll file is as follows:

<last name>,<first name>,<rate>,<hours>,<gross pay>

each employee is represented by a line with five fields separated by commas.

Here is an example file:

```
Nelson, Frank, 10.0, 20.0, 200.0
Harrison, Johanna, 15.5, 40.0, 620.0
Mannon, Ethyl, 25.5, 35.5, 905.25
```

WARNING: Do not write your program to only work with this file as our test script will NOT be using the exact same values as above, just the same format. In fact, it's guaranteed NOT to have 3 lines but it will be short (less than 200 lines). Your program should be able to take ANY number of lines in the above format (separated by commas) with any reasonable values (ie. the first two fields will be string values and the last three fields will be floats).

NOTE: When debugging your program, I would suggest you create your own test files (Don't upload these test files to Moodle; we only want your code files).

Menu Handling

After your program loads this file on start-up, then the following menu is displayed:

a - add a new employee
d - delete an employee
m - modify an employee
p - print all employees

You will then ask the user for a choice. You can use the skeletal program 'full_functions.py' from the Functions lecture (see Feb 19 for the zip file of code for the functions lecture) but you will need to modify it to work with the homework (ie. it shouldn't print anything about songs :-)). Also, feel free to take out the code that asks if the user wants to change their choice (it's more annoying than I thought).

Choice Descriptions

The choices do the following:

a - add an employee: Prompt the user for last name, first name, rate and hours. Add employee to list of employees.

d - delete an employee: Prompt the user for an index into the list. Then delete employee at this index.

m - modify an employee: Prompt the user for an index into the list and then prompt for a field (“last name”, “first name”,”rate”, “hours”). Then prompt for a new value. Replace the new value for the given field for the employee at the given index.

p - print all employees: All employee information is printed to the screen in a nice format (your choice of what is meant by 'nice'). Also include the index into the list for debugging purposes (ie. so you can see whether the right employee info got changed.)

The program should update the grossPay field whenever an employee's pay rate and hours changes. This way when the print statement prints gross pay it will always be the correct value (also, it will be correct when the program writes to payroll.txt at the end).

Ending the Program

After the program is done with processing a choice, the user should be asked to continue. If yes, then the program should re-display the menu. If no, the program writes the payroll information back to "payroll.txt". This should be done by re-opening the “payroll.txt” file with a “w” mode which erases whats in the file already. The outputted format is the same as the inputted format so that when the program is re-run, the new “payroll.txt” can be read and parsed correctly.

Requirements

First, your program needs to work correctly, In addition, to get full credit, your program needs the following:

- It needs to have one function called main() with no parameters. This function is the starting point to run your program. The program should call main() when run from shell. It should NOT call main if imported from another file. (see full_functions.py for how to do this).
- It needs to have at least 6 functions that you wrote (it will probably have much more than that). Note that you will need to load and save the file plus do four tasks so putting these into separate functions is a good idea.
- No function body or any global code should be more than 15 lines long. (NOT counting blank lines and comments). If the function starts getting longer, break it into two or more functions.
- ***Do NOT change global variables inside of functions (don't use the global statement). Pass variables into functions as parameters.***
- ***The variable holding the employee information should NOT be global.*** Create it by reading it in inside main (the reading part can be done from another function and the list returned to main). Pass it to functions when it's needed and write the results out to file in main (or, again, pass it into another function as a parameter).
- You should have two files: One file to hold your main function called main.py. Another file to hold your functions and any global constants called functions.py. Zip these two files into one file (for Mac users, use whatever the equivalent of zip is) before uploading them to Moodle. You don't need to add your own payroll.txt file as we will use our own version of payroll.txt.

You are free to use any combination of lists, dictionaries or classes to store the employee information as long as everything works correctly. ***You don't need to use classes for this assignment; you can do this assignment with a combination with lists and dictionaries.***

NOTE: If using classes, it is best to design one class to store ONE employee's info. Then create a list of objects of this class. In other words, don't store the list of employees inside the class itself.

Available from:	Thursday, 7 March 2013, 06:00 PM
Due date:	Wednesday, 20 March 2013, 12:30 AM

Upload a file

You are logged in as Nguyen Quan (Logout)

CS161