|  |  |  |
| --- | --- | --- |
| **ITMD 413/513** | ***Open Source Programming*** | Lab 5 |

**PROJECT File Processing - The Payroll Program 100 points**

**Objective** This lab will consist of options to users to view, add, modify or delete payroll record data. Reference to Section 6.3 of the text for nice examples on how to handle modifications to a file.

***PROJECT DESCRIPTION***

Download the accompanying **employees.txt** file from your course Web site and add it the same program directory you have for this lab’s Python source file. You will notice the format of file has layout is follows:

**<first name> <last name> <rate> <hours>**

where each line represents an employee’s name and payroll record data.

As you can see, there are a total of 4 employees in the file with pertinent payroll record data.

For your programming tasks perform the following:

Create menu options to process payroll as follows:

• Display a gross payroll report for all employees

• Display a gross payroll report for just one select employee.

• Add an employee record

• Delete an employee record

• Modify an employee record

• Exit program

Create with a loop that will show a menu of the above options to the user where if the user chooses:

Option 1 - A report of pertinent employee data will print to the screen

Option 2 - It will allow for the user to choose an employee by name, where if the name exists in the file, pertinent employee data will print to the screen

Option 3 - Prompts the user for their name ( first name, then a last name ) followed by their rate of pay and hours worked and have that data added to your payroll file

Option 4 - Prompts the user for their name ( first name, then a last name ) and if a record exists matching the employee’s first name and last name in the file, it will be deleted.

Option 5 - Prompts the user for their name ( first name, then a last name ) and if a record exists in the file matching the employee’s first name and last name, the user will be allowed to modify any data items ( rate and / or hours ) within the particular record.

Option 6 quits the program. Display a cordial message to the user upon the termination

of the program.

Include the following processing logic in your program ( make use of user defined functions that will fire for each menu option for solid organization of your app ) :

* For menu option 1: Use a function called **printall()**, to display all Employee data will be displayed as follows:

First name Last name and the gross pay, together on one line of output.

* For menu option 2: Use a function called **printEmp()**, to display only a single Employee’s data chosen by the user to be displayed as follows:

First name Last name and the gross pay, on one line of output.

**\*Note-** Gross pay is calculated simply as ( rate **x** hours ) worked.

Make sure gross pay data for either option 1 or 2 , is appropriately formatted

( currency style ) when displaying your report information.

* For menu option 3: Use a function called **addEmp()** to add (or append) record data to your file
* For menu option 4: Use a function called **deleteEmp()** to delete record data from your file
* For menu option 5: Use a function called **modifyEmp()** to modify record data from your file

//

* For either menu option 6: Use a function called **exitApp()** to exit of the program. You can evoke Python’s built - in exit() function to accomplish this

Note that for options 4 and 5 you will need to write to a temporary file each time to reflect changes to the original file. Then merely remove the temporary file and rename your temp file to the original file name to reflect the changes.

For example, the coding logic action for option 4 - record deletion:

• **import os**   
 **# needed for the removal and renaming functions**

• open original file ( **employees.txt** ) for reading

• open up a temporary file ( example **temp.txt** ) for writing

• start a **while()** loop to read all record data as parsed data

• within the loop, check record if a name exists and needs deleting, if the name exists do not write the record to the temp file, else write the record that does not need deleting to the temp file

• exit the **while()** loop

• close both the data files

• **# delete original employees.txt file**

**os.remove("employees.txt")**

• **# rename the temporary file**

**os.rename("temp.txt", "employees.txt")**

• **# display a message if the record was not found or**

**# if it has been found, message the user that**

**# the file has been updated!**

• Continue in the same manner as the example code logic above for option 4 for option 5 to allow for file modification.

Include any **error traps** ( **exceptions** ) you deem necessary in your program with regards to the processing of your file / data, etc.. Examples may include file not found, or record not found, employee already exists, etc.



Helpful Hints: When adding file data to the **employees.txt** file make sure to delimit each written item to the file by a space. Also make sure to add a new line character at the end of each record added to the file.

When processing payroll data, simply use a split function to parse the data by a space and have the resulting fields stored into some list easily read for processing.



Make sure to fully document your functions and include a solid program description describing your program.

**Run and Test Your Completed Payroll Application**

Test transactions of your program for various scenarios / menu options as follows for full credit.

For full credit, execute your program at run time **in the order shown below** as

follows:

**(1)** Run option 3 and add in yourself as a employee when prompted. Include your name appropriately, rate of pay and hours worked and have the data added to your **employees.txt** file.

**(2)** Run option 4 and delete Peter Piper from the **employees.txt** file.

**(3)** Run option 4 again and attempt to delete Peter Jones, a former employee from the file. There should be an error message to the screen to show the employee does not exist.

**(4)** Run option 5 modify Manish Mannon and change his pay rate to 21.22 .

**(5)** Run option 2 and show the employee result for Manish Mannon.

**(6)** Run option 1 and show all employee results.

**Grads**

Include overtime pay for any gross pay output. Overtime consists of rate multiplied by time and a half for any hours exceeding 40 hours for an employee. Also, allow the user to include a file name to process the payroll from. Show an example error message when no payroll file is found. Finally segregate all your functions into another Python file and have your main program import that module for processing.

Submit your .py file(s) plus a snapshot of your output showing all the above transactions to a doc file, and a snapshot of your employees.txt file into your doc file, after it has been modified ( i.e., when options 3 - 5 above were executed ) for full credit. Label snapshots accordingly.

**Suggested Program Starter Code**

|  |
| --- |
| **def** menu() :  pstr = **"Choose from the following payroll choices\n"** pstr += **"(1) A gross PR payroll report for all employees\n"** pstr += **"(2) A gross PR payroll report for a single employee by name\n"** pstr += **"(3) Quit Program"** print (pstr)  **def** main() :  empFile = open(**"employees.txt"**, **"r"**)  line = empFile.readline()  answer = **""** menu()  choice = int(input(**"Enter Menu Choice Now! "**))  **if** choice == 1 :  empFile = open(**"employees.txt"**, **"r"**)  **for** line **in** empFile :  answer += line  print (answer)  **if** (choice == 2) :  name = input(**"Enter in a name to search employee "**)  **while**(line != **""**) :  line = line.split(**" "**)  rate = float(line[2])  hours = float(line[3])  **if** (name == line[0]+**" "**+line[1]):  print(name, **"$%4.2f"** % (rate \* hours))  **break** line = empFile.readline()  empFile.close()  main() |