The University of Hong Kong

Department of Computer Science

Academic Year 2018 – 2019 Semester 1

COMP1117A: Computer Programming Assignment 3

Overview

We have discussed fundamental computer programming such as loops, functions lists, etc. in Python recently. In this assignment, you are going to implement a staff salary management system for a company. Given a data file "***Assignment3-DataFile.txt***" and the code block of main() function, you are going to complete ***seven self-defined functions*** with the requirements below.

Reminders

This assignment involves a lot of console input/output. You are reminded that the VPL system on HKU Moodle evaluates your program with a full score under the condition that your ***program output is the EXACT MATCH of the expected output***. In other words, any additional or missing space character, tab character, newline character, etc. will be considered during the evaluation of your program. Also, you are suggested to ***make more test cases on your own*** for testing your program.

Assignment Specifications and Requirements

1. main() Function and loadData() Function

In this assignment, you are given a data file "***Assignment3-DataFile.txt***" which can be downloaded under ***Assignments*** Section of COMP1117A on HKU Moodle.

The content of "***Assignment3-DataFile.txt***" is as follows:

```
Billy 12300 11700 11100 10300 10400 14800 14900 13600 12300 14600 13500 14900
Betty 11900 11800 15000 13000 12500 14000 11500 11100 12400 10900 20000 10300
Apple 13600 13700 10900 11900 12000 14900 13600 12400 11700 13700 10300 13900
Kelly 11400 11600 14400 10800 12700 14900 13300 12700 11900 13800 11800 13500
Gigi 14400 12400 11600 11600 12800 13600 11500 14300 13200 10200 14400 14400
```

For ***EACH row of*** staff salary ***records***, it has a specific saving format:

<NameOfStaff><space><Salary1><space><Salary2><space><Salary3><space><Salary4><space><Salary5><space><Salary6><space><Salary7><space><Salary8><space><Salary9><space><Salary10><space><Salary11><space><Salary12><newline>

The 1st column of data is the name of staff (i.e., Billy) ***with ONE word only***. From 2nd column to 13th column of data is the salary for each month (from January to December) for that particular staff. For example, Billy earns a salary of 12300 in January and Kelly earns a salary of 12700 in August. In between 1st column and 13th column of data is ***separated by a space character***. After each row of staff salary records (i.e., After 13th column of data), there is ***a new line character***.

The code block of main() function is as follows (You are ***NOT allowed to change any line*** of the programming statements in main() function):

```
def main():
    salary_records = []
    salary_records = loadData(salary_records)
    command = input("Command:")

    while command != 'Quit':
        if command == 'Print':
            printSalaryRecords(salary_records)
        elif command == 'Total':
            totalSalaryPaid(salary_records)
        elif command == 'TotalEach':
            totalSalaryForEach(salary_records)
        elif command == 'WhoIsMax':
            maxTotalSalary(salary_records)
        elif command == 'MaxEach':
            maxTotalSalaryForEach(salary_records)
        elif command == 'Sort':
            sortRecords(salary_records)
        else:
            print("Please input a valid command!")
        command = input("Command:")
    print("BYE")

main()
```

In the main() function, there is a function call ***loadData(salary_records)*** before getting the user input. The implementation of ***loadData(salary_records)*** is as follows:

- It accepts one parameter which is an initially empty list "***salary_records***".
- In the function body of ***loadData(salary_records)***, it:
  - opens the data file "***Assignment3-DataFile.txt***",
  - reads the data,
  - stores the data in the initially empty list "***salary_records***".
- Finally, it returns the list "***salary_records***" to the main() function.
- You are NOT required to check whether the data file exists in the same directory of the Python File. You can always assume that the program reads the data file without any errors.
- You are NOT required to check whether there is any data missing. You can always assume that the data file is well-saved with 13 columns of data. 1st column of data must be the name of staff while the others are salary records from January to December for that particular staff.

The main() function should ***get a string input*** from the user ***repeatedly*** until getting "***Quit***" (to terminate the program) that indicates which function is going to be called in the main() function. You are reminded that each string input is ***CASE-SENSITIVE***. With respect to this, the table below provides more information for you:

String Input	Description
Print	To call printSalaryRecords() function
Total	To call totalSalaryPaid() function
TotalEach	To call totalSalaryForEach() function
WhoIsMax	To call maxTotalSalary() function
MaxEach	To call maxTotalSalaryForEach() function
Sort	To call sortRecords() function
Quit	To quit from the program

2. printSalaryRecords() Function

- It prints the salary records from January to December for each staff.
- It accepts one parameter which is the list "***salary_records***".
- The print() statement is provided for printing the header of salary records:
  - print("Name \t Jan \t Feb \t Mar \t Apr \t May \t Jun \t Jul \t Aug \t Sep \t Oct \t Nov \t Dec")
- For printing the salary records for each staff, it has a specific format which is similar to the saving format of the data file EXCEPT <\t>:
  - <NameOfStaff><\t><Salary1><\t><Salary2><\t><Salary3><\t><Salary4><\t><Salary5><\t><Salary6><\t><Salary7><\t><Salary8><\t><Salary9><\t><Salary10><\t><Salary11><\t><Salary12><\t><newline>
- Sample Output (Text in ***green*** is the user input):

```
Command:Print
Name    Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct    Nov    Dec
Billy   12300  11700  11100  10300  10400  14800  14900  13600  12300  14600  13500  14900
Betty   11900  11800  15000  13000  12500  14000  11500  11100  12400  10900  20000  10300
Apple   13600  13700  10900  11900  12000  14900  13600  12400  11700  13700  10300  13900
Kelly   11400  11600  14400  10800  12700  14900  13300  12700  11900  13800  11800  13500
Gigi    14400  12400  11600  11600  12800  13600  11500  14300  13200  10200  14400  14400
Command:Quit
BYE
```

3. totalSalaryPaid() Function

- It calculates the sum of 12-month salary paid to ***ALL staff***.
- It accepts one parameter which is the list "***salary_records***".
- You can always assume that each salary record is a positive integer. No validation is required for each salary record.
- Sample Output (Text in ***purple*** is the user input):

```
Command:Total
Total salary paid: 768600
Command:Quit
BYE
```

4. totalSalaryForEach() Function

- It calculates the sum of 12-month salary paid to ***a particular staff***.
- It accepts one parameter which is the list "***salary_records***".
- You can always assume that each salary record is a positive integer. No validation is required for each salary record.
- In the function body, it gets the user input for indicating which particular staff.
  - If the staff is in the list "***salary_records***", then prints the sum of 12-month salary paid to that staff. (Refer to the Sample Output)
  - Otherwise, prints the error message. (Refer to the Sample Output)
- Sample Output (Text in ***purple*** is the user input):

```
Command:TotalEach
Which staff?Kelly
Kelly earns 152800
Command:TotalEach
Which staff?Justin
Justin not found
Command:Quit
BYE
```

5. maxTotalSalary() Function

- It finds the staff with the highest salary.
- It accepts one parameter which is the list "***salary_records***".
- You can always assume that each salary record is a positive integer. No validation is required for each salary record.
- It is possible that there is ***MORE THAN ONE staff with the highest salary***.
- For printing the result, we have three cases with corresponding formatting.
  - Case 1: We have one staff with the highest salary.
    - Sample Output (Just an example ***WITHOUT referring to the data file***)

```
Command:WhoIsMax
Apple earn(s) the most
Command:Quit
BYE
```

  - Case 2: We have two staff with the highest salary.
    - Sample Output (Just an example ***WITHOUT referring to the data file***)

```
Command:WhoIsMax
Billy and Betty earn(s) the most
Command:Quit
BYE
```

  - Case 3: We have more than two staff with the highest salary.
    - Sample Output (***WITH referring to the data file***)

```
Command:WhoIsMax
Billy, Betty and Gigi earn(s) the most
Command:Quit
BYE
```

6. maxTotalSalaryForEach() Function

- It finds which month(s) a particular staff earns the highest salary among 12 months.
- You can always assume that each salary record is a positive integer. No validation is required for each salary record.
- In the function body, it gets the user input for indicating which particular staff
  - If the staff is in the list "***salary_records***", then finds which month(s) that staff earns the highest salary among 12 months and prints the result. (Refer to the Sample Output)
  - Otherwise, prints the error message. (Refer to the Sample Output)
  - Sample Output (Text in ***purple*** is the user input):

```
Command:MaxEach
Which staff?Justin
Justin not found
Command:Quit
BYE
```

- It is possible that there is ***MORE THAN ONE month with the highest salary***.
- For printing the result, we have three cases with the corresponding formatting.
  - Case 1: We have one month with the highest salary.
    - Sample Output (Text in ***purple*** is the user input):

```
Command:MaxEach
Which staff?Betty
Betty earns the most in November
Command:Quit
BYE
```

- Case 2: We have two months with the highest salary ***in chronological order***.
  - Sample Output (Text in ***purple*** is the user input):

```
Command:MaxEach
Which staff?Billy
Billy earns the most in July and December
Command:Quit
BYE
```

- Case 3: We have more than two months with the highest salary ***in chronological order***.
  - Sample Output (Text in ***purple*** is the user input):

```
Command:MaxEach
Which staff?Gigi
Gigi earns the most in January, November and December
Command:Quit
BYE
```

7. sortRecords() Function

- It sorts a list ***first by the sum of 12-month salary in ASCENDING order***, ***then by the staff name in ALPHABETIC order*** ***with implementing Selection Sort learnt in the lecture***.
- It accepts one parameter which is the list "***salary_records***".
- You can always assume that each salary record is a positive integer. No validation is required for each salary record.
- You are NOT required to update the list "***salary_records***" after sorting. For simplicity, you can create a new list in the function to store the sorting result and print it.
- For printing the result, it has a specific format for each row.
  - <NameOfStaff><space><sumof12MonthSalaryOfThatStaff><newline>
  - Sample Output (Text in ***purple*** is the user input):

```
Command:Sort
Apple 152600
Kelly 152800
Betty 154400
Billy 154400
Gigi 154400
Command:Quit
BYE
```

Submission

- Your program ***must follow the format*** of the sample input / output strictly.
- Your program ***should be properly indented and documented***.
- You should only submit source codes (*.py), not executables (even if you know how to compile a Python program into an executable).
- You must write down your university number at the beginning of your program (in the comments).
- Please note that we would use another set of test cases to grade your programs.
- Submit your program electronically using the Moodle system to Assignment 3 under Assignments section. ***Late submission will NOT be accepted.***

Plagiarism

Plagiarism is ***STRICTLY PROHIBITED***. No mark will be given for this assignment to both the source and the copier once caught. The following constitutes an act of plagiarism:

- Submitting the work of another student as your own, or asking someone else to do your assignment;
- Direct copying;
- Study someone else's program and then rewrite it as your own;
- Using a substantial part of other's work in your programs;
- Providing your assignment as the source for any of the above actions.