CS/CYCS1110 (Python) –Fall 2021
Programming Project #4

Due Date (A Two-week Project) 100 points	
Tuesday Labs	**10/26/21 @ 11:59pm**
Wednesday Labs	**10/27/21 @ 11:59pm**

Project Objectives

- Use file processing and exception handling.
- Use string slicing, operations, searching, testing and manipulating.
- Use functions, parameter-passing, return values.
- Use if/elif/… else.
- Use nesting loops/input validation.
- Use incremental development to write and test your program.
- NO GLOBAL VARIABLES ALLOWED

- Use proper Python naming conventions: `variable_name`, `function_name`
    - Function names must describe what the function does.

Project Overview

Consider a small bank which uses a text file to maintain information about its customers. Each customer record in that master file is stored on one line of the file and contains the following information:

- Account number (6 characters)
- Account balance (10 characters)
- Account holder (remaining characters on the line)

There is exactly **one** space (blank) between the fields.

Account numbers range from 100000 to 999998 (inclusive). The account number 999999 is reserved as a sentinel to mark the end of the customer records (the last line of the file contains "999999").

Account balances range from **0.00** to **9999999.99** (inclusive).

Design, implement and test a Python program which allows the bank to interactively enter transactions and produces a new (updated) master file.

Deliverables

The deliverable for this assignment is the following file: Note/Students must develop this project on Codio

proj04_LastName.py

Provides this information in your project:

```
# Project No.:
# Author:
# Description:
```

Project Structure

1. The program will prompt the user to enter a file name prefix. That prefix will be used to generate the names of the two files by appending "_old.txt" and "_new.txt" to the prefix. For example, the prefix that you should provide is "customers". The names will be "customers_old.txt" (the old master file) and "customers_new.txt" (the new master file).

If the old master file cannot be opened for reading or the new master file cannot be opened for writing, the program will display an appropriate error message and halt.

2. For each customer record in the old master file, the program will:

- Display the customer information
- Prompt the user for any transactions related to that customer and process those transactions
- Write the customer record to the new master file (unless the account has been closed)

The list of valid transactions is given below.

The format of the new master file will be the same as the format of the old master file: each customer record will contain the fields specified above, and the last line of the file will contain "999999".

3. The program will recognize the following transaction codes:

- **d** – deposit
- **w** – withdrawal
- **c** – close
- **a** – advance to next customer

For code "d", the program will prompt the user for the amount of money to be deposited into the customer's account. That amount will be added to the account balance, as long as the new total does not exceed the upper bound.

For code "w", the program will prompt the user for the amount of money to be withdrawn from the customer's account. That amount will be subtracted from the account balance, as long as the new total does not exceed the lower bound.

For code “c”, the program will delete that customer record from the master file, as long as the account balance is zero. When an account is closed, the program will advance to the next customer in the old master file.

For code “a”, the program will advance to the next customer in the old master file.

If a transaction is not valid, the program will display an appropriate error message and ignore that transaction.

4. The program will assume that an existing master file is error-free.

5. The program will detect, report and recover from all errors related to user inputs.

6. The program will include three functions which help decompose one line of the old master file into the three separate items:

- **def get_number(one_line):** given one line of the master file, return the account number
- **def get_balance(one_line):** given one line, return the account balance (as a float)
- **def get_name(one_line):** given one line, return the account holder’s name (as a string)

Project Notes

1. You may **not** use any collection (such as a list, dictionary or map) in your program.

2. Be sure to prompt the user for the inputs in the specified order. Also, your program cannot prompt the user for any other inputs.

3. An example master file is given below, where the first digit of the account number is the first character on each line:

```
100000   43736.57 Rossum, Guido V.
100789 5681745.99 Eich, Brendan
320056       5.01 Ritchi, Dennis MacAlistair.
650430    2398.12 Wall, Larry
999999
```

4. An example of a series of user inputs related to the example master file is given below:

```
w
1000.00
w
500.50
d
200.00
a
a
```

```
w
5.01
c
a
```

The new master file resulting from that series of transactions is given below:

```
100000   42436.07 Rossum, Guido V.
100789 5681745.99 Eich, Brendan
650430    2398.12 Wall, Larry
999999
```

5. Consider the situation where the user enters "a" (and only "a") for each customer in the old master file. The new master file which results from that series of transactions is identical (character by character) to the old master file.

6. As noted above, you are required to define and use three functions (get_number(), get_balance(), and get_name()) but you may define and use additional functions, if you wish.

Iutput File (customers_old.txt)/ It is provided on eLearning.

```
100000   43736.57 Rossum, Guido V.
100789 5681745.99 Eich, Brendan
320056       5.01 Ritchi, Dennis MacAlistair.
650430    2398.12 Wall, Larry
999999
```

Sample Output/Processing Input/Output Files (use EXACT format – and NO HARDCODING of the data itself)

First Sample Output

```
Enter a file prefix: custooomers
The file name custooomers_old.txt does not exist
```

Second Sample Output

```
Enter a file prefix: customers
Verifying input: 100000 43736.57 Rossum, Guido V.
Enter a command (a,c,d,w): w
Enter withdrawal amount: 1000.00
Enter a command (a,c,d,w): K
Entered an invalid command K.
Enter a command (a,c,d,w): w
```

```
Enter withdrawal amount: 500.50
Enter a command (a,c,d,w): d
Enter deposit amount: 200.00
Enter a command (a,c,d,w): a
New balance: 100000 42436.07 Rossum, Guido V.
Verifying input: 100789 5681745.99 Eich, Brendan
Enter a command (a,c,d,w): a
New balance: 100789 5681745.99 Eich, Brendan
Verifying input: 320056 5.01 Ritchi, Dennis MacAlistair.
Enter a command (a,c,d,w): w
Enter withdrawal amount: 5.00
Enter a command (a,c,d,w): c
Account not closed because money is still in it.
Enter a command (a,c,d,w): w
Enter withdrawal amount: 0.01
Enter a command (a,c,d,w): c
Account is closed
Enter a command (a,c,d,w): a
New balance: 320056 0.00 Ritchi, Dennis MacAlistair.
Verifying input: 650430 2398.12 Wall, Larry
Enter a command (a,c,d,w): a
New balance: 650430 2398.12 Wall, Larry
```

Output File (customers_new.txt)

```
100000   42436.07 Rossum, Guido V.
100789 5681745.99 Eich, Brendan
650430    2398.12 Wall, Larry
999999
```