CSE 231 **Fall 2013**

Computer Project #4

Assignment Overview

This assignment focuses on the design, implementation and testing of a Python program which uses file processing and exception handling to solve the problem described below.

It is worth 30 points (3% of course grade) and must be completed no later than 11:59 PM on Monday, October 7.

Assignment Specifications

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.

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, if the prefix is "customers", then 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)

7. The program will include a separate function which compares two floating point values for equality:

- **def equal_floats(x_flt, y_flt):** given two floats, compare for equality and return a Boolean

By definition, two floats are equal if the absolute value of their difference is less than 1.0e-8.

Assignment Deliverables

The deliverable for this assignment is the following file:

proj04.py – the source code for your Python program

Be sure to use the specified file name ("proj04.py") and to submit it for grading via the **handin** system before the project deadline.

Assignment 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 MacGregor, Charlotte D.
100789 5681745.99 Williamson, Bryan
320056       5.01 Canady, George Lucas
650430    2398.12 Fremont, Lorraine Elizabeth
999999
```

Note that the master file is in ascending order, although that isn't necessary for this project.

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 MacGregor, Charlotte D.
100789 5681745.99 Williamson, Bryan
650430    2398.12 Fremont, Lorraine Elizabeth
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 four functions (get_number(), get_balance(), get_name(), and equal_floats()), but you may define and use additional functions, if you wish.