420-LCU-05 Programming in Python

Assignment 4 – Final Project

Due May 05th, 2022

1- **Identification section**: This section must be either in a comment, with a '#' preceding each line, or enclosed within triple quotes ('''). The grader and I need this section for the accurate processing of your assignment. Assignments missing this may lose up to 5% of the mark.

```
#Your Name and ID
#420-LCU Computer Programming, Section #
#S. Hilal, instructor
#Assignment 4
```

2- **Submission**: Submit your assignment in one .zip file , which will include your python file and AccountsLog.txt file. Be sure to respect other instructions specified in the assignment. An important part of an assignment is to correctly follow the instructions as closely as possible.

3- **Late assignments**. Will be accepted if reasonable progress is demonstrated.

4- **This assignment should be short! You have all the pieces to make it work.**

Learning Objectives:

- Define and use a simple class, class & instance attributes, and methods.
- Manage a dictionary of objects and use formatted print to print info to a file or screen
- Use standard module **datetime** to get date and time information.

Description: Creating and managing different bank accounts for different clients at different banks. Your program will define a class **BankAccount** to create (instantiate) the different bank accounts and manage transactions as described below.

Class BankAccount:

A bank account has these attributes but ***you can include*** others if you need. ***Justify***.

1- **code**: A 6-digit unique account code that has to be stored as **int**

2- **name**: Account holder's name. A character string (8-15 characters)

3- **bank**: A character string (2-10 characters)

4- **acc_type**: A character string; Valid types: **chequing, saving, invest, loan, TFSA, RRSP**

5- **balance:** When creating an account***, default initial balance is zero*** but user can start with an amount (**int**) 0<=balance<=10000

6- **password:** ***private*** attribute. Reuse code & logic from Lab4

7- **Last_access:** ***private*** attribute. Date and time of last transaction on the account. ***Updated automatically*** by the class each time a transaction takes place. (Note: creating an account is a transaction)

The Account class should implement the necessary methods to create, delete, and display the information (print) of an account (similar to Students class). In addition, define methods: update_pwd, verify_pwd (make sure that user enters the correct password before carrying out a transaction), withdraw, deposit, transfer (between accounts) and get_account_balance.

The file accounts.txt contains the initial information for 5 bank accounts. The user can create other accounts as needed. Similar to A3, program starts by reading the file and creating the accounts. Each bank account is automatically created with a default password (**of your choosing**) that can be changed by the user after account creation. The account password is required for completing any type of transaction on any account.

Application Menu: Welcome to the Bank Accounts Management App

1- Print All Accounts (tabular format) (prints code, name, bank, acc_type, balance, last_access)
2- Admin prints passwords and last_access of all accounts. (prints code, password, last_access)
3- Create an account (Enter code, client name, bank name, account type and balance)
4- Withdraw an amount from an account. (account code & amount)
5- Deposit an amount to an account (Enter account code & amount)
6- Transfer an amount between accounts (Enter from and to account codes and amount)
7- Get balance of a given account (Enter account code)
8- Update Password (request old , verify and then ask for new password)
9- Display the log file.
10- Exit

Enter your requested option:

Each of the options 3-6 requires multiple user inputs (i.e. amount, name of account). You can decide to either get all the input together as usual or issue multiple input requests for each value separately. **Each option will require a max of 8 lines of code**.

Transaction Output: Each option displays a message confirming the transaction and **updates log file**.

Managing All Accounts: To be able to manage all accounts, we need to create a ***dictionary of accounts***: ***key is the account ID*** and ***value is the BankAccount Object***.

```
#Main Program Code
#the class does not know anything about the dictionary. It is managed in main code outside of the class.
MyAccounts = {}  # Dictionary of Accounts
#Read and create accounts from accounts.txt. Below is Sample code to create an account in option 3.
account_info = input("enter account info:") # user enters 122456,John Smith,TD,saving,200
acc = account_info.split(",")
account_code = int(acc[0])
MyAccounts[account_code]=BankAcount(int(acc[0]),acc[1],acc[2],acc[3],int(acc[4])  # key=ID , value= object
```

Log File: Your program will create and update a log file (**AccountsLog.txt**). The log file will have the date, time and description of ***every transaction*** on all accounts. **Use mode="a" (append) to open the file so all transactions can be logged and kept**. Each entry line in the log file is of the format shown below. Make sure that your code will align the log info in the file. (Use formatted printing).

Date & time account code Transaction Type Amount Balance

Note: If a transaction involves 2 accounts (e.g. transfer), create a log file entry for each account.

Transaction Types: created, withdraw, deposit, get balance