# BIOS6642 Introduction to Python Programming
# Class Project
# Due: May 15, 11:59PM

The task of this class project is to write a program to manage customer accounts for a bank. A real bank account management system is complicated and here we implement a simplified version. In this simple, small bank account management system, each account has the following attributes:

– Account number. This is a unique integer between 100 and 9999.
– Account owner's name. This is a string, not necessarily unique.
– Account owner's age. This is a positive integer.
– Account owner's residency (in which state the owner lives). This is a string, such as Colorado, Utah, California, and so on.
– Current balance. This is a non-negative integer.
– The date when the account is opened. This is a string in the format of *month*/*day*/*year*.

Please download the file *account_data.csv* from Canvas. This file contains account information, such as account number, owner's name, age, residency, current balance and the date of account opening. This file should be used to initialize the bank account database when the system starts running. This bank account management system must provide the following operations for the user:

– User login. This requires the user to provide a user name and a password during runtime. Here assume all of us use the same user name (i.e., *Admin*) and password (i.e., *python*). The system will successfully start running only if correct user name and password are provided by the user. The system allows the user to re-enter user name and password, if they are not correct. However, if the user provides 3 successive incorrect inputs for user name and password, the system (i.e., your program) automatically stops running, and you have to re-start the system to enter user name and password. After successful user login, the system should automatically initialize the customer account database by using the file *account_data.csv*, and then the system will ask the user to provide inputs for account management (see below).

– Display all the accounts (i.e., the account attributes) on the screen. The system must provide options of how to display the account information, which is sorted by (1) account number, (2) account owner's name, (3) account owner's age, (4) account owner's residency, (5) current balance, or (6) the date when the account is opened. Your program must implement the display options of (3), (5) and (6). You are not required to implement the options of (1), (2) and (4), but you are welcome to implement all of them.

– Print account statistics on the screen. This operation displays (1) total balance, average balance and median balance of all accounts, and (2) total balance, average balance and median balance of accounts for each following age group: (a) under 35, (b) 35 - 44, (c) 45 - 54, (d) 55 - 64, (e) 65 - 74, and (f) 75 and older.

– Display a subset of accounts on the screen, given certain criteria. Your program needs to implement the following options: (1) display the accounts with the owners having ages between *min_age* and *max_age*, which are positive integers given by the user during runtime; (2) display the accounts with the owners having a residency, *residency*, which is a string provided by the user during runtime; (3) display the accounts with the owners having current balances between *min_balance* and *max_balance*, which are positive integers given by the user during runtime; (4) display the accounts with opening dates between *min_date* and *max_date*, which are strings provided by the user during runtime.

– Query and display an account given (1) an account number or (2) an account owner's name. If the given account number or owner's name does not exist in the current account database, the system should print a proper message and ask the user to re-enter another account number or owner's name. If multiple accounts are found given an account owner's name (note that the owner's name is not unique), please display them all on the screen.

– Open a new account. The account owner's name, age, residency and current balance are provided by the user during runtime. The account number is randomly generated between 100 and 9999. Please note that each account number is unique and it must be different from the others in the current account database. The date of opening the account is the day when the system runs (i.e., current date). If the account is successfully created, the system automatically updates the account database, i.e., add this newly opened account to the current database, and also updates the file *account_data.csv* on your disk.

– Delete an account given an account number, which is provided by the user during runtime. If the given account number exists in the current account database, the system deletes the corresponding account from the database and also from the file *account_data.csv* on your disk. If the given account number is not found in the current database, the system should print a proper message and ask the user to re-enter another account number.

– Conduct transactions for a specific account. In this operation, the system provides the following options to perform transactions for a given account, which is provided by the user during runtime:

  • Deposit money into the given account. The amount of money is provided by the user during runtime. If the deposit is successfully performed, the system automatically updates the account database and also the file *account_data.csv* on your disk.

  • Withdraw money from the given account. The amount of money is provided by the user during runtime. In order to get a successful withdrawal, the current balance (before the withdrawal) of the account must be higher than the amount that the user intends to withdraw. If the withdrawal is successfully performed, the system automatically updates the account database and also the file *account_data.csv* on your disk.

- Transfer money from the given account (sender) to another account (receiver). The receiver account and the amount of money to transfer are provided by the user during runtime. In order to get a successful transfer, the current balance (before the transfer) of the sender account must be higher than the amount that the user intends to transfer. If the transfer is successfully performed, the system automatically updates the account database and also the file *account_data.csv* on your disk (note that both sender and receiver accounts will be updated).

- Display current balance for the given account on the screen.

- Account settings such as changing owner's name, age and/or residency for the given account. If this change is successfully performed, the system automatically updates the account database and also the file *account_data.csv* on your disk. The account number, current balance and the date of account opening cannot be changed manually.

- Display recent transactions for the given account on the screen. Here the recent transactions are the last few (e.g, 5) transactions of money deposit, withdrawal, and transfer as well as the activities in account settings, such as changing owner's name, age and/or residency. Please note that these transactions include the events/activities of money deposit, withdrawal, transfer, and owner's name/age/residency changes that occurred during the last running of the bank account management system. This means that these transactions should be recorded in a file for persistent storage such that they would not be deleted when you quit the system. When you rerun the bank account management system and want to display the previous transactions for a specific account on the screen, you would need to read the file to get the information of the events/activities.

– Provide an option to quit the system.

For each operation above, please handle potential exceptions properly if necessary. For example, if an operation asks the user to enter a number 2 but receives an English word *two*, the system should be able to handle a potential exception of $ValueError$.

Please test each operation above and take a screenshot of the output, and write a report to clearly show the output of each operation. Please submit your source codes and the report to Canvas. The source codes should be properly documented such that they are readable.