Assignment 1 (14%)

Due: 31st October 2019 at 23:59

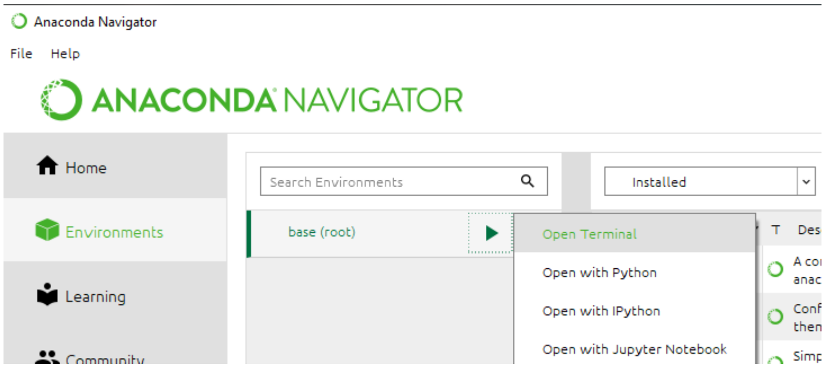
Objective

Students are asked to develop an ATM Python application for a local bank called PyBank. The application should provide login, cash withdrawal, transfers, balance checking and currency exchange functions.

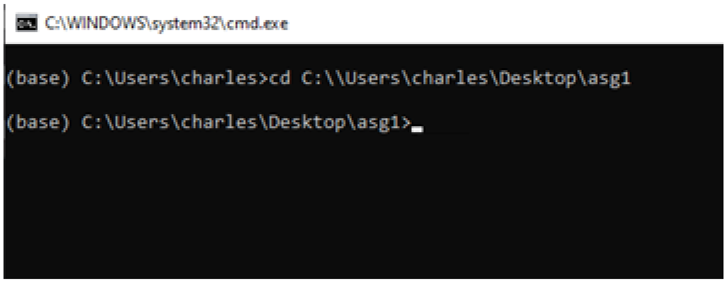
Setup:

Your program will be executing in command prompt. You can use Spyder to write your program and save the program in a location, e.g. C:\\Users\charles\Desktop\asg1\atm.py. Then you can use the anaconda prompt to test the program.

1. Launch Anaconda Navigator
2. Choose Environments, base(root) ▶ □, Open Terminal



3. Type: cd <location>
   (both for Mac and Windows)

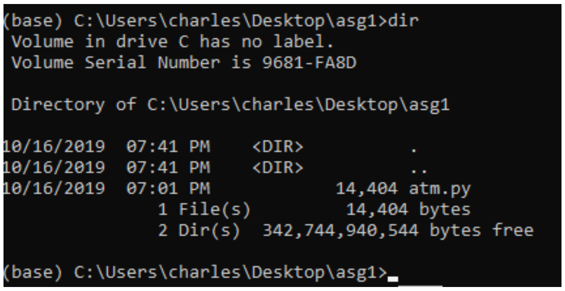


C:\WINDOWS\system32\cmd.exe

```
(base) C:\Users\charles>cd C:\\Users\charles\Desktop\asg1

(base) C:\Users\charles\Desktop\asg1>_
```

4. Type: dir (for Windows) , ls (for Mac)



```
(base) C:\Users\charles\Desktop\asg1>dir
 Volume in drive C has no label.
 Volume Serial Number is 9681-FA8D

 Directory of C:\Users\charles\Desktop\asg1

10/16/2019  07:41 PM    <DIR>          .
10/16/2019  07:41 PM    <DIR>          ..
10/16/2019  07:01 PM            14,404 atm.py
               1 File(s)         14,404 bytes
               2 Dir(s)  342,744,940,544 bytes free

(base) C:\Users\charles\Desktop\asg1>_
```

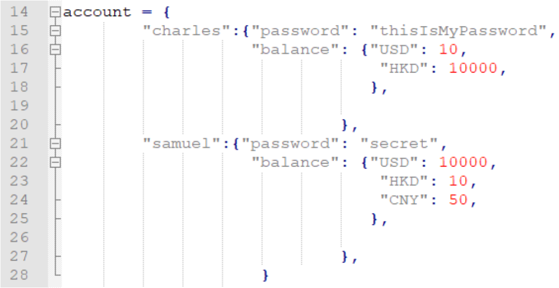
5. Run the program: python atm.py

Information:

Here is the given data:

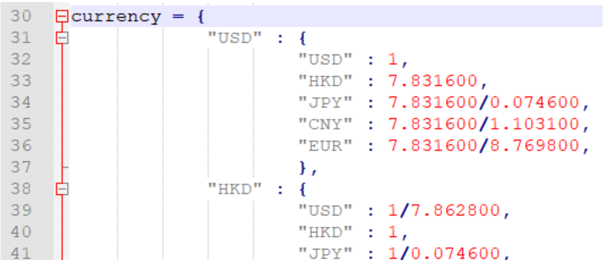
"***login_tried***" is a variable to store the number of times that a user fails to login.
"***endProgram***" is a variable to store a Boolean value that indicates whether the program should exit the main menu loop.
"***login***" is a variable to store the login name the user entered.

"***account***" in line 14 is a dictionary object to store the username, password, and the account balance of the users. You can add more accounts or modify the amount of money for program test.



```
14  account = {
15          "charles":{"password": "thisIsMyPassword",
16                     "balance": {"USD": 10,
17                                 "HKD": 10000,
18                                },
19  
20                    },
21          "samuel":{"password": "secret",
22                     "balance": {"USD": 10000,
23                                 "HKD": 10,
24                                 "CNY": 50,
25                                },
26  
27                    },
28            }
```

"***currency***" in line 30 is a dictionary object to store the fixed exchange rate of different currencies. For example, if someone wants to exchange USD$ 10 to HKD, the amount of HKD he/she will get is:
10 * currency["USD"]["HKD"], that is HKD 78.316.



```
30  currency = {
31              "USD" : {
32                      "USD" : 1,
33                      "HKD" : 7.831600,
34                      "JPY" : 7.831600/0.074600,
35                      "CNY" : 7.831600/1.103100,
36                      "EUR" : 7.831600/8.769800,
37                      },
38              "HKD" : {
39                      "USD" : 1/7.862800,
40                      "HKD" : 1,
41                      "JPY" : 1/0.074600,
```

Requirements

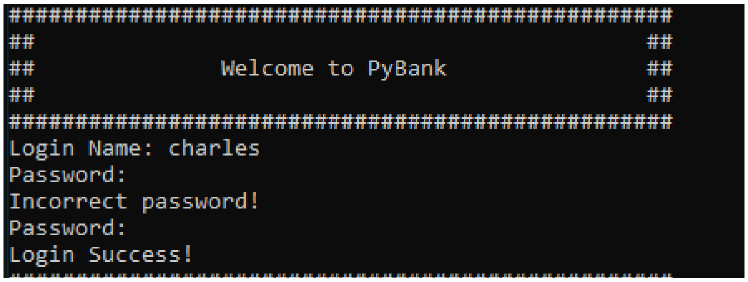
Task 1 login validation

In this task, you are required to verify a user's identity:

- Check the existence of a username before asking for password
- Max number of attempts: 3
- The password entered by the user should not be shown in the console

If the user passes the login process, the variable "***login***" should store the username, e.g. charles.
If the user fails to pass the login process, set the value of "***endProgram***" to True.

Sample screenshot:

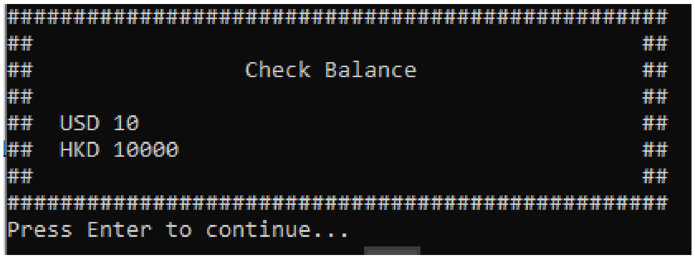


```
##################################################
##                                              ##
##              Welcome to PyBank               ##
##                                              ##
##################################################
Login Name: charles
Password:
Incorrect password!
Password:
Login Success!
```

Task 2 Balance Checking

In this task, you are required to complete the function "check_balance()" that shows the balance of the user account. At the end of the function, it needs to set the return value to **False**. The return value is then assigned to the variable "endProgram".

Sample screenshot:

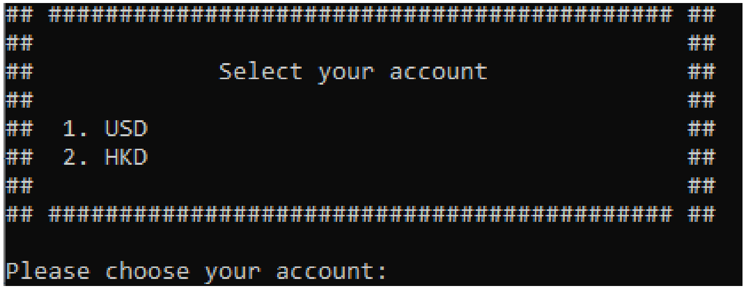


```
##################################################
##                                              ##
##                Check Balance                 ##
##                                              ##
##   USD 10                                     ##
##   HKD 10000                                  ##
##                                              ##
##################################################
Press Enter to continue...
```

Task 3: Cash Withdrawal

In this task, you are required to complete the function “cash_withdrawal()”.

1. The function shows the following menu to the user and asks the user to choose an account.



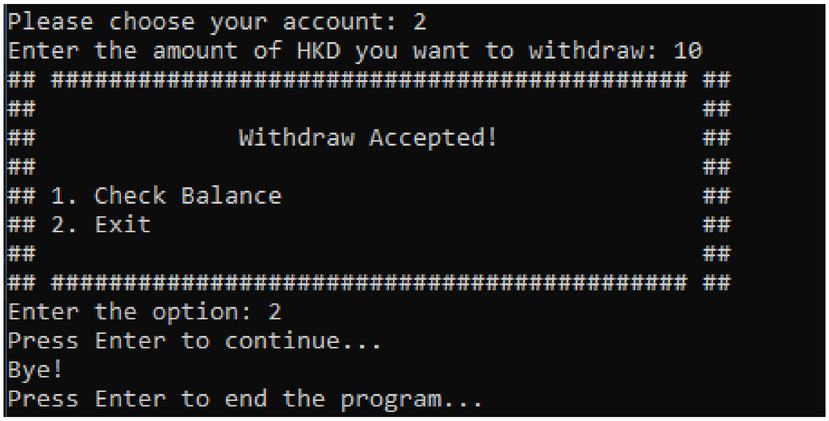
```
## ############################################ ##
##                                              ##
##             Select your account              ##
##                                              ##
##  1. USD                                      ##
##  2. HKD                                      ##
##                                              ##
## ############################################ ##

Please choose your account:
```

2. Ask the user for the amount of money he/she would like to withdraw.
3. The function will then process the request.
    a. The maximum amount of each withdraw transaction is limited to **$50000**.
    b. The function then checks whether the user has enough amount of money.
    c. The function accepts integer numbers only.
4. If the user can successfully withdraw the money, the function will ask whether they want to
   (1) check the balance; or
   (2) exit the ATM program.
    a. If the user chooses to check the balance, a corresponding function will be invoked.
    b. Otherwise, set the return value of the function to **True**.

p.s. For any other situation, set the return value to **False** for continuing the loop of the application.

Sample screenshot:



```
Please choose your account: 2
Enter the amount of HKD you want to withdraw: 10
## ############################################ ##
##                                              ##
##              Withdraw Accepted!              ##
##                                              ##
## 1. Check Balance                             ##
## 2. Exit                                      ##
##                                              ##
## ############################################ ##
Enter the option: 2
Press Enter to continue...
Bye!
Press Enter to end the program...
```

Task 4: Transfer

In this task, you are required to complete the function “transfer()”.

1. The function shows a screen menu for a user to choose an account for transfer money.
2. The function asks the user for a receiver’s account.
    a. If the receiver does not have that currency account, the transfer process will be terminated.
3. The function then asks the user for the amount of money he/she would like to transfer.
4. The function will then process the request.
    a. The maximum amount of each withdraw transaction is limited to **10000**.
    b. The function will check whether the user has enough amount of money.
    c. The function accepts integer numbers only.
5. If the user can successfully transfer the money, the function will ask whether the user wants to
   (1) check the balance; or
   (2) exit the ATM program.
    a. If the user chooses to check the balance, a corresponding function will be invoked.
    b. Otherwise, set the return value of the function to **True**

p.s. For any other situation, set the return value to **False** for continuing the loop of the application.

Task 5: Currency Exchange

In this task, you are required to complete the function “currency_exchange()”.

1. The function shows a screen menu to a user and the user may choose an account for currency exchange. (**FROM_currency**)
2. The function then asks the user for the currency he/she wants to get (**TO_currency**)
   e.g. **FROM_currency = USD**, **TO_currency = HKD** means the user wants to exchange USD$10 to HKD 78.316
3. The function asks the user for the amount of TO_currency he/she would like to get. (In the above example, the user should have typed 78, for HKD 78)
4. The function will then process the request.
    a. There is no exchange limit.
    b. The function will check whether the user has enough amount of money or not.
    c. The function accepts integer numbers only.
    d. If the user does not have a TO_currency account, a new account will be created for the user.
5. If the user can successfully exchange the money, the function will ask whether he/she wants to
   (1) check the balance; or
   (2) exit the ATM program.
    a. If the user chooses to check the balance, a corresponding function will be invoked.
    b. Otherwise, set the return value of the function to **True**

p.s. For any other situation, set the return value to **False** for continuing the loop of the application.