**UMGC Data 620 Assignment 8.3**

**Your name:**

**Date:**

# Each student runs the Python development environment Jupyter Notebook. You will complete code as outlined below, and then check your answers online with a homework quiz.

*Part 1: Business Context*

You are the owner of a growing credit card company. As one of your first tasks, you are automating the process of managing customer transactions with Python (i.e. purchases, refunds, and inquiries). Currently, your company stores your data as two separate .csv files (this stands for “comma separated value” files, and is typically what your standard Microsoft Excel file can be saved as). One file contains all the customer data, including their customer ID, first name, last name, current credit card balance, and credit limit. The other file is a list of transactions, which includes the transaction ID, relevant customer ID, type of transaction, and requested balance change. The balance does not change if the type of transaction is an ‘inquiry’.

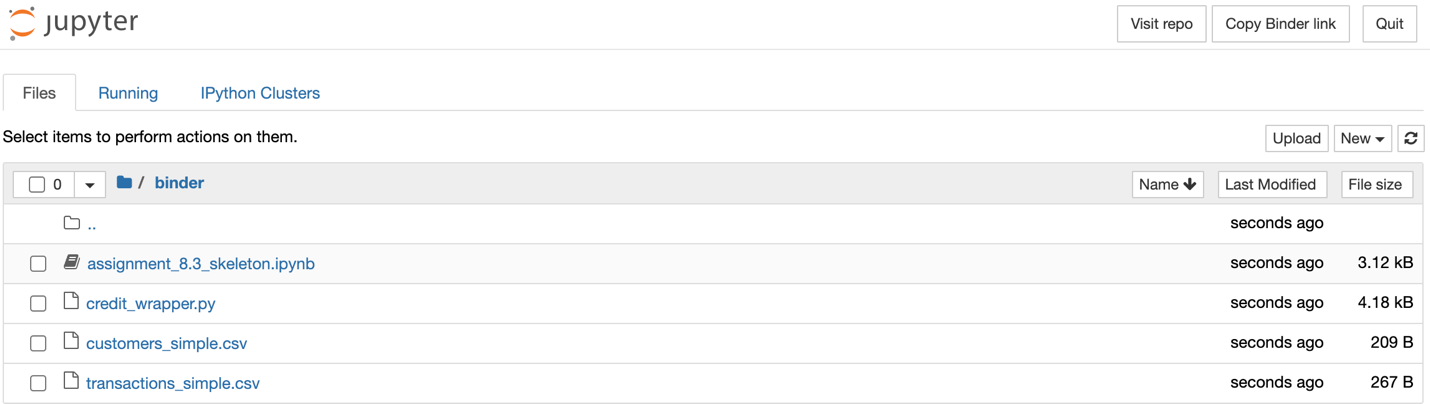
Luckily, a colleague has written part of the code for you! Currently the program will take in the two .csv files and load the data into memory (you don’t need to worry about this part). All that is left is to fill in a few functions that they couldn’t get to finish. We will explain this task below.

*Part 2: Setup*

For this assignment, we have a few files that need to be uploaded:

* assignment\_8.3\_skeleton.ipynb
* credit\_wrapper.py
* customers\_simple.csv
* transactions\_simple.csv

Like in previous assignments, we upload all the above files to a Try Jupyter notebook. Your repository should look like this:



Click on the skeleton. Then you can run the cells by opening the “Run” menu and clicking “Run All”:

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generatedWhen you run the cells, you should end up getting an error. Don’t worry! This is because the three functions new\_purchase, refund, and inquiry all return None currently. When you finish implementing these functions, the program should end up working fine.

Our wrapper works by taking in a list of customers and transactions (which are loaded in the second cell of the Jupyter Notebook). It first creates an empty .txt file called ‘log.txt’ (if this file already exists, it overwrites it). For each transaction, it uses the customer ID to find the corresponding customer data. It then checks the transaction type: if it is an ‘Inquiry’ it calls inquiry(), if it is a ‘Purchase’ it calls new\_purchase(), and if it is ‘Refund’ it calls refund(). Each of these functions are called with the appropriate arguments, pulled from the transaction and customer data loaded at the start. For each function call it writes the printed message to the log.txt file. The program stops running once there are no more transactions.

Diagram

Description automatically generated

For grading purposes each action performed by the code changes a hash code that is generated at the start of the program’s runtime. This is called a checksum. At the end, the program will return a 4-digit number which we will use to determine if everything ran properly or not. You do not need to worry about how this works.

*Part 3: An Aside on Function Returns*

For this assignment, you will need to know a few things about declaring Python functions. Generally, you want your functions to give a value for an output; this is done using a ‘return’ statement. For example, a simple Python function could look like the following:

Graphical user interface, text, application

Description automatically generated

This function takes in an input x and checks if it is the number 5 or not. If it is 5, it returns ‘True’ (this is called a Boolean value), and if not it returns ‘False’. Here we see that our function has two different return calls for each case. This is useful for when we don’t know what our function should be returning, or if we don’t want our function to continue running after we’ve found our desired output.

Graphical user interface, text

Description automatically generatedNote that here, our function returns only one value. However, it is possible to return multiple things at once! Let’s say we want our above function to also return the value x. Then we could write something like this:

It returns the outputs as a pair of outputs. This is especially useful to create *flags* in your programs. What we mean by this is, if we want to return a value but also a Boolean value indicating some property about that value is true. So in this example case, we want to flag whether or not the input is five and return the value alongside the flag.

This will come especially handy when writing the new\_purchase function, since we ask you to flag whether or not the purchase is allowed, while also returning the appropriate new balance.

You can get a little more information about functions with multiple return values at this link: <https://pythonbasics.org/multiple-return/>

*Part 4: Your Task*

Now it’s time for the actual assignment! In the .ipynb file, we have outlined three functions for you to fill in: inquiry(), new\_purchase(), and refund(). Each function is outlined below.

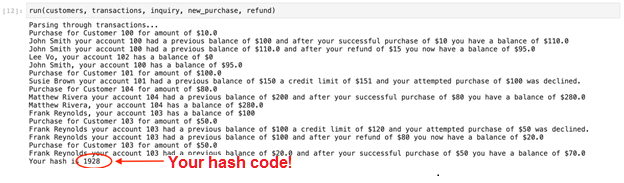
* inquiry
  + Takes in the customer ID, first name, last name, and balance as arguments (note that first name and last name will be strings, while balance and customer ID will be integer values
  + Returns a string message of the form “[First name] [Last name], your account [Customer ID] has a balance of $[balance]”.
* new\_purchase
  + Takes in the customer’s current balance, credit limit, and transaction amount (how much they spent)
  + new\_purchase needs to check whether to allow the purchase and return the new balance (and a Boolean flag saying the transaction is allowed) or whether the transaction puts the balance over the credit limit (which would then also return a Boolean flag)
    - Specifically, we expect an output of the form ‘return True, new\_balance’ if the purchase is allowed, and ‘return False, current\_balance’ if the transaction is not allowed
* refund
  + Takes in the customer’s balance and transaction amount
  + Returns the updated balance by subtracting the refund amount from the balance
    - It is okay if the balance is a negative number

When you execute the files, the wrapper will write an output for each transaction to the text file and print the changes that your functions make to each customer’s balance. The cell before the cell with your functions runs a check, so make sure each customer ID in the transactions file matches the customer IDs in the customers file. The output if everything works should be:

A picture containing background pattern

Description automatically generated

The percentages might differ based on the size of the transactions. When you run the program using the customers\_simple.csv and transactions\_simple.csv you should get a similar output to the following:



The number you see at the bottom of this output is the hash code of the program, or the code you will submit to show your program ran successfully.

Graphical user interface, text, application, email

Description automatically generatedYou will be running this program for different customer/transaction files. In order to change this, you need to change the file name in quotations in the cell above the code you wrote your functions in. These are circled below. For example, if you wanted to run the program on a file called ‘new\_customers.csv’ and ‘new\_transactions.csv’ , you would replace the line customer\_file = ‘customers\_simple.csv’ with customer\_file = ‘new\_customers.csv’.

Graphical user interface, text, application, email

Description automatically generatedLastly, we want to look at the ‘log.txt’ file that the wrapper creates. You can just open them by clicking the file in the same place you uploaded the files. You can get to this repository by clicking the ‘File’ button in the toolbar, and going to “Open”.

The file is created in the same folder that you run the .ipynb and .py file in. You should then see the text file show up when you go back to this folder then.A picture containing application

Description automatically generated