CSCI 140 Programming for Data Science Project 1

Due Wednesday, 21 September 2022 at 1700 hrs

In our first project, your task is to create 2 short programs, complete one program, and correct a fourth program. There are four programs to submit, each performing one task. They can all be completed using only the material introduced in the first couple of days of class. You will submit these as separate files with the extension .py, but you can write and test your programs in a jupyter notebook if you prefer. In the lab, you will practice creating, saving, and running **.py** files. We suggest that for each program you write an algorithm as a simple numbered list of steps and then create a Python implementation. Your submission will consist of a PDF document containing your reflections on the project, and four Python programs saved in **.py** files named as specified.

Background

What is an exchange rate? It is the amount required to exchange one currency for another. The exchange rate keeps fluctuating as currencies are actively traded (one currency is exchanged for another) 24/7. Let's say we use U.S. dollars (USD) to buy Canadian dollar (CAD), the exchange rate is for the CAD/USD pair. Here, the first currency listed (CAD) always stands for one unit of that currency and the exchange rate is essentially the amount of the second currency (USD) required to buy one unit of the first currency (CAD). For example if the CAD/USD pair is 0.7518, it means that it costs 0.7518 USD to buy 1 Canadian dollar.

Program One: Converting USD to GBP

Write a program that solicits one input from the user: Total amount in USD GBP/USD exchange rate. Your program should calculate the final amount in GBP (Great Britain Pound Sterling) based on a fixed GBP/USD exchange rate, which for our purposes is going to be 2.13. **For testing purposes, the calculated amount must be rounded to the nearest integer and should not print any decimal component.**

You must format the program's interaction as illustrated in the following examples. Do not vary the phrasing, capitalization, spacing, etc. You must also write a properly styled, easily readable and understandable program.

```
Enter the total amount in USD: $1500
The total amount in GBP is about 704 pounds.
```

Your program should work for any non-negative numeric input provided by the user. Other inputs will not be tested. Save this program in a file called **USD_to_GBP.py**

Program Two: Calculating Percentage Markup

When we convert currency via a bank/currency exchange house, the bank/currency exchange house earns a profit by marking up the exchange rate, as will any payment service provider or credit cards.

For example, if the CAD/USD market exchange rate is 1.33 i.e. it costs 1.33 U.S. dollars to buy 1 Canadian dollar and the bank exchange rate is 1.37. The percentage markup (profit) charged by bank with respect to the market exchange rate is 3%.

Write a program that calculates the percentage markup. Your program will solicit two inputs from the user: the market exchange rate and the bank exchange rate. **For testing purposes the final amount must be rounded to the nearest integer.** Save this program in a file called **profit.py**.

Program execution should look like what is shown below. Pay special attention to formatting, spacing, and capitalization of the input and output lines.

```
Enter the CAD/USD Market exchange rate: 1.33
Enter the CAD/USD Bank exchange rate: 1.37
Percentage markup is 3%.
```

Program Three: Currency Conversion

In Program 1, you converted currency from USD to GBP. We now want to consider exchanging one foreign currency for another, using the exchange rate information with respect to USD. The table below shows different foreign currency exchange rates with respect to USD.

Currency	**Foreign currency/USD** **Exchange Rate**
Australian Dollar (AUD)	AUD/USD: 0.7278
South Africa Rand (ZAR)	ZAR/USD: 0.0669
Euro (EUR)	EUR/USD: 1.1765
Indian Rupee (INR)	INR/USD: 0.0135

The a program will take three inputs: an integer for selecting home currency, an integer for selecting destination currency, and an amount in home currency. **You have been given two lines of code that you cannot change in any way and which must be used for your program in the file con_cur.py.**

The program should provide the output in the requested format. Program execution should look like what is shown below. Pay special attention to formatting, spacing, and capitalization of the input and output lines.

Select your home currency: 1 for AUD, 2 for ZAR, 3 for EUR, 4 for INR: 1
Select your destination currency: 1 for AUD, 2 for ZAR, 3 for EUR, 4 for INR: 4
Enter total amount in home currency: 400AUD
The total amount in INR is about 21564.

You can assume that the input for amount will always be positive numbers, but they may have fractional components and will always be followed by the abbreviation for the home currency (ex: 400**AUD**). You can also assume that the inputs for selecting a currency will always be one of the integers 1,2,3, or 4. **The final value computed should be rounded to the nearest integer**. No other inputs will be tested.

N.B.: you must only use programming structures discussed so far – this means no conditionals and no imported mathematical functions/modules and no object types that we have not discussed yet. You don't need them here.

Program Four: Number of Transactions

When we use our credit cards internationally, we are charged something known as foreign transaction fee. This fee is a fixed percentage of the amount you swipe the card for and is charged every time you swipe the card. It is deducted from the total credit limit amount.

Let's say you are travelling internationally and have a fixed credit limit on your credit card that you **cannot** exceed. For our purposes the foreign transaction fee will **always** be 5% and the amount that you can spend each time you swipe is **fixed**. The total number transaction that can happen with the credit card depends on how much money remains until you hit your credit limit. This is best shown by a few examples.

Remember: for this program we are NOT converting currency. We are only dealing in USD. All the entries will be in format $123

Example 1:

TRANSACTION FEE = 5%
Credit limit = $100
Money spent per transaction = $30
Total number of transactions = 3

Example 2:

TRANSACTION FEE = 5%
Credit limit = $762.83
Money spent per transaction = $52.93
Total number of transactions = 13

Example 3:

TRANSACTION FEE = 5%
Credit limit = $5092.45
Money spent per transaction = $12.34
Total number of transactions = 393

In the file **travel.py**, you have been given a code for a program that provides constant for the foreign transaction fee, and solicits the credit limit and the money spent each time the card is swiped from the user. The program is supposed to calculate the total number of transactions, but the program is incorrect.

Your job is to debug the program. Debugging the code means that you edit the lines in place. You should not add lines, delete lines, or move the order of lines. Correct the lines as they are and preserve as much of the original code as possible.

HINT: figure out what each line is trying to accomplish first before making any changes.

Here are some examples of program execution. Pay special attention to the formats of the inputs and outputs – your printed output should match exactly in terms of spacing and text. **The final value should be an integer**, i.e. cast to an int.

Program execution for Example 1:

Enter the credit limit amount: $100
Enter money spent per transaction: $30
The total number of transactions possible are 3.

Program execution for Example 2:

Enter the credit limit amount: $762.83
Enter money spent per transaction: $52.93
The total number of transactions possible are 13.

Program execution for Example 3:

Enter the credit limit amount: $5092.45
Enter money spent per transaction: $12.34
The total number of transactions possible are 393.

Your program should work for any inputs in the format shown – inputs representing numeric values will always be positive numbers

N.B.: you must only use programming structures discussed so far – this means no conditionals and no imported mathematical functions/modules and no object types that we have not discussed yet. You don't need them.

SUBMISSION EXPECTATIONS

- **Project1.pdf** : A PDF document containing your reflections on the project, and information on any extra credit opportunities you pursued. **You must also cite any sources you use. Please be aware that you can consult sources, but all code written must be your own. Programs copied in part or wholesale from the web or other sources will receive 0 points and will result in reporting of an Honor Code violation.**
- **USD_to_GBP.py** Your implementation of program one, following the format specified above.
- **profit.py** Your implementation of program two, following the format specified above.
- **con_cur.py** Your completed implementation of program three.
- **travel.py** Your corrected implementation of program four.

Please be aware that you will receive no feedback if your files are incorrectly named or are in an incorrect format.

POINT VALUES AND GRADING RUBRIC

Program 1: 15 points
Program 2: 21 points
Program 3: 25 points
Program 4: 25 points
Write-up: 2.5 points
Auto-grader: 11.5 points

The general grading rubric for projects is below. Some of the specifics given in the rubric are not applicable to this project (e.g. import lines and copious commenting).

Grade Level	Execution	Correctness and Algorithms	Formatting	Style and References
A	Code executes without errors, requires no manual intervention	Code correctly implements all algorithms, passes all test cases (with possible exception of extremely rare case as applicable), and meets all required functionality	Inputs and outputs formatted as per the specification both in terms of content and type, may be minimal formatting issues (spacing, punctuation, etc.); correct file formats and names	Code uses current structures and modules discussed in class; meaningful variable names; commented as appropriate; any references cited in write-up
B	Code executes without errors, may require minimal manual intervention such as change of an import line	Meets all required functionality with exception of minor/ rare test cases (as applicable)	Inputs and outputs formatted as per the specification both in terms of content and type, some minor formatting issues may be present (e.g. incorrect string text), correct file formats and names	Code uses current structures and modules discussed in class; meaningful variable names; commented as appropriate; any references cited in write-up
C	Code does not execute without minor manual intervention such as correcting indentation or terminating parentheses or a single syntax error	Code contains logical errors and fails subset of both rare and common test cases, overall algorithmic structure reflects required functionality but implementation does not meet standards	Inputs and outputs have major formatting issues, files incorrectly named but file formats correct	Code uses current structures and modules discussed in class; meaningful variable names; commented as appropriate; references cited in write-up
D	Code does not execute without major manual intervention such as changing variable names, correcting multiple syntax errors, algorithmic changes	Code contains major logical errors and fails majority of test cases, lack of cohesive algorithmic structure, minimal functionality	Inputs and outputs have major formatting issues, files incorrectly named but file formats correct	Code uses some structures and modules from class, but otherwise largely taken from outside sources with citations; variable names and code structure hard to decipher
F	Code requires extensive manual intervention to execute	Code fails all or nearly all test cases and has no functionality to solve the proposed problem	Inputs and outputs have major formatting issues, files in incorrect formats which cannot be graded	Code uses structures and modules from outside sources only with no citations; variable names and code structure hard to decipher; code written in other programming languages or Python 2.7

References:

1. https://www.investopedia.com/articles/forex/090314/how-calculate-exchange-rate.asp
2. https://www.easymarkets.com/int/en-gb/learn-centre/discover-trading/currency-acronyms-and-abbreviations/
3. https://www.corporateinformation.com/Currency-Exchange-Rates.aspx?c=840