Programming
Homework assignment 2
Personal expense manager

General information

Deadline: May 26th, 23:59:59
Submission: through the Canvas LMS as a zip archive

1 Problem description

Most of the financial organizations today provide online banking services for their clients. These applications often have integrated tools for managing personal expenses through either a web interface or a mobile application.
The interesting task, however, is to integrate such services of different banks so that you have a single place (application) to monitor the movement of your funds. Integration of information systems related to multiple banks can be a very complicated task, however there is a simplified way to solve the same problem. Most of the online banking services send text messages (SMSs) as confirmations to inform clients about all the operations related to their bank accounts (credit cards). In this assignment, you will implement a system that uses these messages to build a simple personal expense management application.

Description of tasks

As the source dataset you are provided with a list of SMSs received by your smartphone. Each SMS has a source phone number, from which it was received, and the text of the message. Your application has to process SMSs from 2 phone numbers:

1. 480 (SuperBank). The online service of this bank sends two types of SMSs:

   Withdrawal: card *6678: 10 EUR, balance: 578 EUR
   Transfer: card *6623: 100 EUR, balance: 870 EUR

   Text messages sent by Superbank start with either "Withdrawal" or "Transfer" depending on the type of operation then followed by the last 4 digits of the credit card, then by the operation delta (always positive, depending on the operation type the money is either added to the card (transfer) or removed from the card (withdrawal) and finally the current balance.

2. 720 (GorgeousBank). The online service of this bank sends the following types of messages:

   *1238: -10 EUR, left: 450 EUR
   *1253: +500 EUR, left: 700 EUR

   SMSs of the GorgeousBank start with * followed by 4 last digits of the client's credit card, then by the operation delta (can be either positive or negative depending on the type of operation) and then the current balance.

Basic part (6 points)

1. Start a new project
2. Create a text file that will contain the source dataset of SMS messages. Each message needs to have the following three attributes: phone number, date and time of message receival, text. The file can contain messages not only from the two target phone numbers, but also from other ones.
3. Read and process the dataset to extract a list of money transfer operations. You can only hardcode two phone numbers, which correspond to the two banks, however the remaining data (card numbers, amounts, etc.) have to be extracted from the file contents.
4. Implement the user interaction interface that will show the current funds of a client and calculate expenses in a given month and year (entered by the user) - check program flow below

Extra part No. 1 (2 points)

Check the openpyxl library for reading and writing XLSX files from Python programs. Use openpyxl to export a full monthly report. Unlike your program console UI, the Excel report should contain all transactions sorted in ascending order of date. At the bottom of the transaction list, add a total line (which should match in numbers with what your program prints in the console as a short summary of the monthly report). Make sure that the Excel report is nice to read, all columns have appropriate width, the total line is highlighted, a caption appears at the top.

Extra part No. 2 (2 points)

Adjust your program in such a way that new algorithms of processing SMS messages (i.e. from a new bank online service) can be added easily.
It is sometimes hard to say what exactly is easy and what is not, but to get the maximal 2 points, you are expected to produce a solution that will require adding/changing no more than 5 lines of code in the main program to make the new logic functional.
Hint: you need to extract specific logic of processing each bank's SMS to a separate module and then load the required modules dynamically. The main program should be left with the general logic only. Some of the functional programming principles may help you solve this task.

Hints

1. Dates can be represented as strings, for easy comparison many application store dates in the YYYY-MM-dd HH:mm:ss format, you can then extract month and year using slices on strings. You can also check the Python datetime type (needs to be imported)

General requirements

- Use functions and optionally modules to achieve a better program structure
- Protect against incorrect user input and file related errors
- Apply proper coding style, use meaningful names of variables and functions

Program flow example

You don't have to exactly replicate the program flow shown below, but the general logic should be kept.
You are also free to introduce additional complexity.
User input is shown in blue:

```
Choose an option
1 - Show current funds
2 - Expenses per month
3 - Exit from the program
Your choice: 1
Your current funds:
*4523 (SuperBank): 2540 EUR
*6687 (GorgeousBank): 1200 EUR
Total: 3740 EUR
Press any key
User presses a key
Choose an option
1 - Show current state
2 - Expenses per month
3 - Exit from the program
Your choice: 2
Enter month and year in the following format MM-YYYY:
09-2018
Select a credit card:
1 - *4523 (SuperBank)
2 - *6687 (GorgeousBank)
3 - Total
4 - Exit to the main menu
Your choice: 1
Report for September 2018, card *4523 (Superbank)
Received: 1354 EUR
Spent: 867 EUR
Delta: +487 EUR
Export a full report to Excel? y
Report created, opening ...
Select a credit card:
1 - *4523 (SuperBank)
2 - *6687 (GorgeousBank)
3 - Total
4 - Exit to the main menu
Your choice: 3
Report for September 2018, all credit cards
Received: 2098 EUR
Spent: 1400 EUR
Delta: +698 EUR
Export a full report to Excel? n
Select a credit card:
1 - *4523 (SuperBank)
2 - *6687 (GorgeousBank)
3 - Total
4 - Exit to the main menu
```

```
Your choice: 4
Choose an option
1 - Show current state
2 - Expenses per month
3 - Exit from the program
Your choice: 3
Application terminates
```