Technical Programming II

Practical Semester Test

Important information

- The due date for this assignment is **2 October 2023 at 23:00.**
- Submit a **single** PDF file containing all your source code to the folder identified as **Practical Assessment PDF**. The PDF file must be named with your student number only.
- Submit a single ZIP file with all the relevant source code (Python) files to the folder identified as **Practical Assessment Source**. The file must be named with your student number only.
- No deviation from the submission requirements will be entertained. Should you submit the wrong files to the wrong folders, you will fail this assessment.
- A code similarity report will be generated. This process considers submissions by your fellow students, all web sites, as well as code generated by AI engines.
- Should your similarity score exceed 35%, your final mark will be reduced with 1 mark for every percentage point exceeding 35%.
- Submissions with a similarity score exceeding 50% will automatically fail and will not be assessed.

Part 1 – The basic application [80]

Create a program that imports sales data from one or more CSV files, allows the user to add sales data, and writes the data to a file that contains all the sales. How you construct these files is up to you.

The screenshot of the running application displayed in **Figure 1** can be used as a guideline.

Specifications

- The program should accept float entries for the amount and integer entries for the year, month, and day.
- Store the functions for getting sales data from a user in a module named sales. Use docstrings and type hints to document the sales module.
- Calculate the quarter based on the month value:

  1, 2, 3 Quarter 1

  4, 5, 6 Quarter 2

  7, 8, 9 Quarter 3

  10, 11, 12 Quarter 4

- Use a list to store the sales data for each user entry and use a list of lists to store multiple entries.
- Create a menu that allows the user to view existing sales data, add new sales data, and import sales data.
- Allow users to import sales data from a CSV file that contains sales amount, year, month, and day. Use a text file to keep track of imported files so a user can't import a file more than once.
- Store the functions for writing and reading the files in a separate module.
- When the program starts, it should read the sales data from a csv file.
- When the program ends, it should write the sales data, including any imported data and any data entered by the user, to the same csv file.
- Don't store the quarter value. Instead, calculate it from the month as needed.

```
SALES DATA IMPORTER

COMMAND MENU
view   - View all sales
add    - Add sales
import - Import sales from file
menu   - Show menu
exit   - Exit program

Please enter a command: import
Enter name of file to import: region1_sales.csv

        Date            Quarter         Amount
-----------------------------------------------
1.*     2021-?-15       0               $?
2.      2021-8-15       3               $6426.0
-----------------------------------------------
TOTAL:                                  $6426.0

File 'region1_sales.csv' contains bad data.
Please correct the data in the file and try again.

Please enter a command: import
Enter name of file to import: region1 sales.csv

        Date            Quarter         Amount
-----------------------------------------------
1.      2021-4-15       2               $4246.0
2.      2021-8-15       3               $6426.0
-----------------------------------------------
TOTAL:                                 $10672.0

Imported sales added to list.

Please enter a command: add
Amount:          5892
Year:            2021
Month (1-12):    12
Day (1-31):      15

Sales for 2021-12-15 added.

Please enter a command: view
        Date            Quarter         Amount
-----------------------------------------------
1.      2021-2-15       1               $3425.0
2.      2021-4-15       2               $4246.0
3.      2021-8-15       3               $6426.0
4.      2021-12-15      4               $5892.0
-----------------------------------------------
TOTAL:                                 $19989.0

Please enter a command: exit

Bye!
```

Figure 1

- Handle the exception that occurs if the program can't find the csv file, the file the user is trying to import, or the text file that tracks imported files.
- Handle the exceptions that occur if the user enters a string or float where an integer is expected, or a string where a float is expected.
- Handle the exceptions that occur if the data in the imported file can't be converted to a float or an int. Use asterisks and question marks to notify the user of bad imported data.
- The program should validate the user's entries as follows:
- The sales amount should be greater than zero.
- The sales month should be between 1 and 12. The sales day should be between 1 and 31, except:
- For months 4, 6, 9, and 11, the maximum day value is 30.
- For month 2, the maximum day value is 28 (don't handle leap years). The sales year should be greater than or equal to 2000 and less than or equal to 9999.

Part 2 – Improve the application [80]

Improve the appearance of the console and the readability of the code.

Use **Figure 2** as a guideline.

```
SALES DATA IMPORTER

COMMAND MENU
view   - View all sales
add    - Add sales
import - Import sales from file
menu   - Show menu
exit   - Exit program

Please enter a command: add
Amount:              7392
Date (yyyy-mm-dd)    2021-15-37
Date must be valid and in 'yyyy-mm-dd' format.

Date (yyyy-mm-dd)    2021-4-20
Region:              m

Sales for 2021-04-20 added.

Please enter a command: view
     Date           Quarter        Region              Amount
-------------------------------------------------------------
1.   2020-12-22     4              West            $12,493.00
2.   2021-09-15     3              East            $13,761.00
3.   2021-05-15     2              East             $9,710.00
4.   2021-08-08     3              Central          $8,934.00
5.   2021-04-11     2              Mountain        $39,393.00
6.   2021-04-20     2              Mountain         $7,392.00
-------------------------------------------------------------
TOTAL:                                            $91,683.00

Please enter a command: import
Enter name of file to import: sales_q1_2021_z.csv

File name 'sales_q1_2021_z.csv' doesn't include one of the
following region codes: ['w', 'm', 'c', 'e'].

Please enter a command: import
Enter name of file to import: sales_q3_2021_w.csv

     Date           Quarter        Region              Amount
-------------------------------------------------------------
1.   2021-07-15     3              West            $13,761.00
2.*  ?              0              West             $9,710.00
3.*  2021-09-15     3              West                     ?
-------------------------------------------------------------
TOTAL:                                            $23,471.00

File 'sales_q3_2021_w.csv' contains bad data.
Please correct the data in the file and try again.
...
```

Figure 2

Specifications

- Use the decimal module to make sure the program doesn't yield incorrect total sales due to floating-point errors.
- Use the locale module to display the sales amount using formatting that's appropriate for your current locale.
- Use f-strings to format the widths and alignment of the columns in the display table.
- Use string operations to ensure the name of an imported file follows this format:

  sales_q*n*_*yyyy*_*r*.csv

  where *n* represents the sales quarter, *yyyy* represents the sales year, and *r* represents the sales region.

- Valid regions are w for West, m for Mountain, c for Central, and e for East.
- The program should get the sales region code from the filename and display and store it along with the other sales data.
- The program should also get a valid sales region from the user when they add sales data with the 'add' command.
- The imported CSV files should have one value for the sales date in YYYY-MM-DD format, rather than individual year, month, and day values.
- The year, month, and day should be entered by the user as a single entry in YYYYMM-DD format, rather than as individual year, month, and day values.
- When a user enters an invalid date or when an imported file contains an invalid date, the user should be given the opportunity to correct the date.
- Store the sales data as a list of dictionaries rather than a list of lists. To save the sales data to a CSV file, you'll need to convert it back to a list of lists.
- Store the valid regions in a dictionary with the region code (w, m, c, or e) as the key and the region name (West, Mountain, Central, or East) as the value.
- Use the regions dictionary to display the region name rather than the region code.

Part 3 – Create an object-oriented program [80]

Convert the program from procedural to object-oriented. This shouldn't change the functionality of the code, but it should make the code more modular and reusable and easier to maintain.

Use **Figure 3** as a guideline.

```
SALES DATA IMPORTER

COMMAND MENU
view   - View all sales
add    - Add sales
import - Import sales from file
menu   - Show menu
exit   - Exit program

Please enter a command: view
     Date           Quarter        Region             Amount
-----------------------------------------------------------
1.   2020-12-22     4              West             $12,493.00
2.   2021-09-15     3              East             $13,761.00
3.   2021-05-15     2              East              $9,710.00
4.   2021-08-08     3              Central           $8,934.00
-----------------------------------------------------------
TOTAL:                                             $44,898.00

Please enter a command: exit

Bye!
```

Figure 3

Specifications

- Use a Region class that provides attributes for storing the code and name for a region.
- Use a Regions class that stores the valid regions as Region objects in a list. This class should include a method that allows you to get a Region object by region code, and a method or property that returns a list of valid region codes. It should also provide a __str__() method that returns a string that includes the valid region codes.
- Use a File class to store a filename and Region object, as well as the expected naming convention for an imported file. This class should include a method that gets and returns the region code from the filename. It should also include methods or properties that check whether the filename is valid and that return the expected naming convention.
- Use a DailySales class that provides attributes for storing the amount, date, region, and quarter for sales data. This class should include methods that convert the data in a row of a CSV file to the appropriate

data types, convert the data in a DailySales object to a list that can be saved to a CSV file, and calculate the quarter based on the month, and methods or properties that check whether the sales data is valid.

- Use a SalesList class that stores DailySales objects in a list and provides an attribute for indicating whether any of the DailySales objects it contains has bad data. This class should include methods that add a DailySales object, retrieve a DailySales object by index, and add the contents of another SalesList object to the current list. This class should also provide a method or property that indicates how many items are in the list it contains. It should also include an iterator.
- In the module that contains these classes, provide a constant named DATE_FORMAT that the other modules in the program can use to format dates.
- Add a FileImportError class that inherits the OSError class. In the module that works with data files, the function that imports sales data should raises a FileImportError for the following issues:
    - The filename isn't in the expected format.
    - The filename doesn't contain a valid region code.
    - The file with the specified name has already been imported.
    - A file with the specified name is not found.
- Modify the code in the program so it catches a FileImportError and displays its error message to the user.
- Create separate Python files (modules) for the following:
    - The user interface (console)
    - The required classes and the constant
    - The functions that are required to work with data files.

--END--