Design and Programming

Assignment One

Preamble

The CSV (Comma Separated Values) format is a common import and export format for spreadsheets, databases and other applications, particularly in the data sciences. A CSV file stores tabular data in which each data field is separated by a delimiter (usually a comma). CSV files are saved with the .csv file extension.

Reading CSV files in Python

Python contains a module called **csv** for handling CSV files. The reader class from the module is used to read data from a CSV file. The CSV file is opened using the open() method in 'r' (read) mode which returns a **file object**. This is passed to the reader() method of the CSV module that in turn returns a **reader object** that can be used to iterate through the lines in the specified CSV file. Note that the reader() method has other optional arguments that govern how the CSV file is interpreted. It is not a trivial problem as fields within a CSV file may contain commas, for example, and hence are quoted within the CSV.

Note: In the examples below, the 'with' keyword is used along with the open() method as it simplifies exception handling and automatically closes the CSV file at the end of the subsequent block.

Example Python Code for Reading from CSV:

```
import csv
# opening the CSV file
with open('books.csv', mode ='r') as file:
 # reading the CSV file
    csvFile = csv.reader(file)
    print(type(csvFile))
   # displaying the contents of the CSV file
    for lines in csvFile:
        print(lines)
```

Writing to a CSV file

The csv.writer class is used to add data to a CSV file. This class returns a **writer object** which is responsible for converting the data into a delimited string. A CSV file object should be opened with newline='' (empty string) otherwise, newline characters inside quoted fields will not be interpreted correctly.

The csv.writer class provides two methods for writing to a CSV file. They are writerow() and writerows(). writerow() writes a single row at a time. This can be used for writing an initial 'header' row which is conventional for CSV files. writerows() writes multiple rows at a time.

Example Python Code for Writing to CSV:

```
import csv
# field headers
fields = ['Name', 'Age', 'Telephone']
# data rows
rows = [ ['Mike', 36,'621134'], ['Sue',24, '454545'], ['Bill', 36,
'232323']]
# name of csv file
filename = "people.csv"
# write to csv file
with open(filename, 'w') as csvfile:
    # create a csv writer object
    csvwriter = csv.writer(csvfile)
    # write the fields
    csvwriter.writerow(fields)
    # write the data
    csvwriter.writerows(rows)
```

The Problem

A small college library run by volunteers has donated its books to the case study library. They utilised a simple but effective spreadsheet-based cataloguing and loan system. Their books and data files are thus available for some experimental work – although they have not been able to provide member information for data privacy reasons. The library had a standard loan period of 14 days.

The file books.csv contains a list (a subset) of their books in CSV (comma separated value) format. If this exercise proves to be useful, the software produced will be used to analyse the entire dataset. The CSV files are encoded as UTF-8.

The file bookloans.csv contains data on book loans in CSV format. Each row of the file (there is no header row), holds a book_number, member_number, date_of_loan and date_of_return separated by commas. The date_of_return is recorded as 0 (zero) if the book has not been returned. The date_of_loan is a single integer number representing the date of the loan in Microsoft Excel Epoch Format. It represents the number of days elapsed since 1/1/1900. The file may contain loans for books which are not included in the books.csv file provided. These should be discarded. There may also be other surprises in the data which you should check before making any assumptions.

Tasks

1.

Write code to produce a report of the frequency with which books have been loaned in reverse order (the least frequent first). The report should indicate the number of times that books were loaned in 2019, along with their titles and author. The latter are required as there are some texts (classics in some way) which will be retained even if they have proved unpopular.

2.

Write code to produce a report of the most popular genres and sub-genres of books borrowed. This will inform the library as to the interests of its readers and therefore may influence future purchasing decisions.

3.

Write code to produce a report of the average length of time that a user has a book on loan and the proportion of users who have returned books late, together with the average late period.

Your code and documentation should be submitted as a Jupyter Notebook. In producing your solutions, you should consider how best to functionally decompose the system and what data structures are the most appropriate for producing an efficient solution. Functions should be tested, and you should demonstrate how you have achieved this. These design aspects of your solution will attract a maximum of 15 Marks of the 40 available. Achieving the correct functionality will attract a further maximum of 15 Marks and the remaining 10 Marks will be awarded to reflect the quality of your documentation. Your documentation should describe any 'sanity checks' that you have performed on the data, your opinion of the quality of the data and the extent to which it might inform the case-study project going forward. Also, describe any assumptions made and the extent to which you feel the completed tasks have met the requirements set. You need not submit the data files, but your code should function with the original data files present in the same folder as the Jupyter Notebook. You should submit a single Notebook (.ipynb) file.

David Collins,

November, 2020