Creating a procedural computer program

Procedural Oriented Programming (POP)

Password Manager

Situation Description

While talking to a colleague, you discover that they've recently stopped using their password manager of choice "***LastPass***", due to recent concerns over how *GoTo's* Inc security standards have been slipping.

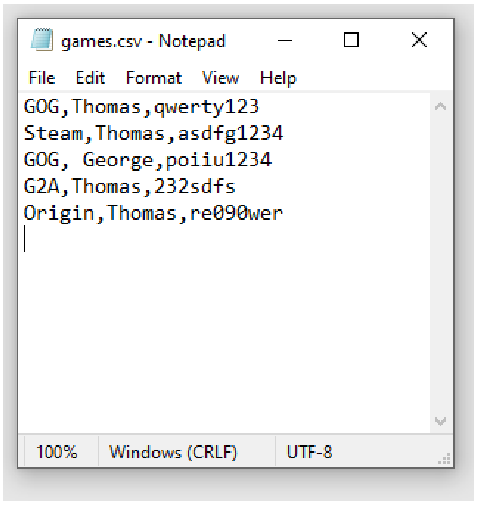
They explained, that they have resorted to storing all their passwords in a single (comma-separated values) csv file that they keep on a flash drive, kept on their keychain and of course they also use BitLocker to password protect the flash drive.

Although your colleague is computer literate and understands the need for password hygiene, they have committed to this process, out of necessity.

This colleague then goes on to explain that although functioning, this approach is less than desirable; lamenting that they miss features from their old password manager. They say, quite simply, they wish they could display credentials by entering the service name, add credentials, remove credentials and update existing credentials.

Your colleague also goes on to say, it would be nice to have the ability to randomly generate passwords, and a system to evaluate the strength of passwords. Perhaps even automatically updating the file. Continuing they say this program doesn't need to look nice, just simply provide the needed functionality even without some of these features.

Figure 1. Shows a sample of what one of these csv files looks like.



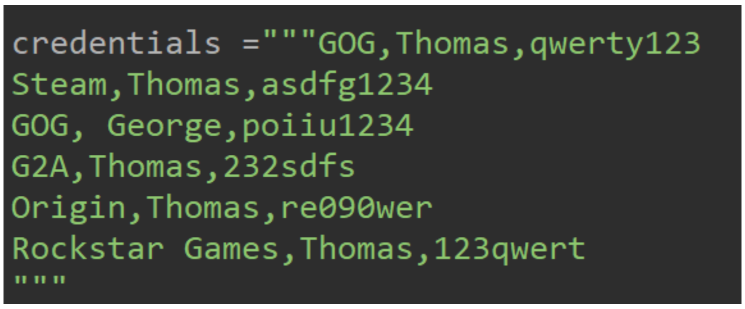
```
GOG,Thomas,qwerty123
Steam,Thomas,asdfg1234
GOG, George,poiiu1234
G2A,Thomas,232sdfs
Origin,Thomas,re090wer
```

Figure 1 Sample CSV file

You kindly; tell your colleague that you have been looking into Python for the last three weeks and you think this could be a fun challenge to practice the skills you learned.

Task A –
Implement a software design using procedural programming & refine a procedural program to improve quality

Create a Python program (**password_manager.py**) that stores credentials, consisting of service name, username, and password. With each element separated by a comma ',' and each credential on its own line. A pre-populated example may, look like the following in figure 2. However, you may choose to use a different name or structure for your data type, such as a list, set or anything else. *(You may wish to pre-populate this variable, while in development).*



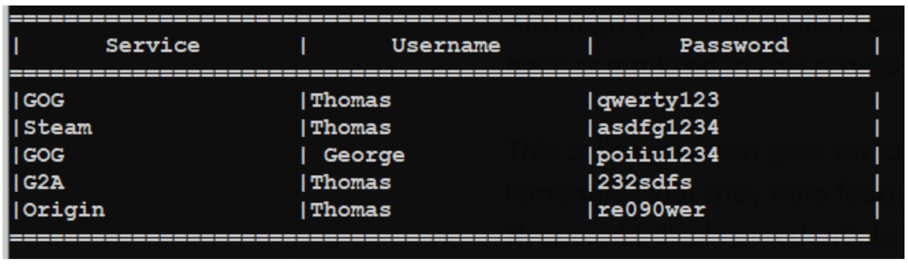
```
credentials ="""GOG,Thomas,qwerty123
Steam,Thomas,asdfg1234
GOG, George,poiiu1234
G2A,Thomas,232sdfs
Origin,Thomas,re090wer
Rockstar Games,Thomas,123qwert
"""
```

Figure 2 pre-populated credentials variable

1.**Display credential:**
*Within this program (**password_manager.py)** add code to do the following.*

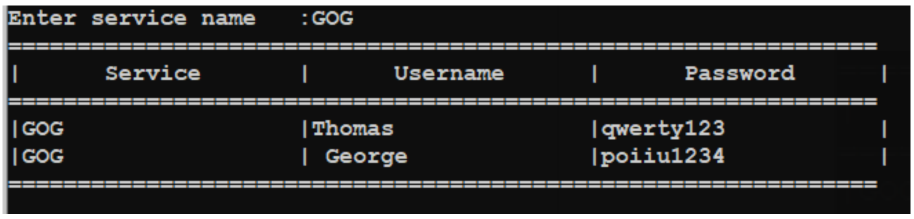
1.1. Display all data from the variable holding the credentials in a formatted table, as seen in figure 3.

1.2. Display only data from the variable holding credentials, that relates to a provided existing service name as seen in figure 4.



```
==================================================================
|        Service         |        Username        |        Password        |
==================================================================
|GOG                     |Thomas                  |qwerty123               |
|Steam                   |Thomas                  |asdfg1234               |
|GOG                     | George                 |poiiu1234               |
|G2A                     |Thomas                  |232sdfs                 |
|Origin                  |Thomas                  |re090wer                |
==================================================================
```

Figure 3 Display all credentials



```
Enter service name    :GOG
==================================================================
|        Service         |        Username        |        Password        |
==================================================================
|GOG                     |Thomas                  |qwerty123               |
|GOG                     | George                 |poiiu1234               |
==================================================================
```

Figure 4 Display credentials related to service "GOG"

2. **Add credentials:**
*Within this program (**password_manager.py)** add code to do the following.*

2.1. When adding a new credential, the user will be prompted to enter all required information, Such as Service name, Account Username, and Account Password. Then this information should be correctly formatted and added to the variable holding the credentials.

3. **Delete credentials:**
*Within this program (**password_manager.py)** add code to do the following.*

3.1. When deleting an existing credential, the user will be required to enter the service name, and the associated account username for the service they wish to delete. Then the variable holding credentials should be modified to remove the related, service name, account username and associated password.

4. **Update credential’s password:**
*Within this program (**password_manager.py)** add code to do the following.*

4.1. When updating an existing credential’s password, the user will be required to enter in the service name, the associated username, and the new updated password. Then the variable holding credentials should be modified to change the appropriate password.

5. **Generate random password:**
*Within this program (**password_manager.py)** add code to do the following.*

5.1. Write code that generates a string of randomly chosen characters; made up of alphanumeric characters, both upper and lowercase letters (A-Z, a-z,0-9). This string should also contain a subset of special characters, for example ('!','*','#’, ‘:','^','(',')','£','€','$', etc, ...).

5.2. Allow the user to specify the quantity of each type of character to be included.

6. **Load credentials:**
*Within this program (**password_manager.py)** add code to do the following.*

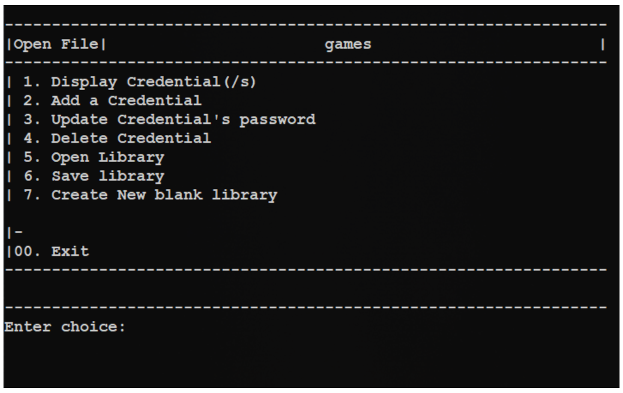
6.1. Write code that will retrieve data from a CSV file, and assign its data to the variable holding the credentials.

7. **Save credentials as:**
*Within this program (**password_manager.py)** add code to do the following.*

7.1. Write code that will store all the data from the variable holding the credentials to a CSV file.
7.2. Allow the user to specify the name of the file.

8. **CLI Menu:**
This may be done in a separate Python file (**CLI_password_manager.py**)

8.1. Create a command line interface menu, as seen in figure 5. This menu should loop, until the user chooses to close the program. Any invalid input should be handled correctly. (*Note; the menu does not have to be exactly the same as shown in figure 5, this is merely to serve as an example.)*



```
----------------------------------------------------------------
|Open File|                    games                           |
----------------------------------------------------------------
| 1. Display Credential(/s)
| 2. Add a Credential
| 3. Update Credential's password
| 4. Delete Credential
| 5. Open Library
| 6. Save library
| 7. Create New blank library

|-
|00. Exit
----------------------------------------------------------------

----------------------------------------------------------------
Enter choice:
```

Figure 5 Example User Menu