COSC 1104

Scripting

Assignment 2

Student Name: Arjun Bindhu Suresh
Student ID : 100990351

Part 1

Problem 1: Student Database

Problem description

Handling the student data can be slow and error-prone for small schools or academic programs with many students. One quick and easy option is to design a system in which the student information (name, roll number, marks) is easily stored and edited. It's a common issue for a school or anyone with a small student database.

A lot of times the students or teachers or the administrators are simply needing an easy system to add a student, see all students records, and find a particular student by their roll number. When you don't have a database system in place, searching or adding students is tedious. The system will make storing and retrieving student information automatically and intuitively.

Proposed Solution

We can write a python code to create a list or dictionary database for students (their name, roll number, marks etc.). It will also have basic features such as adding new students to the database, seeing every student's information and searching for a student based on the roll number. If you have a list or dictionary, then the database will be stored in memory so that you can easily access and edit it. The script will have an easy-to-use user input system. These functions will let you store student information and get back needed information easily for the small-scale academic administration work.

Advantages and libraries

There are various benefits of student database systems like data storage, persistence, scale, ease of use, powerful search and filtering features. By using very basic data models, such as dictionaries or lists, students' records can easily be retrieved, added and modified by the system. As a file storage system built-in with the json library, students' information is stored across sessions so that data never gets lost and can be accessed by all users forever. It is also scalable so that other student information, like grades or attendance history can be added in the future. Its intuitive design means that administrators or educators don't have to know much tech in order to use the system, following simple prompts to add, view or search student records. This is possible thanks to json library, which contains data in lightweight and accessible form, which is very suitable for simple database management. os library takes care of safe file operations, if data file is not present it generates it. Also, datetime provides for tracking when records were added/modified. The system is very user friendly and easy to implement for smaller scale, like a classroom or small school project.

Problem 2: Basic Banking System

Problem description

It is difficult to set up a basic banking system that will allow the user to open an account, pay in, withdraw money and check balance. The program will persist account information such as username, account number, and balance in a dictionary or class for quick access and change. The main features of the system are account opening, which is where you open an account and deposit money into your account and withdraw money if you have enough money. A user can also see their balance on the account. It will verify the inputs and prevent users from withdrawn funds beyond the balance that is left, it will perform normal operations, and only basic errors will be processed. The code will have functions for each operation, conditionals for input validation, and a dictionary or class for the account information which is going to be a simple but functional banking simulator.

Proposed Solution

The solution is to write a Python program to mimic a simple bank system and allow you to create an account, deposit, withdraw, and check your balance. The account information (username, account number, balance) will be written in a dictionary or class. The system will include a set of functions: the account create function, which accepts the user's name and the first deposit, a deposit method, which deposits money into the user's account and adjusts the balance, a withdrawal mechanism, which withdraws but does not overdraw the account balance, and finally a balance checker to view the account balance. The program will be simple to use and will give the user textual instructions on how to take these steps. It will check deposits and withdrawals against conditionals to verify the transactions. The solution is flexible so you can simply update in the future and add more functionality or build out account management features.

Advantages and libraries

A This Basic Banking System has many benefits such as data persistence, where account information is stored and retrieved using the json library which will make sure that the information of users is not lost between sessions. It offers simple data management with simple entities such as dictionaries, and it's easy to run accounts, deposit money, withdraw money, and see balances. With the system's easy to use interface, you can easily interact, and the fast transactions will give you immediate reports on balances. Furthermore, it is scalable for future development and easy to use and understand for a beginner. Packages such as json support data storage, os the file system, and datetime stores transaction dates with the information about time. This combination of libraries and functionality, however, has the Basic Banking System as a secure, extensible, and intuitive way to implement basic banking functionality.

Part 2

https://github.com/Arjun-Bindhu-Suresh/Assignment-2.git

Part 3

1. **What was the most challenging aspect of solving this problem?**

   What was hardest to resolve in both the Student Database and Basic Banking System was data integrity and ensuring that the system processed user input correctly. For the Student Database issue, I had to for-mat data so it would be easy to add, display and search student records. Dictionaries or lists had to be both convenient and performant, particularly when you searched by roll number. Also checking user data (like a roll number format being entered in the correct format, or if a student's details were entered as invalid) was a complex part of the program that required special care.

   The issues for Basic Banking System were about account balances being correct and multiple types of transactions being processed like deposit, withdrawal, and balance check. The system had to be correctly checked to make sure users weren't withdrawing more money than they had in their account. Also, it took design to allow user movements to flow and show users the outcomes of their actions. If a dictionary to store account data was enough for a very small solution, we'd need to persist in user data between sessions if we were going to have a much bigger solution, which would need to be handled more complexly. In the grand scheme of things, both issues necessitated data flow, validation of input, and a system that was easy to understand and not in error.

2. **What was the most valuable thing you learned from this assignment?**

   I learned the most from this assignment, how to create small and yet useful data management systems based on simple data types such as dictionaries and lists. Working on the Student Database and Basic Banking System gave me real-world experience in working with and manipulating user data that's the core of almost every real-world application. I was also taught about input validation and error handling, so the system runs without hiccups when a user inputs wrong or unexpected data.

I also figured out how to make code modular through functions and the results were not only easy to read and maintain but organized and extensible. The tasks also stated that the user's experience should be very clear and intuitive, especially with real-time transactions such as in the banking system. In sum, this project made me better aware of fundamental programming concepts and how they can be used to develop usable systems.

3. **Is there something you would still like to add to this, or something it makes you want to try next?**

   So, both for Student Database and Basic Banking System I would like to make improvements that improve the experience. I could add something to the Student Database such as data persistence in the file store (for example, storing and reading student data from a file) so that it does not become corrupt when the program is closed. We can also do some research by student name or sort options (e.g., marks or roll number) to make the system even easier and faster for larger datasets.

   In the Basic Banking System, I want to see more modern features like transaction history, inter-account money transfer and bank account types (checking, savings). A possible additional feature might be a security system such as PIN/password for account login, to give it more real-world and safe feeling. Then also considering database management systems (DBMS) like SQLite for management and access to account data may be the next option to make the system expandable for bigger applications.