NAME- MRIDUL SINGH RAWAT

SAP ID – 500119881

BATCH 32

ROLL NO. – R2142231074

CONTACT NO. – 7906465904

**summary of the code:**

This Python code defines two classes: **Account** and **PasswordManager**, along with a **main()** function to interact with the **PasswordManager** class.

Here's a summary of each component:

1. **Account Class**:
   * The **Account** class represents user accounts with attributes **name** and **password**.
   * The constructor **\_\_init\_\_()** initializes the account with a name and a password.
   * The **is\_strong\_password()** method checks if the password meets certain criteria for strength, including having at least one uppercase letter, one lowercase letter, one digit, and one special character.
2. **PasswordManager Class**:
   * The **PasswordManager** class manages a collection of user accounts.
   * It uses a dictionary **accounts** to store account names as keys and **Account** objects as values.
   * The **\_hash\_password()** method hashes the password using a simple hashing algorithm based on the ASCII values of characters.
   * Methods include:
     + **add\_account()**: Adds a new account to the manager.
     + **get\_password()**: Retrieves the password for a given account.
     + **delete\_account()**: Deletes an account from the manager.
     + **list\_accounts()**: Lists all the account names.
     + **suggest\_strong\_password()**: Generates a strong password of a specified length.
3. **main() Function**:
   * The **main()** function serves as the entry point of the program.
   * It creates an instance of **PasswordManager**.
   * It runs a loop to continuously prompt the user for options such as adding an account, getting a password, deleting an account, listing accounts, generating a strong password, or exiting the program.
   * It validates user inputs and executes the corresponding actions using methods of the **PasswordManager** class.

**USE OF CLASSES**

there are two classes: **Account** and **PasswordManager**, both of which serve distinct purposes in managing accounts and passwords within the system.

1. **Account Class**:
   * The **Account** class represents individual user accounts within the system.
   * It has attributes:
     + **name**: Represents the name or identifier of the account.
     + **password**: Represents the password associated with the account.
   * The **\_\_init\_\_()** method initializes an **Account** object with a given name and password.
   * The **is\_strong\_password()** method checks whether the password associated with the account meets certain criteria for being considered strong, such as containing at least one uppercase letter, one lowercase letter, one digit, and one special character.
2. **PasswordManager Class**:
   * The **PasswordManager** class manages a collection of user accounts.
   * It utilizes a dictionary named **accounts** to store account names as keys and corresponding **Account** objects as values.
   * Methods include:
     + **add\_account()**: Adds a new account to the manager's list of accounts.
     + **get\_password()**: Retrieves the password associated with a given account name.
     + **delete\_account()**: Removes an account from the manager's list of accounts.
     + **list\_accounts()**: Provides a list of all account names currently managed by the password manager.
     + **suggest\_strong\_password()**: Generates a strong password of a specified length, which can be suggested to users when creating or updating passwords.

These classes encapsulate the functionalities related to managing user accounts and passwords, promoting code organization, reusability, and modularity. The **Account** class handles individual account properties and behaviors, while the **PasswordManager** class orchestrates account-related operations and maintains a collection of accounts. This object-oriented approach enhances the code's readability, maintainability, and scalability.

Top of Form