PYTHON PROGRAM

PASSWORD MANAGER

Overview:

This password manager program uses a combination of hashing and encryption to store user passwords securely in a SQLite database. The program includes features to add, retrieve, update, and delete passwords, as well as a master password to encrypt and decrypt the password database.

The program uses the following modules:

- **`sqlite3`**: to create and manage the SQLite database
- **`hashlib`**: to hash and verify the master password and user passwords
- **`cryptography`**: to encrypt and decrypt the password database
- **`getpass`**: to prompt the user for passwords without displaying them on the screen

The program is divided into several functions:

- **`hash_password()`**: hashes a password using the SHA-256 algorithm and returns the hashed password as a string.
- **`verify_password()`**: verifies a password by hashing it and comparing it to a known hash.
- **`encrypt_database()`**: encrypts the password database using a user-specified master password.
- **`decrypt_database()`**: decrypts the password database using a user-specified master password.
- **`create_database()`**: creates a new password database with a user-specified filename and master password.
- **`add_password()`**: adds a new password to the password database.
- **`get_password()`**: retrieves a password from the password database.

- **`update_password()`**: updates an existing password in the password database.
- **`delete_password()`**: deletes a password from the password database.
- **`main_menu()`**: displays the main menu and prompts the user for a choice.

Python:

1. Start by importing the necessary modules: **`sqlite3`**, **`hashlib`**, and **`getpass`**.

Code::

```
import sqlite3

import hashlib

import getpass
```

2. 2. Connect to a SQLite database where we'll store our encrypted passwords. If the database doesn't exist, create it.

CODE::

```
conn = sqlite3.connect('passwords.db')

c = conn.cursor()
```

c.execute('''CREATE TABLE IF NOT EXISTS passwords (id INTEGER PRIMARY KEY,

website TEXT, username TEXT, password TEXT)''')

conn.commit()

3. Create a function to generate a hash of the user's password, which we'll use to encrypt the passwords stored in the database.

CODE::

def hash_password(password):

return hashlib.sha256(password.encode('utf-8')).hexdigest()

4. Create a function to add a new password to the database. This function should prompt the user for the website, username, and password, encrypt the password using the `hash_password` function, and insert the data into the database.

CODE::

def add_password():

website = input("Enter website: ")

username = input("Enter username: ")

password = getpass.getpass("Enter password: ")

hashed_password = hash_password(password)

c.execute("INSERT INTO passwords (website, username, password) VALUES (?, ?, ?)",

(website, username, hashed_password))

conn.commit()

print("Password added successfully!")

5. Create a function to retrieve a password from the database. This function should prompt the user for the website and username, and then query the database for the encrypted password. If a password is found, decrypt it using the `hash_password` function and print it to the console.

CODE::

def get_password():

website = input("Enter website: ")

username = input("Enter username: ")

c.execute("SELECT password FROM passwords WHERE website = ? AND username = ?",

(website, username))

```
    result = c.fetchone()

    if result:
        hashed_password = result[0]
        password = input("Enter master password to decrypt password:
")
        if hash_password(password) == hashed_password:
            print("Password:", hashed_password)
        else:
            print("Invalid master password!")
    else:
        print("Password not found!")
```

6. Finally, create a simple menu that allows the user to choose between adding a password or retrieving a password.

CODE::

```
while True:
    print("1. Add password")
    print("2. Get password")
    print("3. Exit")
```

```
choice = input("Enter choice: ")


if choice == '1':
    add_password()
elif choice == '2':
    get_password()
elif choice == '3':
    break
else:
    print("Invalid choice!")
```

This is a very basic outline of a password manager program in Python. You can add more features, such as updating or deleting passwords, adding additional security features, or creating a graphical user interface.