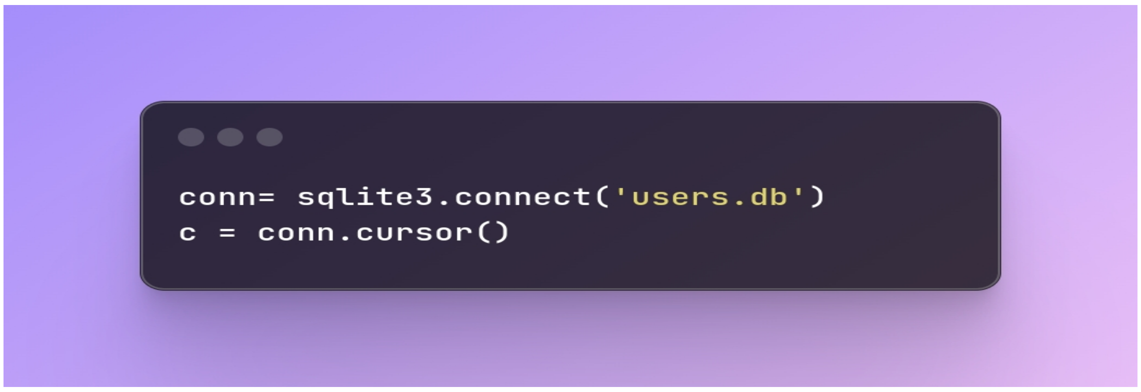
Enhanced User Authentication and Management System

Required Library:

- bcrypt
- sqlite3
- regressionexpression(re)

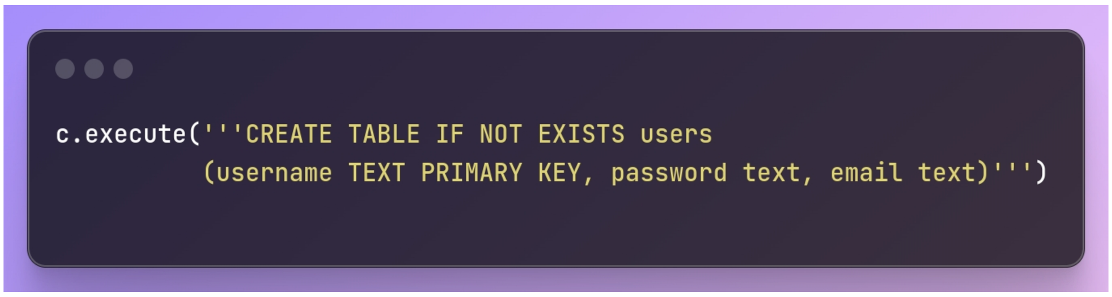
1. User Registration:

Database Connection:



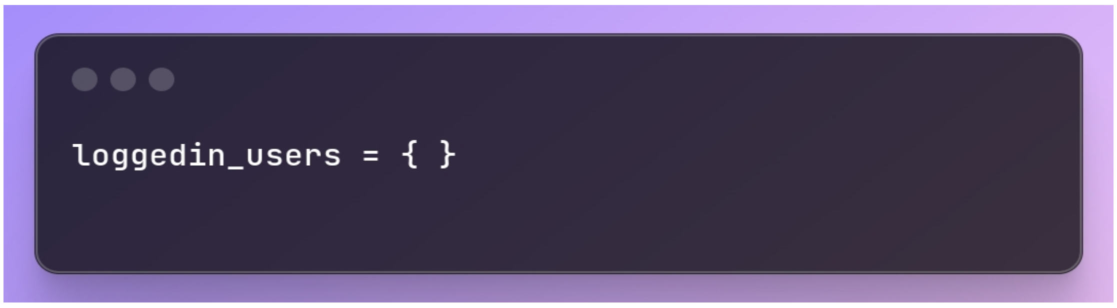
```
conn= sqlite3.connect('users.db')
c = conn.cursor()
```

Creating Table for storing users data



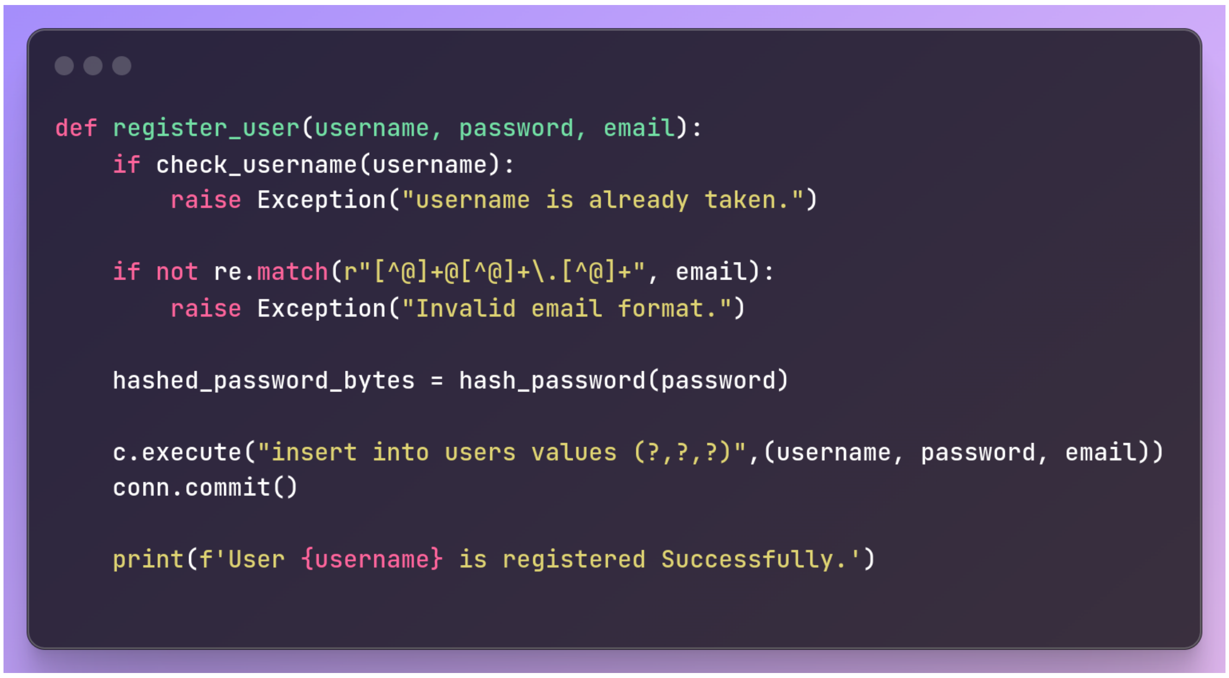
```
c.execute('''CREATE TABLE IF NOT EXISTS users
          (username TEXT PRIMARY KEY, password text, email text)''')
```

Session Management of the logged in users.



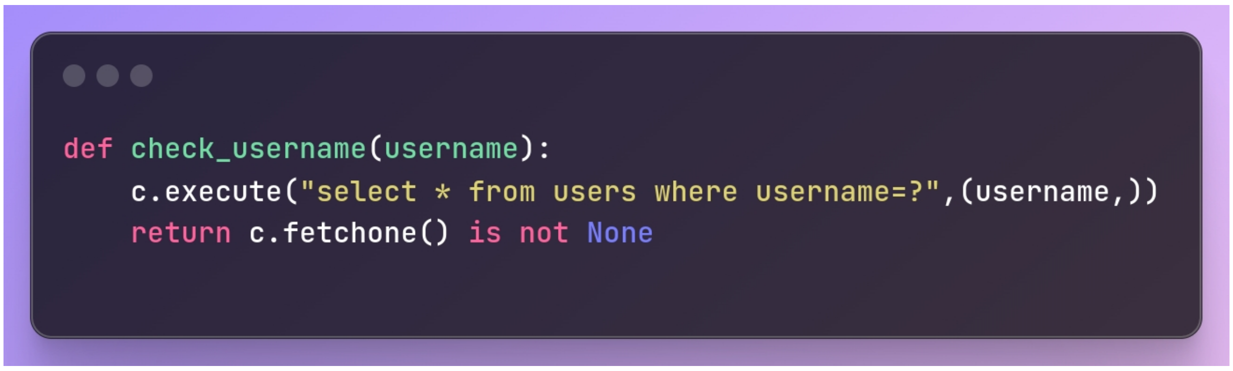
```
loggedin_users = { }
```

User Registration function is defined with username, password and email as a parameters.



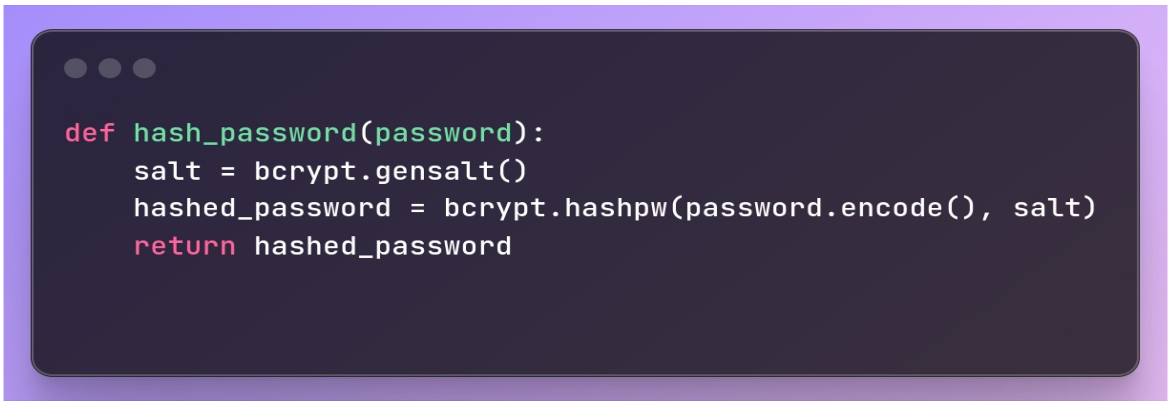
```python
def register_user(username, password, email):
    if check_username(username):
        raise Exception("username is already taken.")

    if not re.match(r"[^@]+@[^@]+\.[^@]+", email):
        raise Exception("Invalid email format.")

    hashed_password_bytes = hash_password(password)

    c.execute("insert into users values (?,?,?)",(username, password, email))
    conn.commit()

    print(f'User {username} is registered Successfully.')
```

Creating function to check username from the database and return the username the table if it is not registered in database.



```python
def check_username(username):
    c.execute("select * from users where username=?",(username,))
    return c.fetchone() is not None
```

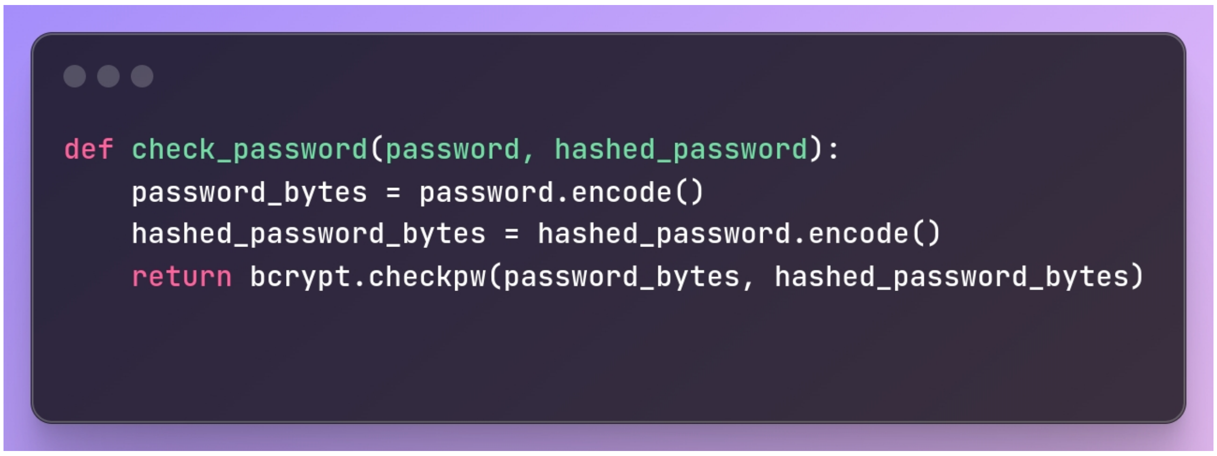
Creating hash function to hashed the given password by using **bcrypt** libraries and **gensalt** method.



```python
def hash_password(password):
    salt = bcrypt.gensalt()
    hashed_password = bcrypt.hashpw(password.encode(), salt)
    return hashed_password
```

****Note :** Invalid salt Error may occur how the hashed password is being stored, retrieved, or used.

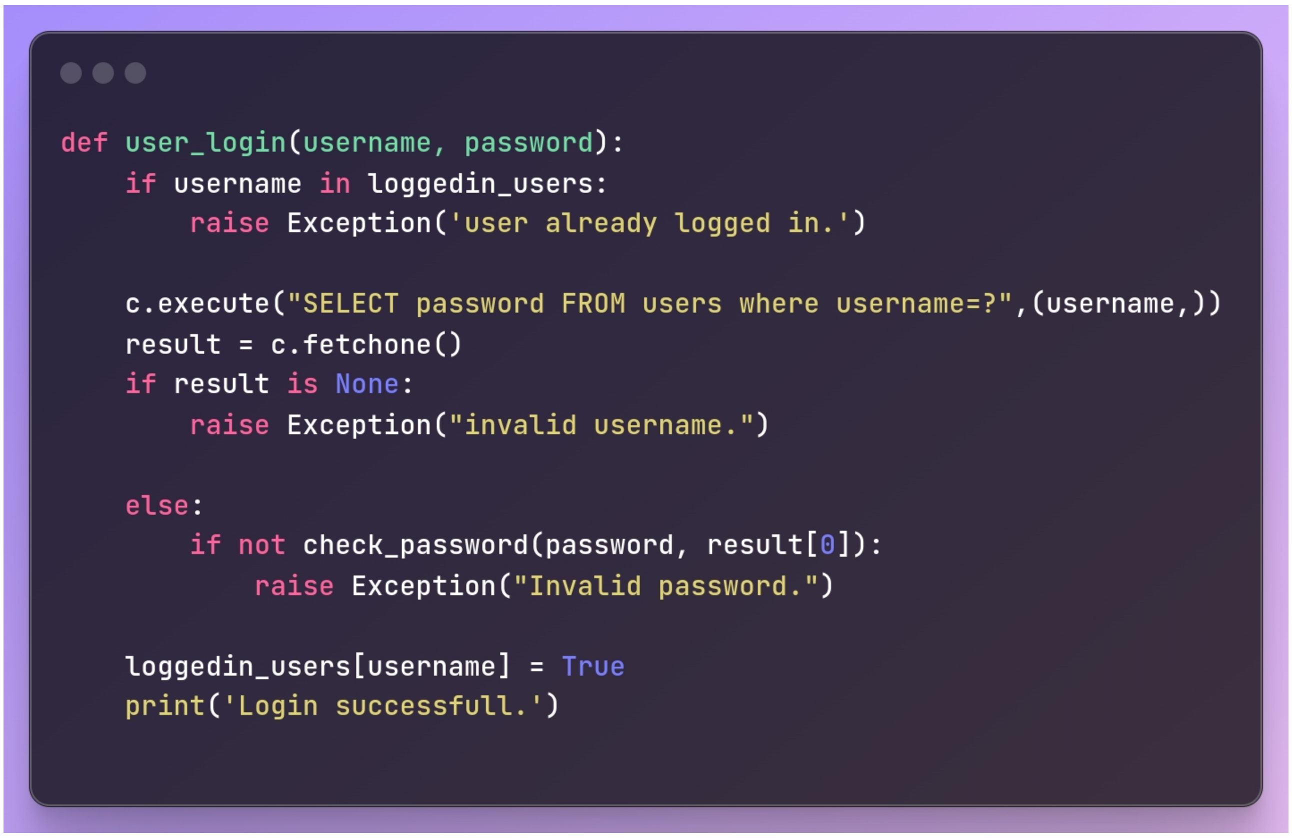
Creating check password functionality in order to verify the input login process of user with the database sources and enable them to login in to the system.



```python
def check_password(password, hashed_password):
    password_bytes = password.encode()
    hashed_password_bytes = hashed_password.encode()
    return bcrypt.checkpw(password_bytes, hashed_password_bytes)
```

2. User Login function:

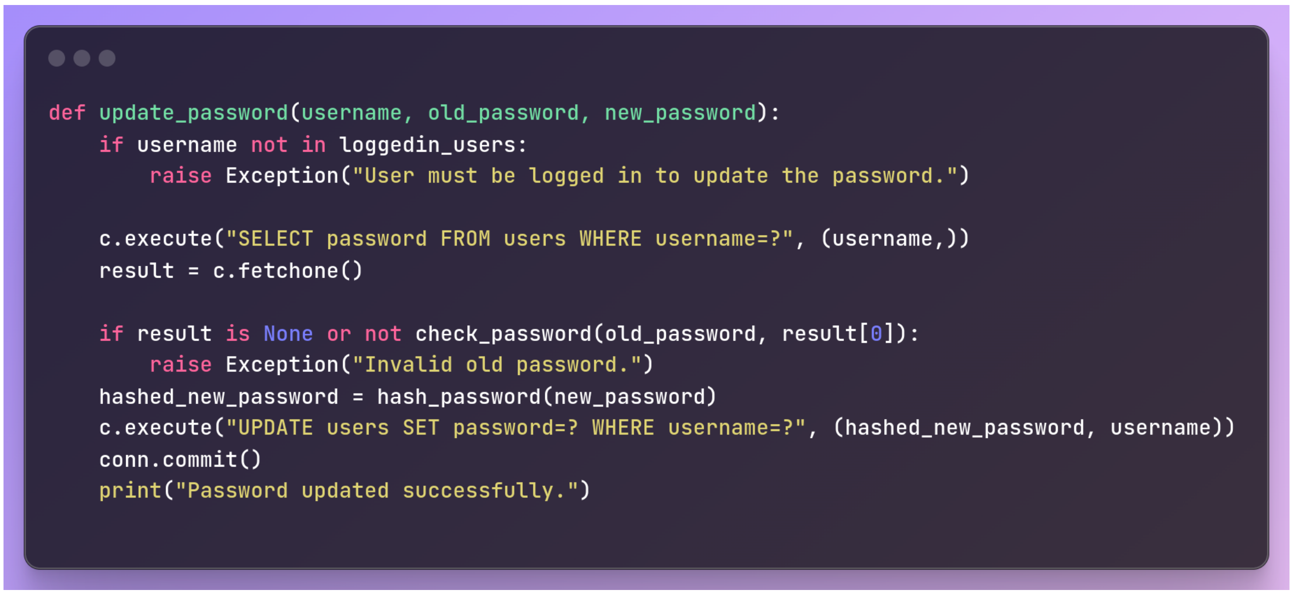
This function fetch the password from database and match the password and username to the input data, for getting login.



```python
def user_login(username, password):
    if username in loggedin_users:
        raise Exception('user already logged in.')

    c.execute("SELECT password FROM users where username=?",(username,))
    result = c.fetchone()
    if result is None:
        raise Exception("invalid username.")

    else:
        if not check_password(password, result[0]):
            raise Exception("Invalid password.")

    loggedin_users[username] = True
    print('Login successfull.')
```

3. Update Password:

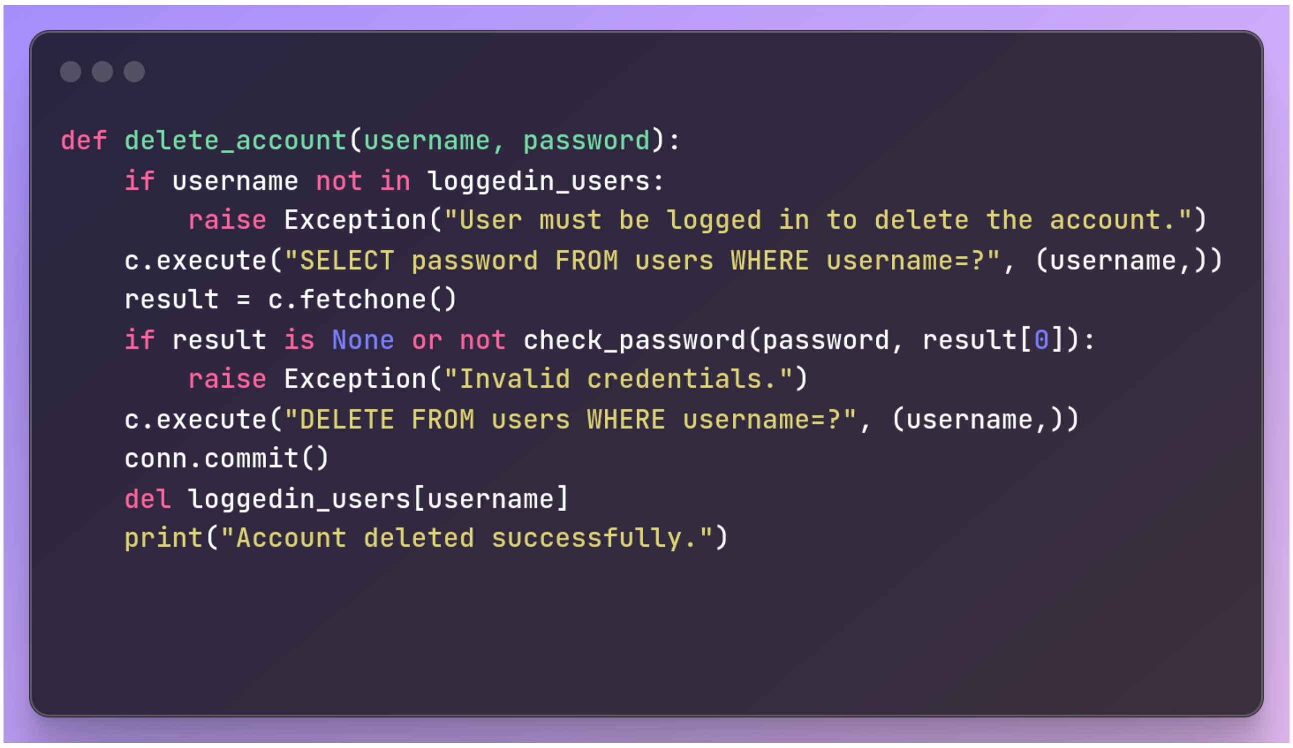
Here, this update password function takes username, old password and new password as a input that allows a user to update their password. It only allow to change the password if the current user is logged in otherwise, the user is not able to perform the update operations.



```python
def update_password(username, old_password, new_password):
    if username not in loggedin_users:
        raise Exception("User must be logged in to update the password.")

    c.execute("SELECT password FROM users WHERE username=?", (username,))
    result = c.fetchone()

    if result is None or not check_password(old_password, result[0]):
        raise Exception("Invalid old password.")
    hashed_new_password = hash_password(new_password)
    c.execute("UPDATE users SET password=? WHERE username=?", (hashed_new_password, username))
    conn.commit()
    print("Password updated successfully.")
```

4. Account Deletion:

This Deletion function aid users to delete their account. It also perform the same functionalities as update password do as the user must have to logged in first then only able to do perform deletion task.



```python
def delete_account(username, password):
    if username not in loggedin_users:
        raise Exception("User must be logged in to delete the account.")
    c.execute("SELECT password FROM users WHERE username=?", (username,))
    result = c.fetchone()
    if result is None or not check_password(password, result[0]):
        raise Exception("Invalid credentials.")
    c.execute("DELETE FROM users WHERE username=?", (username,))
    conn.commit()
    del loggedin_users[username]
    print("Account deleted successfully.")
```