HyperionDev

Introduction to Hashing

Model-Answer Approach

Visit our website

Auto-graded task

The provided Python script uses the `bcrypt` library to securely hash a user-provided password. Firstly, a function named `hash_password` is defined, which takes a password string as input. Within this function, the password string is encoded using the `encode()` method to ensure compatibility with `bcrypt`. A random salt is generated using `bcrypt.gensalt()`, which adds additional randomness to the hashing process, strengthening security against potential attacks like rainbow-table attacks. Then, the `bcrypt.hashpw()` function is called with the encoded password and the generated salt as arguments to produce the hashed password. This hashed password is returned by the function.

The script prompts the user to enter a password, which is then passed to the `hash_password()` function. The resulting hashed password is stored in the variable hashed_password. Finally, the hashed password is printed to the console after decoding it from bytes to a string using the `decode()` method. This approach ensures that passwords are securely hashed using a strong cryptographic algorithm, safeguarding sensitive user data against unauthorized access.