Flask Register/Login Tips

As you begin to add the register and login routes to your web page the following components and code samples may help. We definitely need to safely store passwords and verify if the password a user enters matches the stored password. Since we haven't installed a database for this course, we can just use a flat file. However; we can't just store the password in plain text as someone could easily steal that credential with minimal effort if they came across that file.

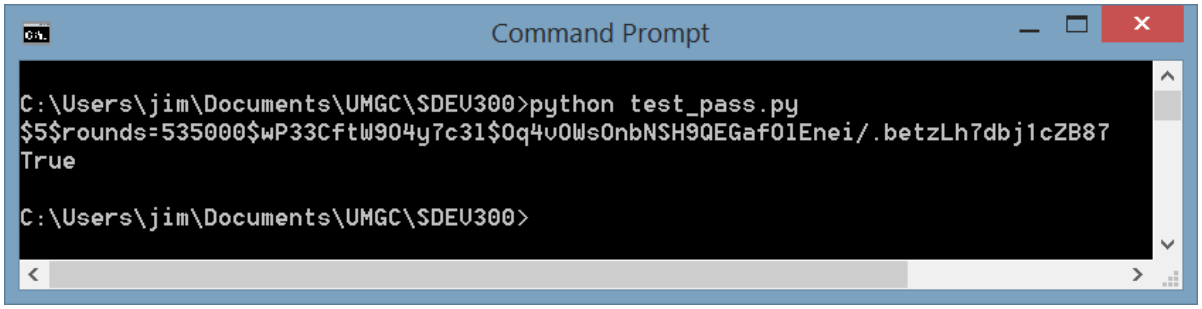
We definitely need a hashing technique to protect the data at rest in the file, and need the ability to compare the hashed version of the password with a user entry to authenticate the user. Fortunately, a hashing module exists called passlib. You can install it just like any other module by typing the following at the command prompt:

```
pip install passlib
```

Once installed, you can test it with a program such as the one below.

```
from passlib.hash import sha256_crypt

#Input a test password
test_pass = "sdev300Key!2"

# Hash the password
hash_pass = sha256_crypt.hash(test_pass)

# View the encrypted password
print(hash_pass)

# Call verify to compare the two entries
print(sha256_crypt.verify(test_pass, hash_pass))
```

If the file had been saved as test_pass.py, figure 1 illustrates running the source code. Note the hashed password is printed followed by True indicating a positive comparison of the passwords.



Be sure to test and experiment with this code as these methods will be quite valuable in the login route when you compare the user entered password to the stored password as well as the register route when the password is initially stored.

More details about the actual algorithm can be found here:

#https://passlib.readthedocs.io/en/stable/lib/passlib.hash.sha256_crypt.html

Previously, in this course, you have read about File I/O. A file will work for storing the credentials for this project. Recall, you can open a file named 'passfile' for append using the following Python code:

```
# Open file
with open('passfile', "a") as f:
    f.writelines(write_str)
```

Additional details and documentation can be found here:

https://docs.python.org/3/tutorial/inputoutput.html#reading-and-writing-files

The nice thing about using with open is the file is automatically closed.

With the combination of hashing and writing (and reading) files you can do most of the work needed for registering and login a user into your web site.

A couple of additional hints:

1. When you append a new user to your password file, you should read and check the file to make sure the user hasn't already registered. You can do this while checking to make sure the username and password are not null as well as the password complexity.

   ```
   if not username:
       error = 'Please enter your Username.'
   elif not password:
       error = 'Please enter your Password.'
   elif not checknotreg(username)
       error = 'You are already registered'
   elif not complex(password)
       error = 'Make your password more complex'
   ```

2. Using both get and post methods in the route method will allow the form to be displayed continuously until errors are resolved.
3. Use `flash(error)` along with `from flask import flash` to alert the user to errors in the forms.
4. Code a little test a lot. This will help you fix errors as they arise.