Private HTTP

Instructions

1. Use the same HTTP server we've worked on in the recap and Advanced Sockets topics, but rename it to `privatehttp.py`. You may use any version of the server, it doesn't matter since we won't rely on any of the features developed in the Advanced Sockets topic; only make sure the server is set to listen on port 8005 and the basic functionality works (can serve static pages).
2. Define hard-coded username/password inside the server code: user "Admin" with password "Password1!"
3. Add support for HTTP request header `Login-Details`. The value of this header shall be a base64-encoded JSON of a username and password.
4. Create a static page `/secret.html` and make it protected - only allow serving this page if the request contains correct Login-Details header that authenticates as the `Admin` user. 4.1. If a client requests the secret page without supplying `Login-Details`: server should return status 401 4.2. If a client requests the secret page, supplying the header `Login-Details` but with a value that's incorrect/unparsable: server should return status 403
5. Write a client script `privatehttpclient.py` and save it alongside the server, in the same directory. The client shall be able to successfully request the secret page. The client script gets username/password as command line arguments and uses them to retreive the secret page, then print its content to the user. You may use a package for HTTP requests, e.g. `requests`.

Example

Commandline: `python privatehttpclient.py Admin Password1!`
JSON: `{"username": "Admin", "password": "Password1!"}` Base64:
`eyJ1c2VybmFtZSI6ICJBZG1pbiIsICJwYXNzd29yZCI6ICJQYXNzd29yZDEhIn0=`
HTTP request header: `Login-Details:`
`eyJ1c2VybmFtZSI6ICJBZG1pbiIsICJwYXNzd29yZCI6ICJQYXNzd29yZDEhIn0=`

To submit

Submit server code `privatehttp.py` and client code `privatehttpclient.py`.