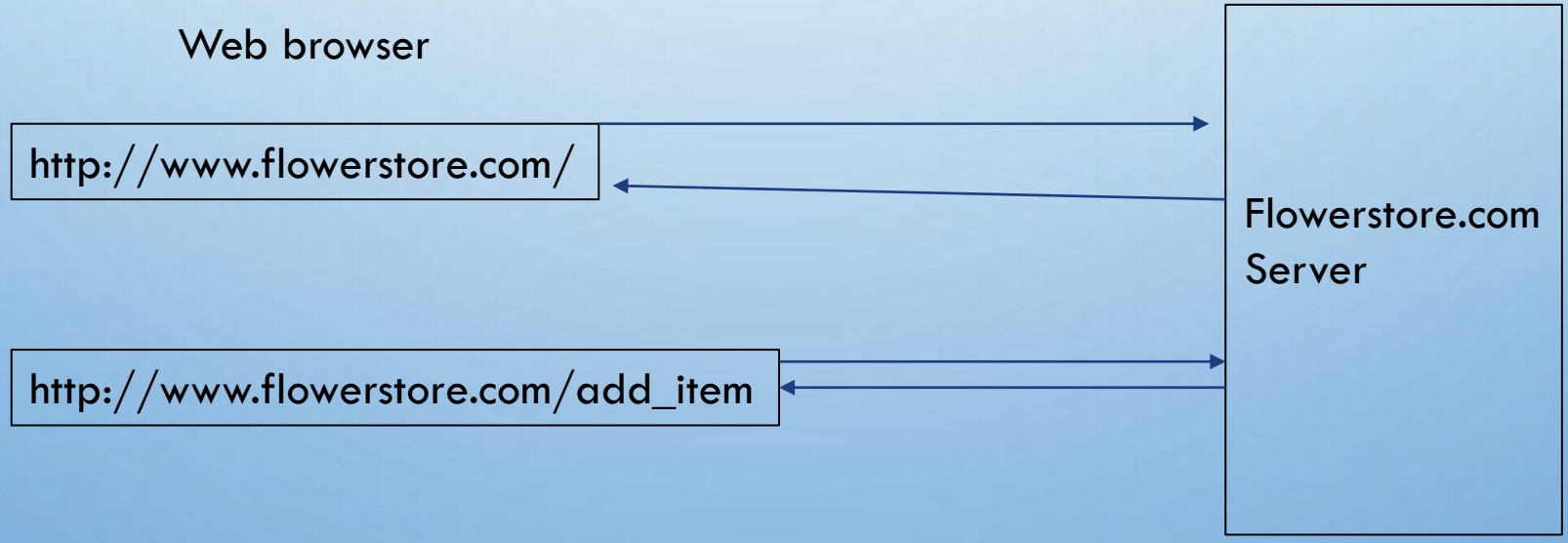
LECTURE 5 FLASK SESSION VARIABLES

Flask session variables

- Flask is a web application framework written in Python.
  - Web applications interact with the HTTP protocol
- HTTP protocol is stateless: each request has no knowledge of the previous request.
  - For some websites to work, the concept of session (store user information across multiple requests) need to be there (e.g. authenticated user or not, items in shopping cart)



Flask session variables

- Flask session allows you can store information specific to a user for the duration of a session across multiple requests.
  - Mechanism: sessions are stored as client-side cookies
  - Browser cookies are small chunks of data stored on your computer by the web browser, with the original intent being to remember stateful information when browsing different websites.
  - In a website with login, a session can be between user login and user logout.
- Flask session document: https://flask-session.readthedocs.io/en/latest/

Flask session variables

- Set the app secret_key

```
# NOTE: The secret key is used to cryptographically-sign the cookies used for storing # the session data.
app.secret_key = 'BAD_SECRET_KEY'
```

- Setting the value of a session variable

```
# Save the form data to the session object
session['name'] = request.form['username']
session['logged_in'] = True
```

- Accessing a session variable
  - if not session.get('logged_in'):
  - If not session['logged_in']:

Example of Flask session and a website with role based access control

- lect16/ExampleCrapsRoleBasedAccessControl.zip
  - To run the website on http://127.0.0.1/ : After unzip the file, 'python3 setup.py' and then 'python3 newPlayer.py'
  - Setup.py creates three users all with password 'test123': Princess Diana (admin, UserLevel = 1), Henry Thorgood (regular user, UserLevel = 2), and Tina Fairchild (regular user, UserLevel = 2).

Create and maintain sessions

- See lect16/ExampleCrapsRoleBasedAccessControl.zip for example of session variables and a website with role based access control.
  - Create a session between login and logout (See newPlayer.py).
  - At login, set the session variables

```
session['logged_in'] = True
session['name'] = nm
if (int(row['UserLevel'])==1):
   session['admin'] = True
else:
   session['admin'] = False
```

  - At logout,. Reset the session variables

```
session['logged_in'] = False
session['admin'] = False
session['name'] = ""
```

Role based access control

- The function for each page checks whether the user is logged in and whether the user is an admin, and acts accordingly.
- Display proper links (function) for each role after user login – see the code for home page.
- Both Flask python code and html template code can be used to realize such functionality.

```
@app.route('/')
def home():
   if not session.get('logged_in'):
      return render_template('login.html')
   else:
      return
render_template('home.html',name=session['name'])
```

```
{% if session['logged_in'] %}
    <h1>Welcome {{name}} </h1>
       {% if session['admin'] %}
          <h2><a href="/enternew">Add new Player</a></h2>
             <h2><a href="/list">List Player</a></h2>
       {% endif %}
         <h2><a href="/logout">Log out</a></h2>
{% endif %}
```

Some detail

- In the code, we also use the flash() function to store messages in the python code to be retrieved in the HTMP template.
- The html template can use the get_flashed_message() function to retrieve the messages.

Encrypting data in SQLite3

- Assume that we store encrypted username and password in SQLite3
- Encryption and decryption operations need to be inserted properly in order for the website to function correctly.
- For example:
  - When login, the username and password need to be encrypted before comparing to the query results from the SQLites (or you can decrypt the query results before comparing)
  - When display the user information, the username needs to be decrypted before displaying.
- See lect16/ExampleCrapsPartEncryptDBVer3.zip for example.
- Notice that Encryption.py file here is slightly different from that in lecture 15.