CS 145 Web Dev Workshop

11/28/17

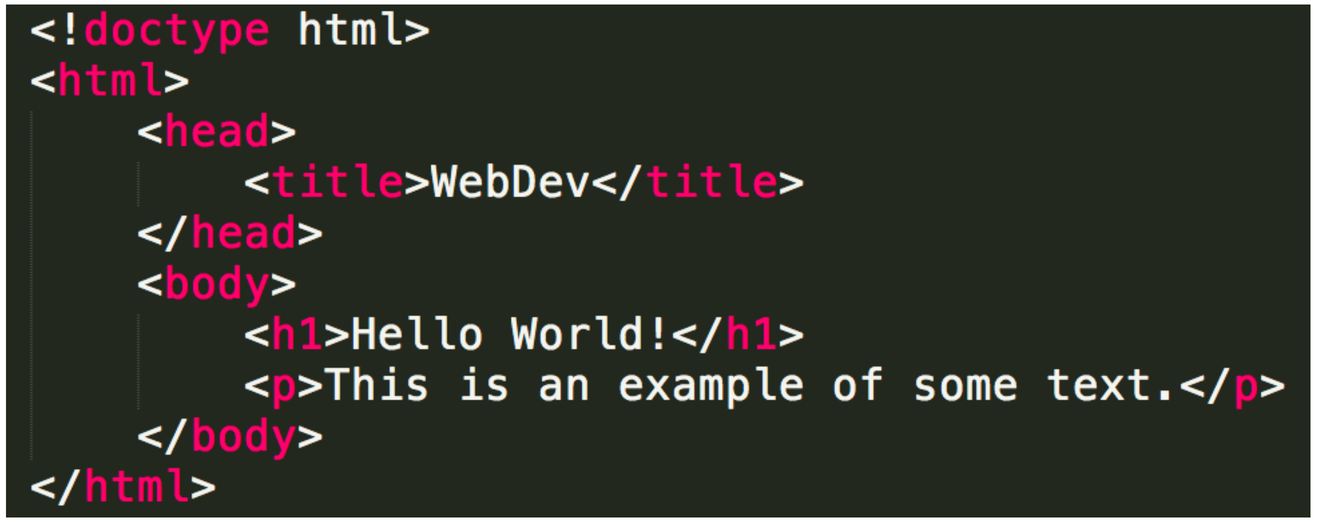
Gameplan

- HTML/CSS
- HTTP
- Routing
- Jinja
- Web.py
- Full stack example

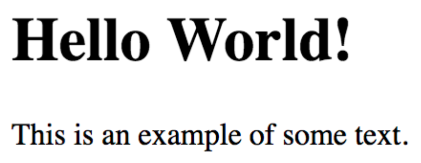
HTML

- Hyper Text Markup Language
- Used to describe structure and style of webpage
- Uses **tags** (also known as **directives)**

Sample HTML



```html
<!doctype html>
<html>
    <head>
        <title>WebDev</title>
    </head>
    <body>
        <h1>Hello World!</h1>
        <p>This is an example of some text.</p>
    </body>
</html>
```



Hello World!

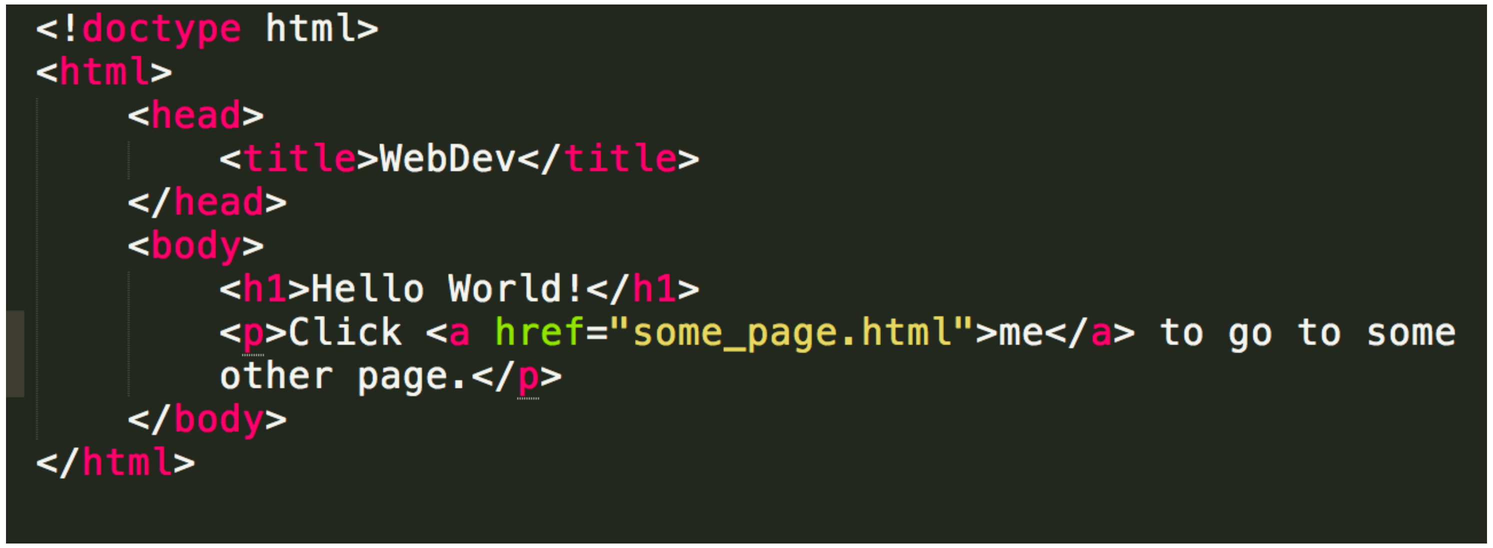
This is an example of some text.

Important HTML Tags

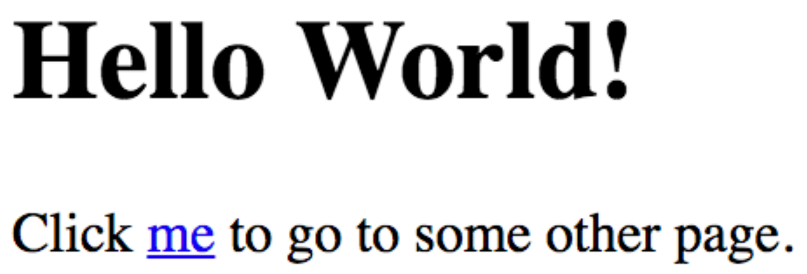
- **<div>:** Defines a section in a document (extends to end of page)
- **<span>:** Defines a section in a document (is the size of elements contained)
- **<p>:** Defines a paragraph.
- **<form>:** Defines an HTML form.
- **<a>:** Allows for linking.

Links

- Use **<a>** tag with href **attribute**



```
<!doctype html>
<html>
    <head>
        <title>WebDev</title>
    </head>
    <body>
        <h1>Hello World!</h1>
        <p>Click <a href="some_page.html">me</a> to go to some
        other page.</p>
    </body>
</html>
```



Hello World!

Click me to go to some other page.

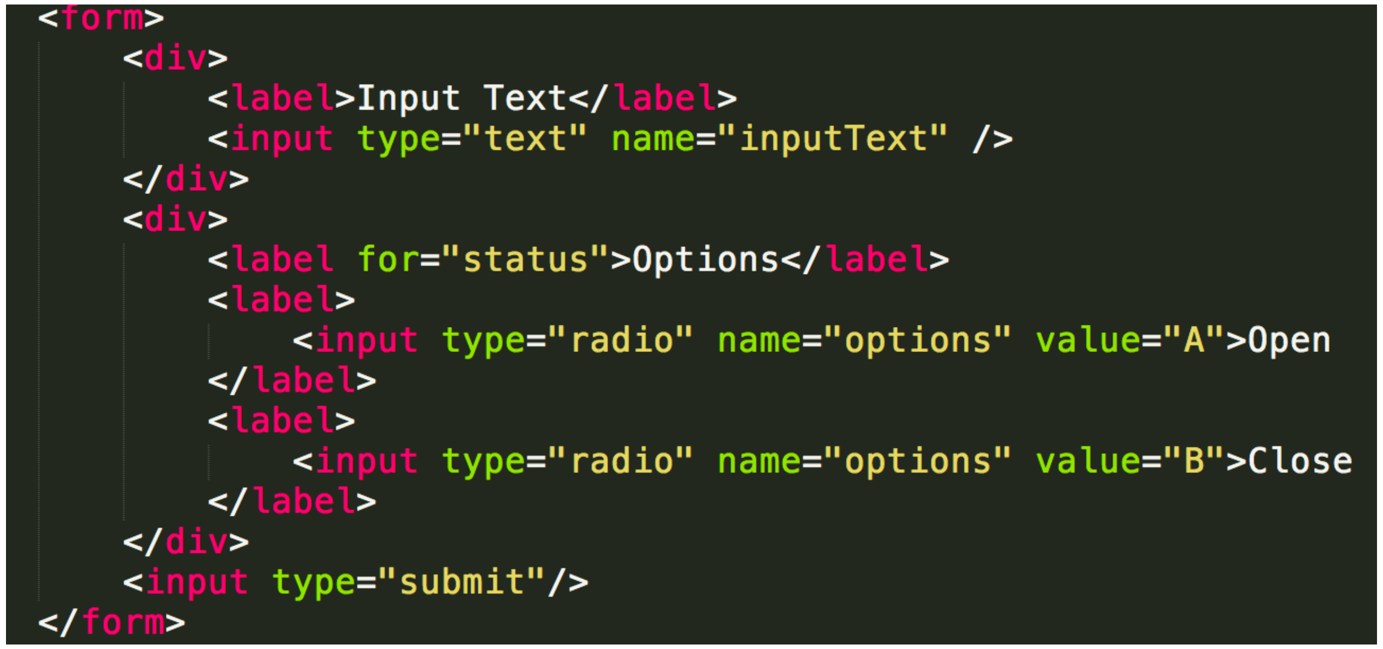
Links

- **Relative link:** <a href=“somePage.html”>
- **Absolute link:** <a href=“http://www.google.com">
- For this assignment: use **relative links!!!**

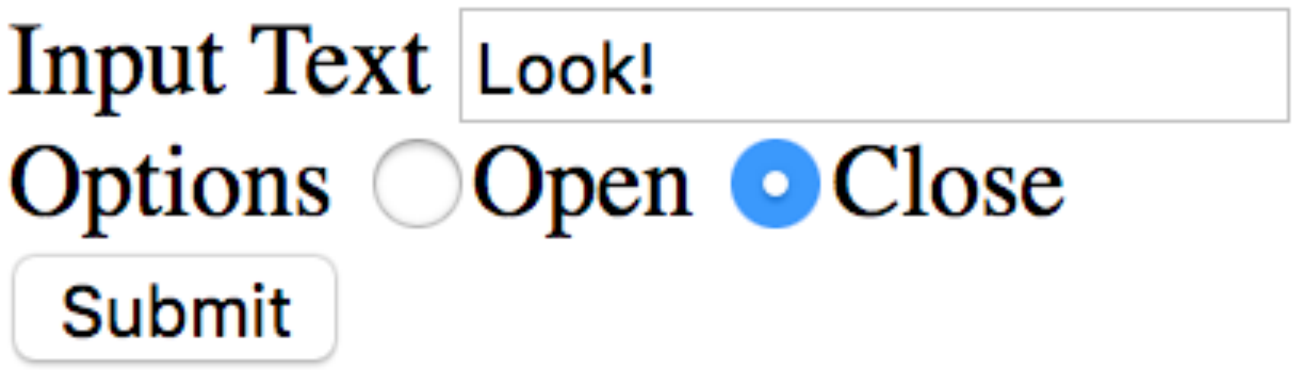
Forms

- Allows user to **input** data
- We can process the data on the **backend**
- **backend:** interacts with database, performs other computations
- Useful for: inserting bids, search

Sample Form



```html
<form>
    <div>
        <label>Input Text</label>
        <input type="text" name="inputText" />
    </div>
    <div>
        <label for="status">Options</label>
        <label>
            <input type="radio" name="options" value="A">Open
        </label>
        <label>
            <input type="radio" name="options" value="B">Close
        </label>
    </div>
    <input type="submit"/>
</form>
```



Input Text: Look!

Options: Open [✓ Close]

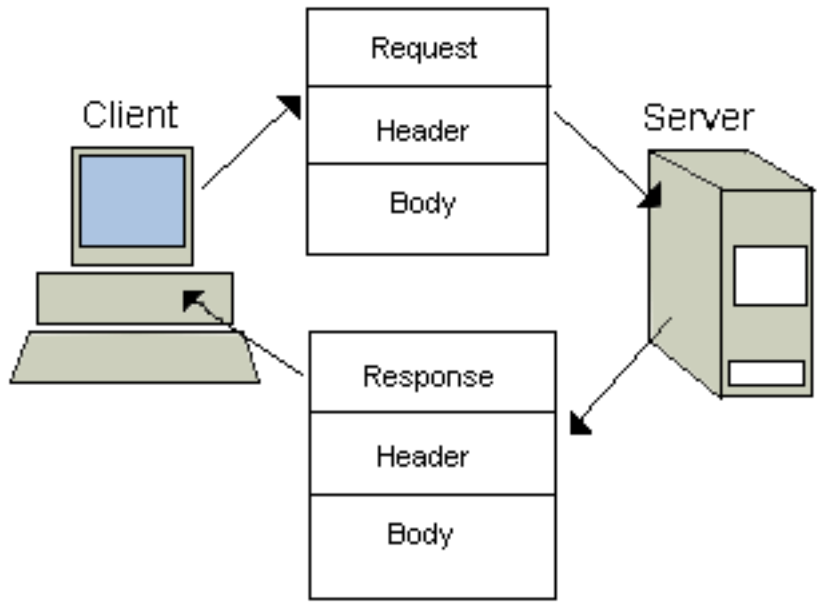
How do we actually use input information? Stay tuned.

CSS

- Cascading Style Sheets
- Used to separate element styling from HTML elements
- Fun properties including colors, positioning etc…
- Not necessary for this assignment! 💩

HTTP

- Hyper Text Transfer Protocol
- Application layer protocol: used to share resources on the internet
- Methods: GET, POST, others..



HTTP Request Methods

- **GET:** Requests a resource from the server
- **POST:** Requests a resource from the server. Encloses information in the body of the request.
- POST requests: submitting web data, adding items to database
- Others: PUT, DELETE, TRACE, HEAD… not used in this assignment!

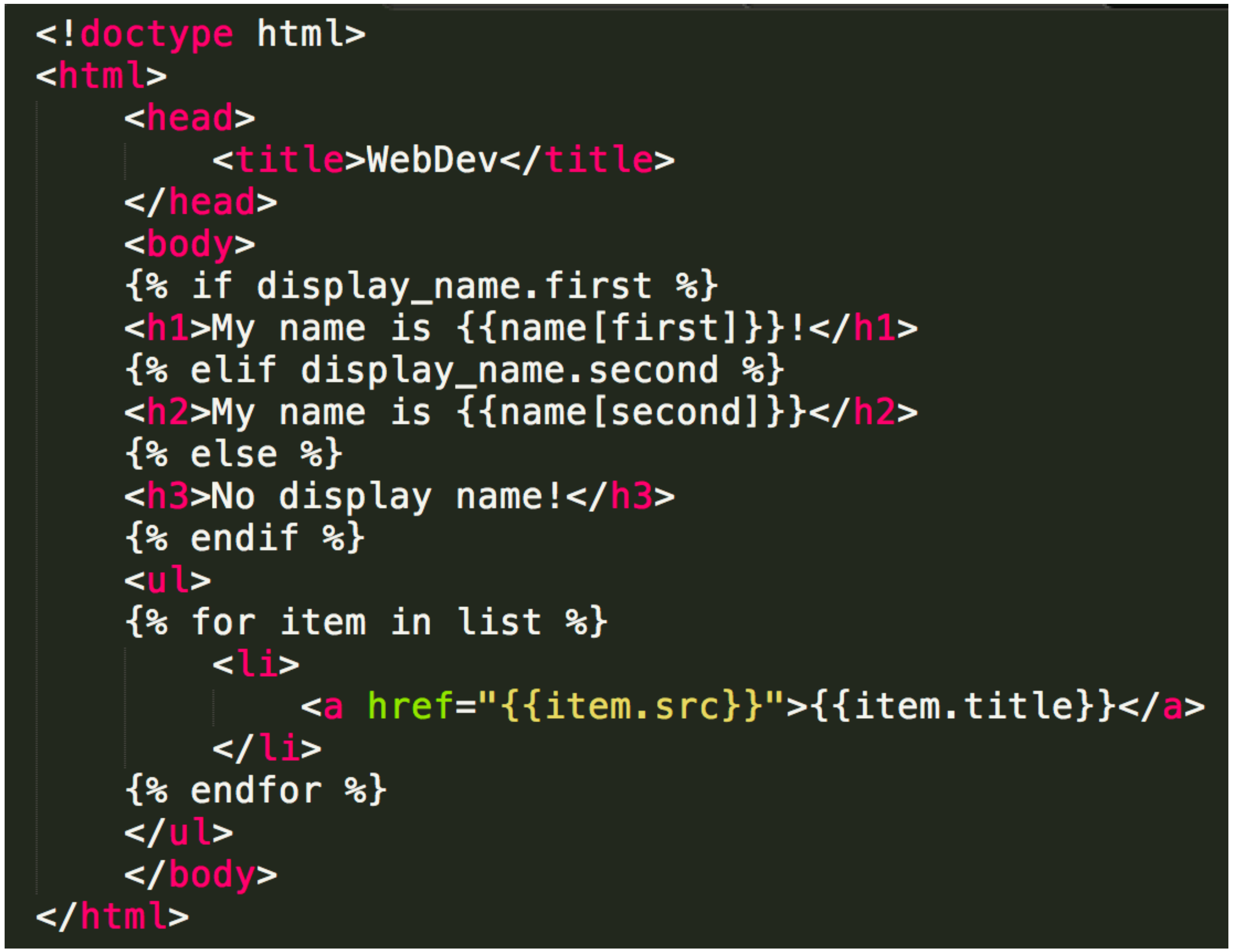
Routing

- Maps URLs to web server functionality
- **Old:** “If I go to mywebsite.com/page1.html, load page1’s HTML”
- **New:** “If I navigate to mywebsite.com/page1, do some processing on the server side and load custom elements

Jinja

- Mixture of **HTML** and **Python**
- Allows to fill out **templates**, use **for loops**, **if statements**
- Very similar to AngularJS

Jinja Templating



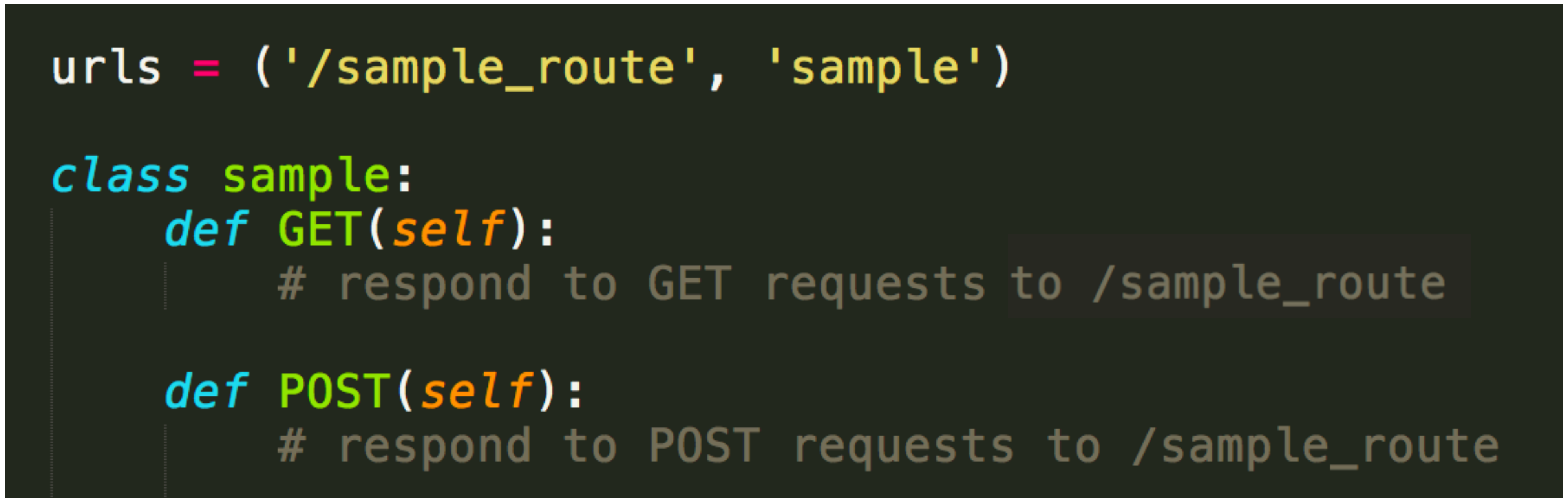
```
<!doctype html>
<html>
    <head>
        <title>WebDev</title>
    </head>
    <body>
    {% if display_name.first %}
    <h1>My name is {{name[first]}}!</h1>
    {% elif display_name.second %}
    <h2>My name is {{name[second]}}</h2>
    {% else %}
    <h3>No display name!</h3>
    {% endif %}
    <ul>
    {% for item in list %}
        <li>
            <a href="{{item.src}}">{{item.title}}</a>
        </li>
    {% endfor %}
    </ul>
    </body>
</html>
```

Jinja Templating

- Where do the Jinja templates get the values of the variables from?
- For us: parameters passed to web.py routes

Web.py

- Allows a Python class based system of routing



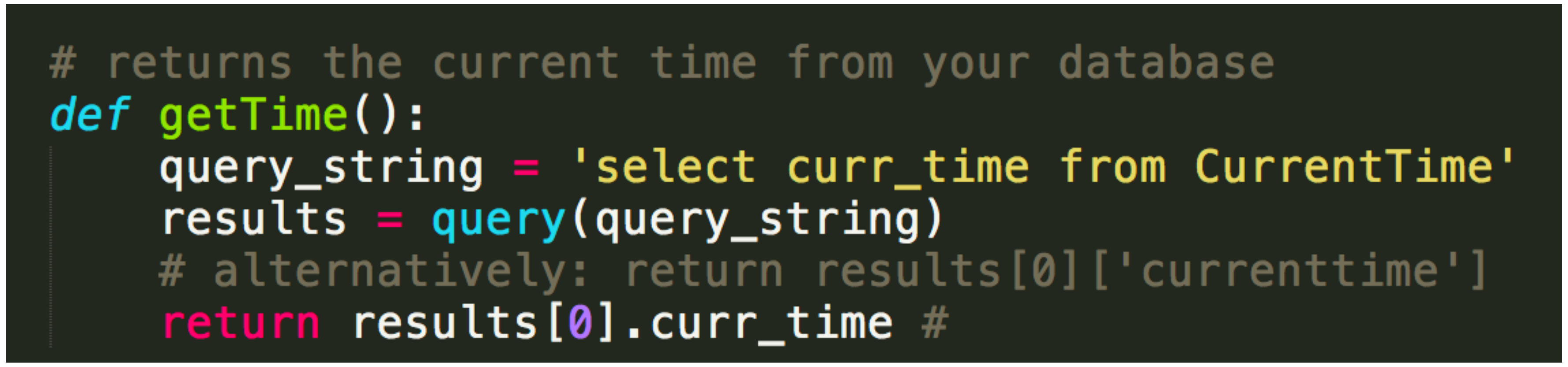
```
urls = ('/sample_route', 'sample')

class sample:
    def GET(self):
        # respond to GET requests to /sample_route

    def POST(self):
        # respond to POST requests to /sample_route
```

sqlitedb.py

- Allows for python interaction with sqlite database
- Uses try-catch syntax to allow you to commit and rollback transactions
- Returns python lists from SQL queries.



```python
# returns the current time from your database
def getTime():
    query_string = 'select curr_time from CurrentTime'
    results = query(query_string)
    # alternatively: return results[0]['currenttime']
    return results[0].curr_time #
```

(Almost) Full Stack Example

Implementation Steps

For each task we want you to implement:

- Write the helper functions you'll need in **sqlitedb.py** to query your database
- Write the python class and set up the corresponding route in **auctionbase.py.**
- Create a **template html file** using Jinja and the object you passed in from the web.py route

Questions?