Web Develop in Flask

Jim Yeh <lemonlatte@gmail.com>

Outline

- *Introduce internet network*
- *What is HTTP request and response*
- *Flask*
- *Write a Todo list*
- *http://tinyurl.com/nccumath-flask*

Internet

- *People use browsers to surf the internet.*
- *You may hear that a server is crashed.*

http://www.

Given a URL

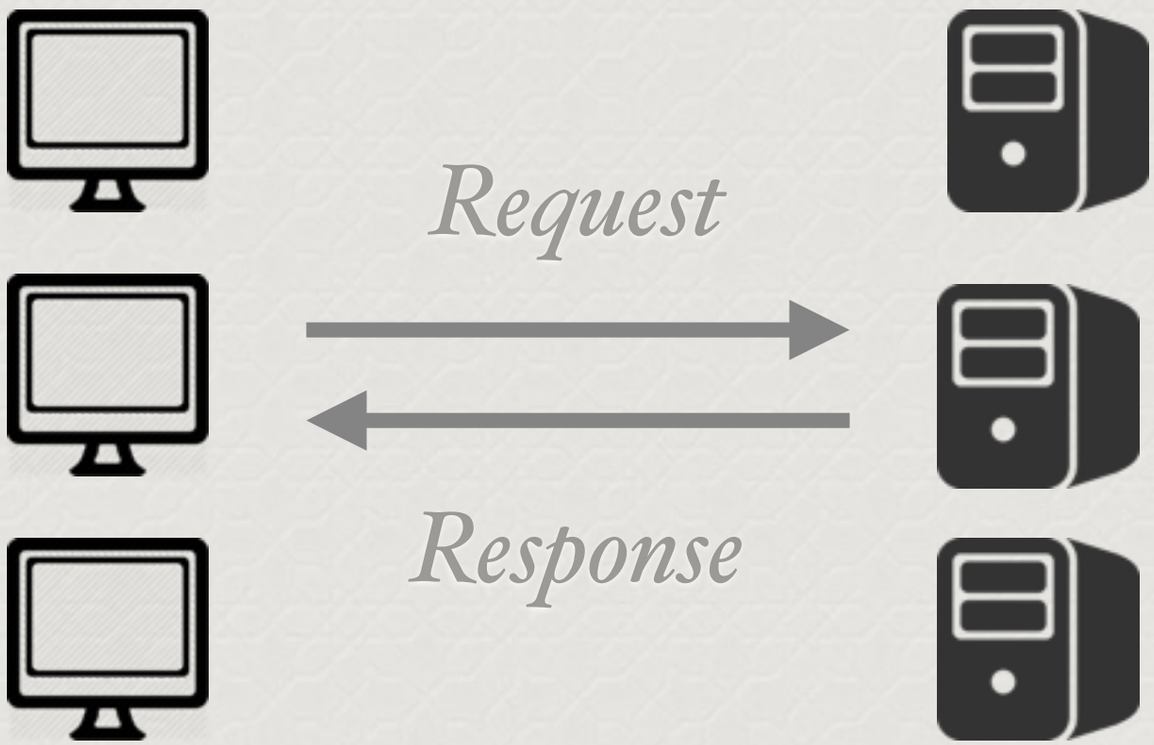
http://www.bing.com/search?q=math

Protocol *IP / Hostname* *Path* *Query String*

HTTP Protocol

- *A communication protocol*
- *A well-known client is **web browser***
- ***Not merely for browser***

Request and Response



HTTP Request Methods

- *GET*
- *POST*

GET

- *Browser a website by url*
- *Request resources (contents)*
- *Example: http://www.bing.com/search?**q=math***

POST

- *Generally, we use it to submit a form*
- *Wrap data in request body*

Response

- *Roughly speaking, it is the contents on a browser.*

IP / Hostname

- *Address of a server*
- *You can register a Hostname (domain name)*
- *Hostname will be resolve to a IP address by DNS*
- *Where a request will be delivered to*

Some Special IPs

- *127.0.0.1 : loopback address*
- *0.0.0.0 : Network traffic from any valid IP address*
- *192.168.x.x : Preserved private IP address*

Path

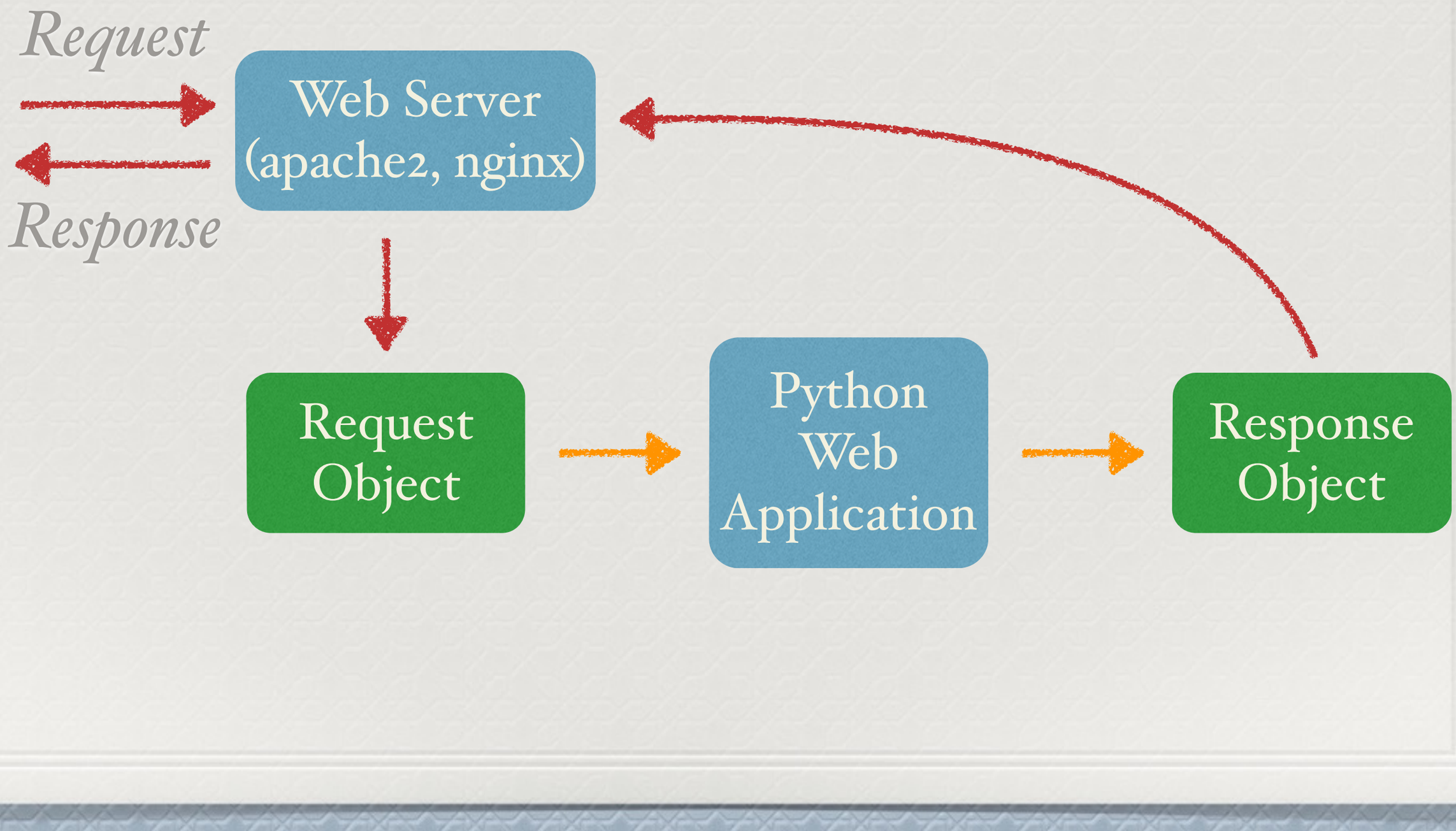
- *Location of a file*
- *Purpose of your request*

Query String

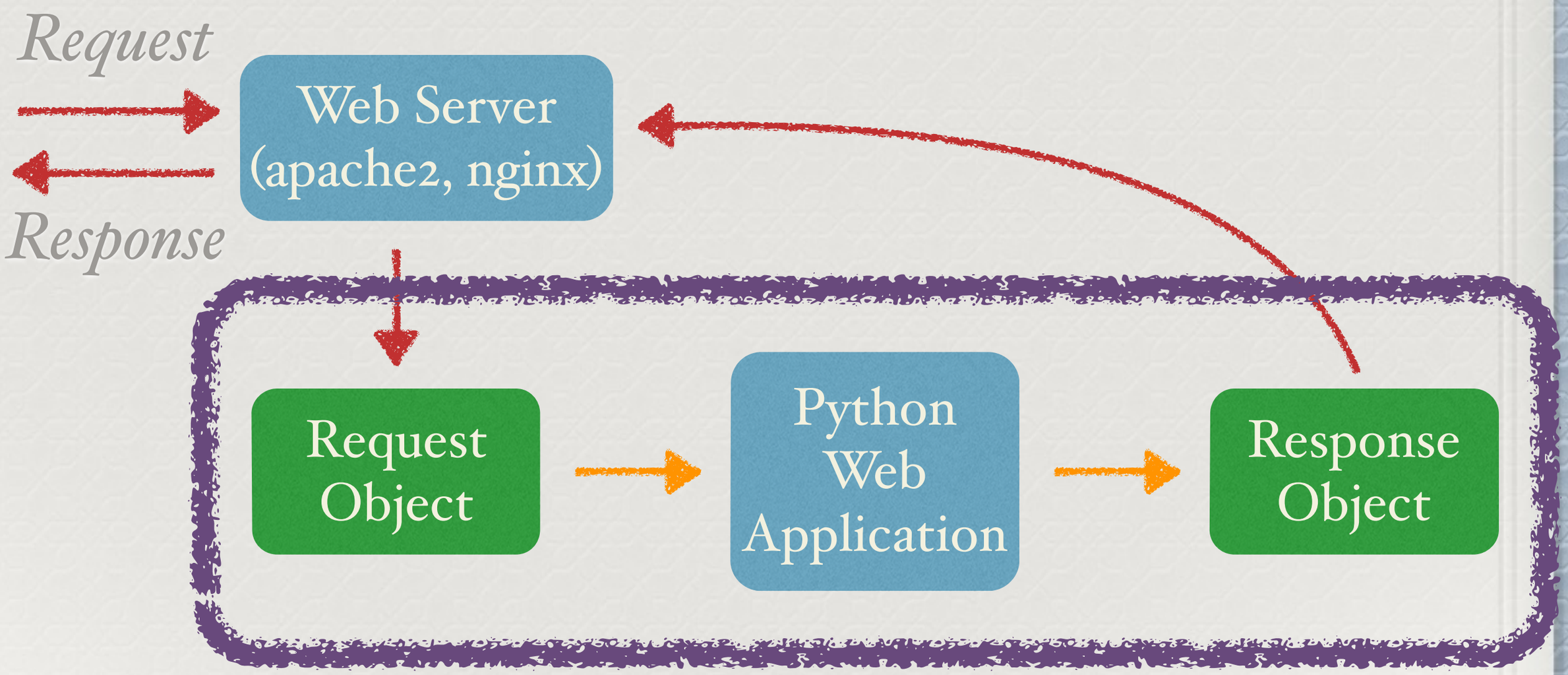
- *Some information that you provided.*

Server Side

Inspect Web Server



Inspect Web Server



What we focus on

Web frameworks

- *Django*
- *Flask*
- *Web.py*
- *Many...*

Flask

Requirements

- *python*
- *flask*
- *sqlalchemy*

First App

server.py

```python
from flask import Flask


app = Flask("todo-list")
app.debug = True # Enable debug mode


@app.route("/")
def hello():
    return "Hello World!"


if __name__ == "__main__":
    app.run(host="0.0.0.0")
```

First App

server.py

```
1  from flask import Flask
2
3
4  app = Flask("todo-list")
5  app.debug = True # Enable debug mode
6
7
8  @app.route("/")
9  def hello():
10     return "Hello World!"
11
12
13 if __name__ == "__main__":
14     app.run(host="0.0.0.0")
```

Create an app

First App

server.py

```
1  from flask import Flask
2
3
4  app = Flask("todo-list")
5  app.debug = True # Enable debug mode
6
7
8  @app.route("/")
9  def hello():
10     return "Hello World!"
11
12
13 if __name__ == "__main__":
14     app.run(host="0.0.0.0")
```

Create an app

Enable Debug Mode

First App

server.py

```
1  from flask import Flask
2
3
4  app = Flask("todo-list")
5  app.debug = True # Enable debug mode
6
7
8  @app.route("/")
9  def hello():
10     return "Hello World!"
11
12
13 if __name__ == "__main__":
14     app.run(host="0.0.0.0")
```

Create an app

Enable Debug Mode

Assign path

First App

server.py

```
1  from flask import Flask
2
3
4  app = Flask("todo-list")
5  app.debug = True # Enable debug mode
6
7
8  @app.route("/")
9  def hello():
10     return "Hello World!"
11
12
13 if __name__ == "__main__":
14     app.run(host="0.0.0.0")
```

Create an app

Enable Debug Mode

Assign path

View function

First App

server.py

```
1  from flask import Flask
2
3
4  app = Flask("todo-list")
5  app.debug = True # Enable debug mode
6
7
8  @app.route("/")
9  def hello():
10     return "Hello World!"
11
12
13 if __name__ == "__main__":
14     app.run(host="0.0.0.0")
```

Create an app

Enable Debug Mode

Assign path

View function

Start the server

First App

server.py

```python
from flask import Flask


app = Flask("todo-list")
app.debug = True # Enable debug mode


@app.route("/")
def hello():
    return "Hello World!"


if __name__ == "__main__":
    app.run(host="0.0.0.0")
```

Run

- *python server.py*

Inspect Web Application

Inspect Web Application

Dispatch path

Request Object

Run View Function

Response Object

Return value(string)

What's different between this and traditional web pages?

Interact with end-users

Case 1 : Say Hi

Hi, Jim

Use GET

- *http://localhost:5000/?name=Jim*
- *Use query string in your view function*

Exercise I

- *Given a dictionary:*

  {“apple”: “a fruit”, “calculus”: “a subject”, ... and more}

- *Query it by GET method*

- *Display “Not found” if the key is unavailable*

Exercise I.I

- *Design a web app that accept multiple query strings.*

Once the length of return string becomes longer and longer...

The function is not readable

```
1  from flask import Flask
2  from flask import request
3
4  app = Flask("todo-list")
5  app.debug = True
6
7
8  @app.route("/")
9  def hello():
10     return ("Hello World! Hello World! Hello World! "
11             "Hello World! Hello World! Hello World! "
12             "Hello World! Hello World! Hello World! "
13             "Hello World! Hello World! Hello World! "
14             "Hello World! Hello World! Hello World! "
15             "Hello World! Hello World! Hello World! "
16             "Hello World! Hello World! Hello World! ")
```

Is there a way to move contents out?

Is there a way to move contents out?

A: Use Template (jinja2)

Create a templates

- *Create a "**templates**" folder* <= Very important
- *Create a txt file (hello.txt) in templates folder*
- *Write contents in this file*

Use template in view function

- from flask import render_template
- render_template(“hello.txt”)

```
@app.route("/hello")
def hello_many():
    return render_template("hello.txt")
```

Not merely a text

Format your content using HTML

HTML/HTML5

- *A markup language*
- *Contents of a web page*

A HTML Element

- <tag_name attr1="value" attr2="value"> content
  </tag_name>
- This is a book
- <h1> This a title </h1>

Some Elements

- …
- <h1> … <h1> to </h5> … </h5>
- …
-
   item
   item

-
   item
   item

HTML Layout

```
<html>
    <head>
        <style>
            /* Some CSS style */
        </style>
        <script>
            // Some javascript scripts
        </script>
    </head>
    <body>
        <h1>Welcome, Jim.</h1>

        <p>This is my first app using HTML.</p>
    </body>
</html>

```

HTML Layout

HTML Block

```
<html>
    <head>
        <style>
            /* Some CSS style */
        </style>
        <script>
            // Some javascript scripts
        </script>
    </head>
    <body>
        <h1>Welcome, Jim.</h1>

        <p>This is my first app using HTML.</p>
    </body>
</html>

```

HTML Layout

HTML Block

Head

```
<html>
    <head>
        <style>
            /* Some CSS style */
        </style>
        <script>
            // Some javascript scripts
        </script>
    </head>
    <body>
        <h1>Welcome, Jim.</h1>

        <p>This is my first app using HTML.</p>
    </body>
</html>

```

HTML Layout

HTML Block

```
<html>
    <head>
        <style>
            /* Some CSS style */
        </style>
        <script>
            // Some javascript scripts
        </script>
    </head>
    <body>
        <h1>Welcome, Jim.</h1>

        <p>This is my first app using HTML.</p>
    </body>
</html>

```

Head

Body

HTML Layout

templates/
index.html

```
<html>
    <head>
        <style>
            /* Some CSS style */
        </style>
        <script>
            // Some javascript scripts
        </script>
    </head>
    <body>
        <h1>Welcome, Jim.</h1>

        <p>This is my first app using HTML.</p>
    </body>
</html>

```

Rewrite hello function

- *Again, use the HTML template that you just finished.*

```
@app.route("/")
def hello():
    return render_template("index.html")
```

What about variables?

What about variables?

A: Code in Template

Variable Substitution

- *You can substitute variables in template files*
- {{ name }} *in HTML template*

```
<body>
    <h1>Hi, {{ name }}.</h1>

    <p>This is my first app using HTML.</p>
</body>
```

Transfer variables to template

- *Use keyword arguments in render_template*
- render_template(“hi.html”, name=name)

```
@app.route("/hi")
def hi():
    name = request.args.get("name", "Someone")
    return render_template("hi.html", name=name)
```

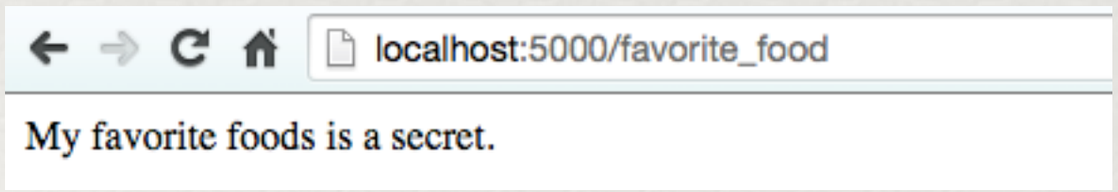
For, If Statement

- {% if name == “Jim” %} ..... {% endif %}
- {% for item in things %} {% endfor %}

Case Study: Food List

- *Display a list of your favorite food if it is public*

- *Otherwise, display a message*



More syntax

- *http://jinja.pocoo.org/docs/dev/templates*

Exercise II

- *Rewrite your dictionary app by HTML template.*

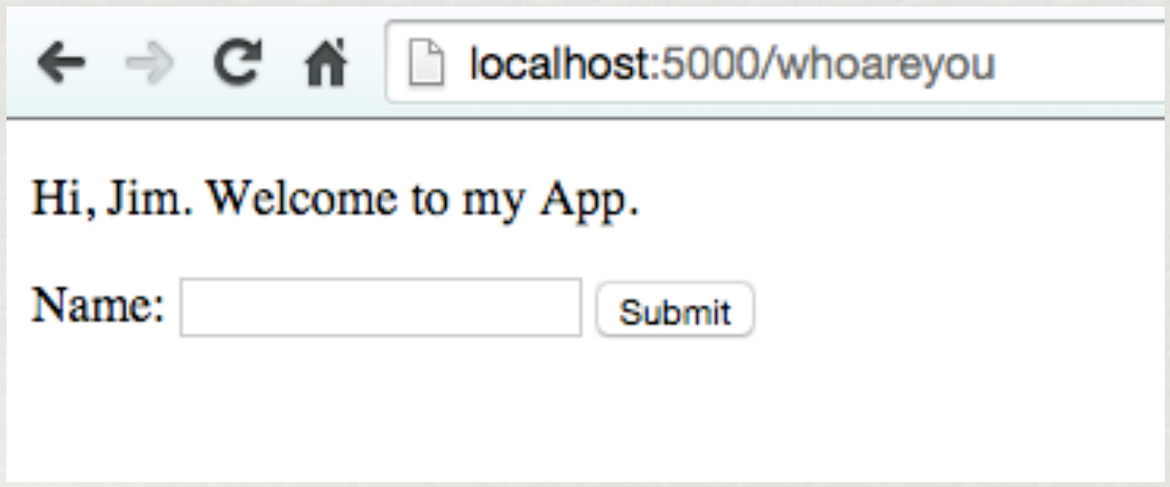
Submit a Form

HTML Form

- *Form elements*
  - form - a block of a form
  - input - input fields
  - label - label for an input

Case: Who are you?

- *Ask for a name*
- *Once a name is submitted, Display the welcome message.*



HTML for whoareyou

Where the form request to

Method for the form request

```
<body>
    <p>{% if name %} Hi, {{ name }}. Welcome to my App. {% endif %}</p>

    <form action="/whoareyou" method="POST">
        <label for=""> Name:
            <input type="text" name="name" value="">
        </label>
        <input type="submit" value="Submit">
    </form>
</body>
```

view function for whoareyou

Allow method for POST

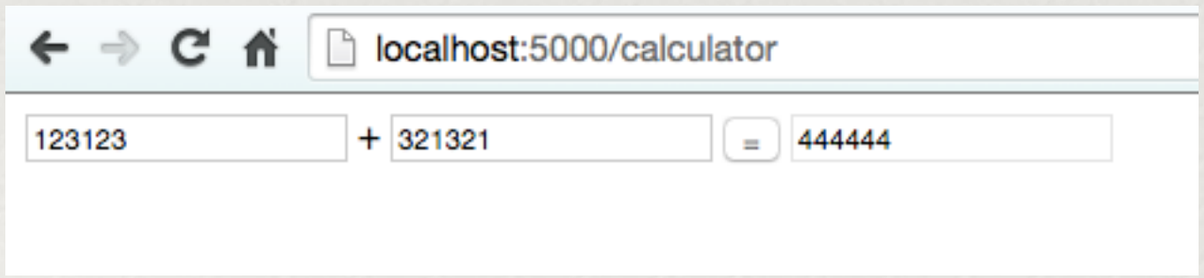
```python
@app.route("/whoareyou", methods=["GET", "POST"])
def whoareyou():
    if request.method == "POST":
        name = request.form.get("name")
        return render_template("form.html", name=name)

    if request.method == "GET":
        return render_template("form.html")
```

Handle POST request

Handle GET request

Exercise III: Calculator

- *Two input fields and a submit*
- *Display the result*



To-Do List

Views

- *List all items*
- *Add a new item*
- *Delete an item*

Flask redirect

- *redirect*
- *url_for*

Save to file

- *Use pickle module (pickle.load and pickle.dump)*
- *pickle.load(open("todo_list.data", "r"))*
- *pickle.dump(dictionary, open("todo_list.data", "r"))*

Questions