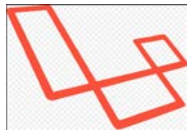


Building the Controller using a Web Framework

django



Topics covered Last Lecture

- Python Language
- Object oriented programming
- Organizing code in Python

Quiz

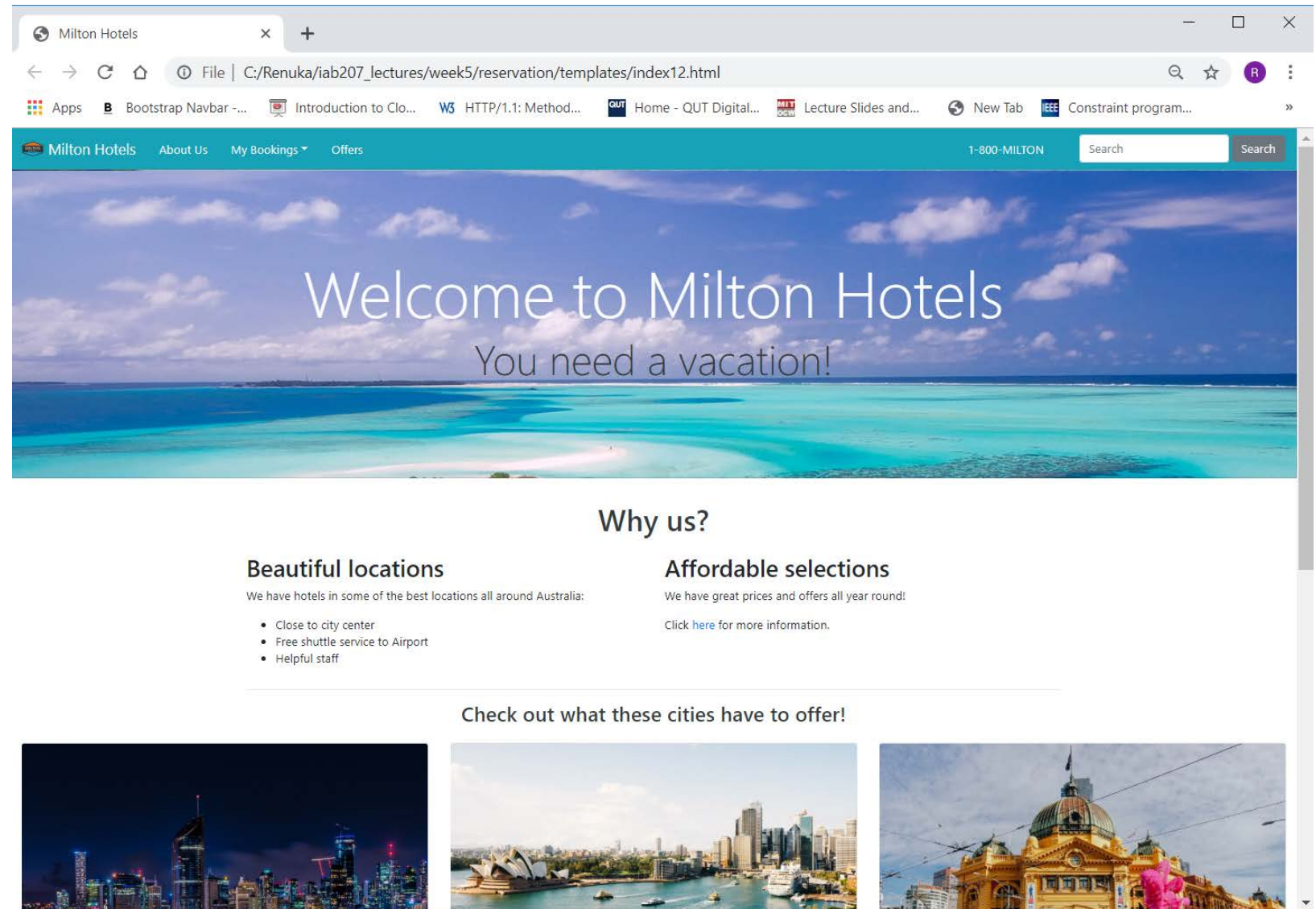
- <https://tinyurl.com/iab207s2w5>

Aims of this lecture

- Web application protocol
- Frameworks and their functions
- The **Flask** framework

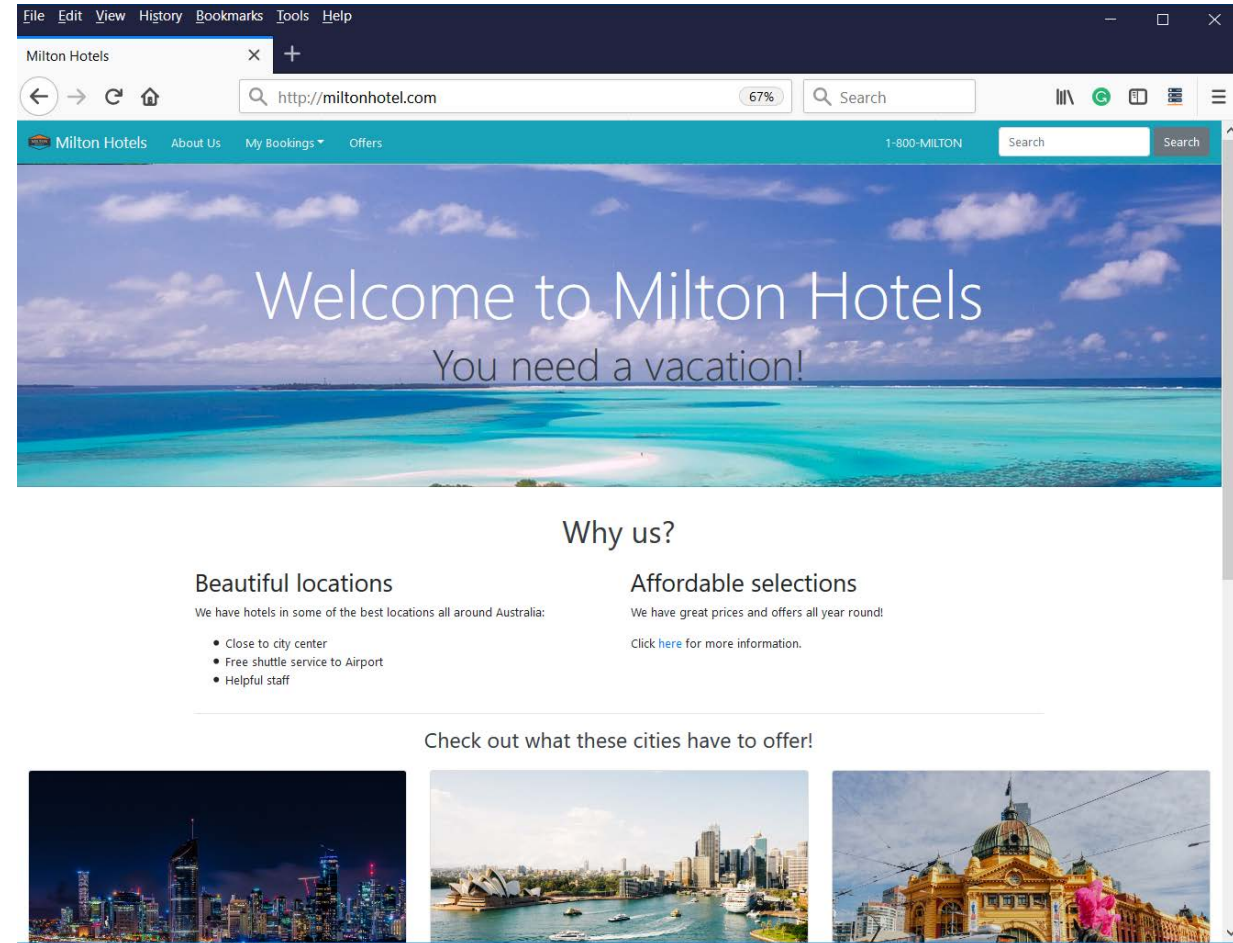


What we've done so far...



Accessing a web application/page/resource

http://<something>.com.au



http – Hypertext Transfer Protocol

Protocol for Web applications

- Defines the rules/is a communication standard to transfer hypertext between two computer machines.
- HTTP (Request/Response Protocol)

Hypertext Transfer Protocol
- Hypertext : Text with links to other text
 - Human readable and not binary protocol

HTTP Specification

- Request message sent from the client to the server
 - what is requested, to whom it is requested, additional information
- Response messages
 - Status information, Message
- Connection: Stateless protocol
 - No memory of any other request made in the past

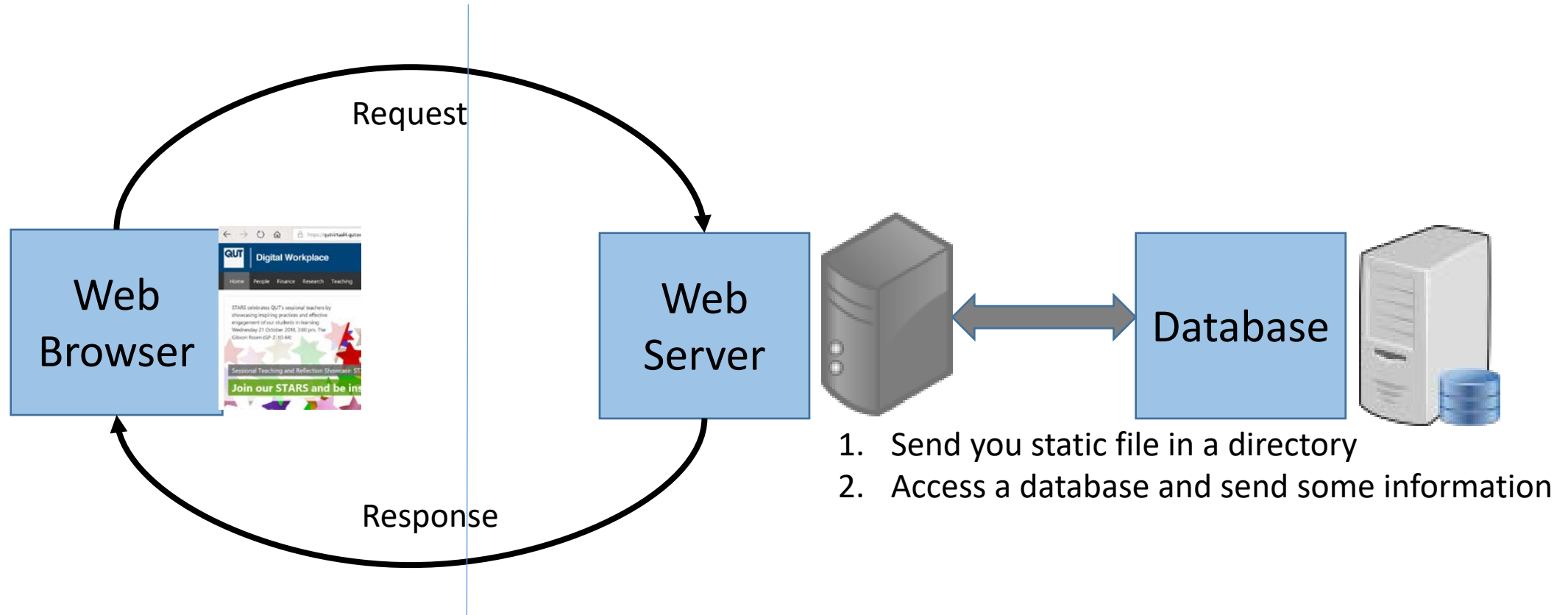
Protocols used: HTTP, TCP, IP and Link

- Many protocols or standards are used when using a web application



1. HTTP (Hyper text Transfer Protocol): What should the message contain?
2. TCP (Transmission Control Protocol) : How can the message be sent reliably?
3. IP (Internet Protocol): Where should the message be sent?
4. Link (Physical Link): Send the message using Fibre link, WiFi

Web application



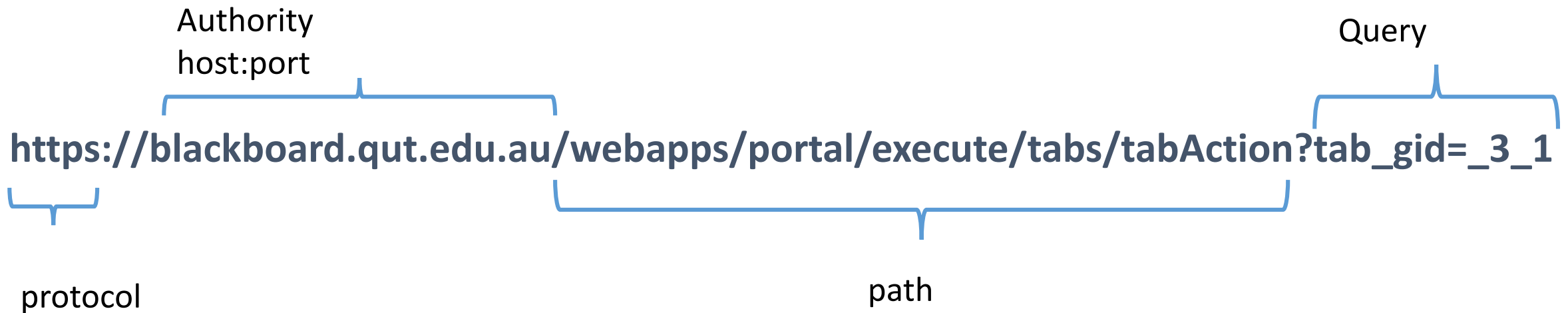
- Web browser **sends a request** to the server
- The Web server **sends a response** to the browser
- A request is always followed by a response

URL – Send request to a Server

- Uniform Resource Locator (URL)
- Human readable address
 - Where on the web the request needs to be sent
- Used by other communication protocols
 - Example – File transfer protocol (ftp)

URL – Uniform Resource Locator

- URL (is a Uniform Resource Identifier - URI)



```
URI = scheme:[//authority]path[?query][#fragment]
```

URI specification

What does the request-response pair contain?

Request to Web application — <http://www.google.com>

Request URL: <https://www.google.com/>

Request method: GET

Remote address: 172.217.25.164:443

Status code: ● 200 OK

Version: HTTP/2.0

Request headers:

Host: www.google.com

User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.9.0) Gecko/52.9.0 Firefox/52.9.0; ADSSO

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: en-AU,en-US;q=0.7,en;q=0.3

Accept-Encoding: gzip, deflate, br

Cookie:

NID=154=qK7h7FhD6ulWT9etjOSltOO4nbsU3Epa0VCml70GRJMaDxrjQYMA9tNG7Hy81HCfs
GHSxoga3Z0VKW0KMYFFyKZabJZrLINJYqnQIZw6Q8D7oEwEnQHLv0zwuh1tp3V_rHOvXaE-
fhUXUNxS09XerrgGan5x2pN2TEbSp_nL0; 1P_JAR=2019-01-12-00; OGP=-5061451:

Connection: keep-alive

Upgrade-Insecure-Requests: 1

Cache-Control: max-age=0

Request Line : URI, Request method

Request Headers

(key-value pairs)

Response from Web application

▼ Response Headers

alt-svc: quic=":443"; ma=2592000; v="44,43,39,35"

cache-control: private, max-age=0

content-encoding: br

content-length: 70592

content-type: text/html; charset=UTF-8

date: Mon, 17 Dec 2018 04:13:18 GMT

expires: -1

server: gws

status: 200

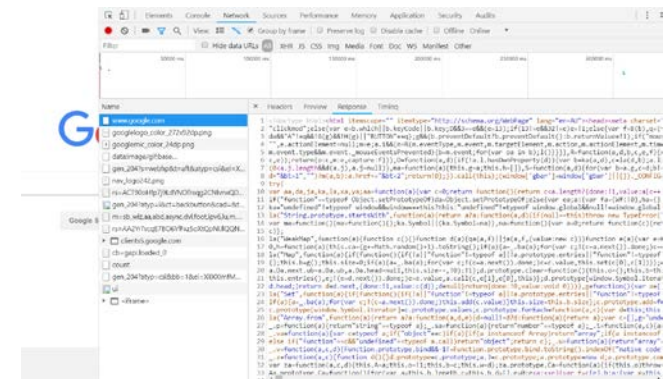
x-frame-options: SAMEORIGIN

x-xss-protection: 1; mode=block

Response Headers

(key-value pairs)

HTML page as response



Request with parameters




- Resource path **/search**
- Query parameters that includes **?q=QUT**
- Multiple parameters are passes with a separator **&**

Status code from the server

- 10X – Information
- 20X – Success
 - 200 – OK
 - 201 - Created
 - 204 – No content
- 30X – Redirect
 - 301 – Moved permanently
 - 307 – Temporary redirect
- 40X – Failure codes
 - 400 – Bad request
 - 401 – Unauthorized
 - 403 - Forbidden
 - 404 – Not found
- 50X – Server failure errors
 - Something went wrong at the server
 - All errors should be handled with a well formed response

Look at another request

← → ↻ 🏠 🔒 https://validator.w3.org/#validate_by_input 🔍 ☆ 👤



Markup Validation Service

Check the markup (HTML, XHTML, ...) of Web documents

Validate by URI

Validate by File Upload

Validate by Direct Input

Validate by direct input

Enter the Markup to validate:

```
<!doctype html>
<html>

<head>
<title>MyPage</title>
</head>
</html>
```

▶ More Options

Check

This validator checks the [markup validity](#) of Web documents in HTML, XHTML, SMIL, MathML, etc. If you wish to validate specific content such as [RSS/Atom feeds](#) or [CSS stylesheets](#), [MobileOK content](#), or to [find broken links](#), there are [other validators and tools](#) available. As an alternative you can also try our [non-DTD-](#)

Name	× Headers Preview Response Timing
<input type="checkbox"/> check	<div>▼ General</div> <div>Request URL: https://validator.w3.org/nu/</div> <div>Request Method: POST</div> <div>Status Code: ● 200</div> <div>Remote Address: 128.30.52.39:443</div> <div>Referrer Policy: no-referrer-when-downgrade</div>
<input type="checkbox"/> nu/	<div>▶ Response Headers (13)</div> <div>▶ Request Headers (14)</div> <div>▼ Form Data view source view URL encoded</div> <div>fragment: <html></div> <div><body></div> <div><p>My first page</p></div> <div></body></div> <div><html></div> <div>prefill: 0</div> <div>doctype: Inline</div> <div>prefill_doctype: html401</div> <div>group: 0</div>
<input type="checkbox"/> style.css	
<input type="checkbox"/> script.js	

- URL remains same

- The content is sent in Form Data

HTTP Request and Response

- **HTTP Request:**

- ***Request Line*** indicating the method (**GET/POST/PUT/DELETE**)
- ***URI – Resource***: Identifies the server and resource on the server
- Additional information about the web client (Request ***headers***)
- Optional ***message body*** (used during POST)

- **HTTP Response:**

- ***Status code*** – indicating if all is right/ something went wrong
- Additional information from server (***headers***)
- ***Message body*** (sending information)

When do you use a GET and POST?

Get and Post methods

- GET

- Use it when requesting data from the server
- Associated to performing a read operation (not alter or make changes)
- Has a limitation of how much data you can send (2048 characters)
- Sends text data as a part of the URL

- POST

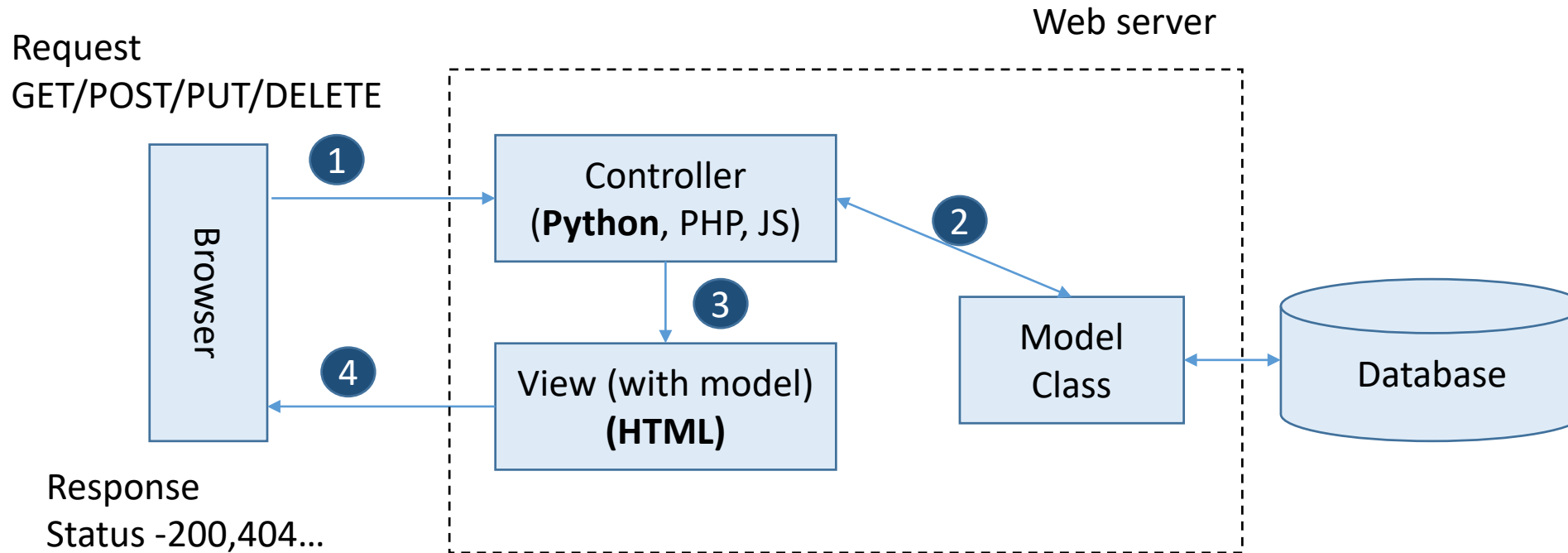
- Send files, or form inputs as a part of the request body (when large)
- Associated to a write operation
- No restrictions on data length
- Sends text or binary data as a part of the message

Request types

- Amazon.com
 - Use 'Inspect' on Chrome/Firefox
1. Search for 'Python programming'
 - Look at the URL
 - Inspect the request
 2. Select a book
 - Click on 'Buy now with one click'
 - Inspect the request

Web application request/response

Model View Controller Pattern



Frameworks

What does the framework do?

- Provides common and generic functionality
- Can be extended and customized
- Flask is a light weight python web framework
 - Often referred to as micro framework
 - Bare minimum functionality makes it very flexible
- Django is another well known web framework for python



django

Server side frameworks

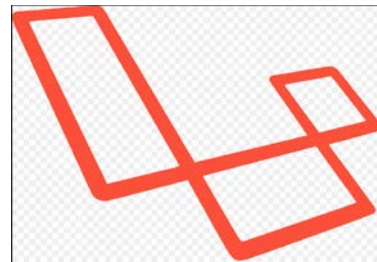
Java



Spring

Struts™

PHP



Laravel

Python

django



Flask

JavaScript



METEOR

Server Side Framework Support

- HTTP Handling
 - Parsing the request, generating response
 - Handling multiple user requests
- URL Dispatching
 - What code to execute for a URL
- Database storage
 - Relational or non-relational database
- Templating
 - Generating HTML with dynamic content
- User authentication

Inversion of Control

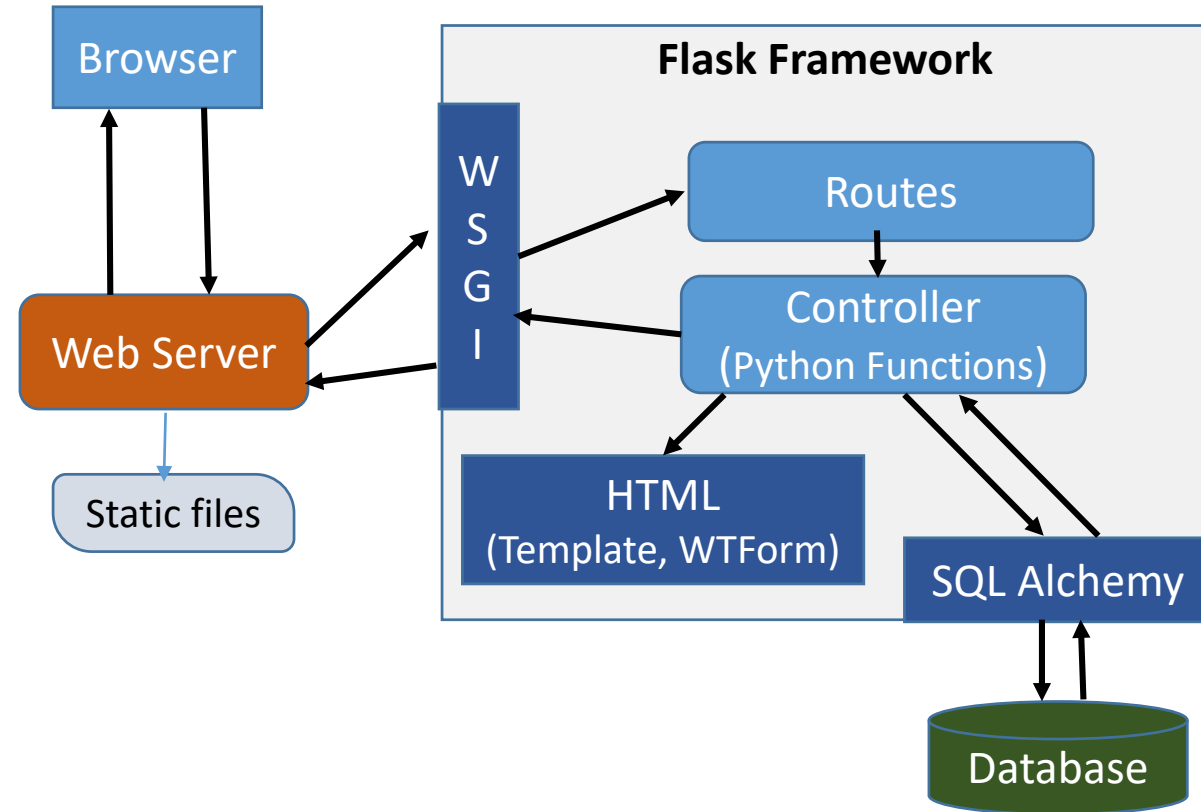
- **Library:**
 - A set of common and generic functions
 - Code using the library calls the functions
- **Framework:** Control is inverted
 - Framework code calls the code using it.
 - Hollywood principle 😊 - 'Don't call us, we will call you'

Build a simple web application using Flask

- Flask Set up
 - Install Python3
 - Install Visual Studio Code
 - Install Flask
- Virtual environment set up is optional
 - Best practice to use when developing multiple python applications
 - We will not use it in our development
- <https://code.visualstudio.com/docs/python/tutorial-flask>

Flask and its Libraries

- WSGI : Web server gateway interface (pronounced whiskey)
 - Webserver passes the request to the framework
- Jinja Template Engine
 - Generating HTML with dynamic content
- Several Flask extensions (we will use)
 - WTForms (web forms)
 - Flask-Login (user authentication)
 - Flask-SQLAlchemy (connect to database)



Flask application

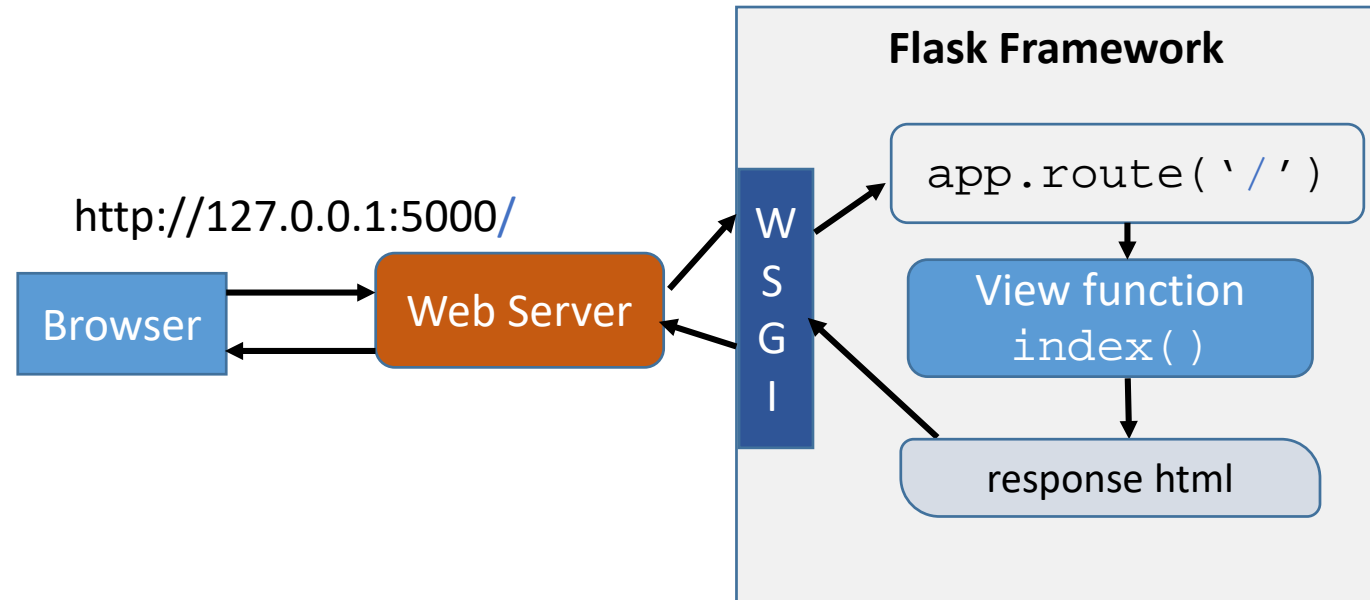
- Flask web application code and demo

```
from flask import Flask, request

#create a flask web application
app = Flask(__name__)
app.debug=True

@app.route('/')
def landing():
    return '<h1>Hello</h1>'
```


Flask App and View functions



Flask comes with a small development web server

Another way to run flask app:

Set FLASK_APP environment variable to flask1_intro

flask run

flask.exe is installed in Python\Scripts folder

Passing path parameters

```
@app.route('/<name>')  
def index(name): #view function  
    print(name)  
    return '<h1>Hello {} </h1>'.format(name)
```

Default the parameter is string.

```
@app.route('/<int:id>')  
@app.route('/<float:id>')  
@app.route('/<path:val>')
```

path – is a special string type that can have forward slash in it (used rarely)

Flask Request

```
def landing():  
    print("Printing the request headers:")  
    print(request.headers)  
    print("Printing the request arguments:")  
    print(request.args)  
    return '<h1>Hello</h1>'
```

Console output

```
Accept: text/html, application/xhtml+xml, image/jxr, */*  
Accept-Language: en-AU  
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64; ADSSO) AppleWebKit/537.36 (F  
Accept-Encoding: gzip, deflate  
Host: 127.0.0.1:5000  
Connection: Keep-Alive
```

Flask Request

- Request object is not passed in the view function but can be accessed
- If two requests are made to the web server, what does this request contain
- Flask maintains independent request information using context (mysterious object)
 - Each browser request causes a separate context
 - Request is added to context

Flask Request (contd.)

```
@app.route('/<name>') # known as path parameter
def get_name(name): #view function
    client = request.headers.get('User-Agent')
    lastname = request.args.get('lname', 'stranger')
    str= '<h3> Hi {}.{}! Your browser supports {}</h3>'.format(name, lastname, client)
    return str
```

Attribute	Description
form	A dictionary with all the form fields submitted in the request
args	A dictionary with all the arguments passed in the query string of URL
values	A dictionary that combines the values in form and args
cookies	A dictionary with all the cookies included in the request
headers	A dictionary with all the http headers included

Sessions in Flask

- What is a session

Session is created by the server

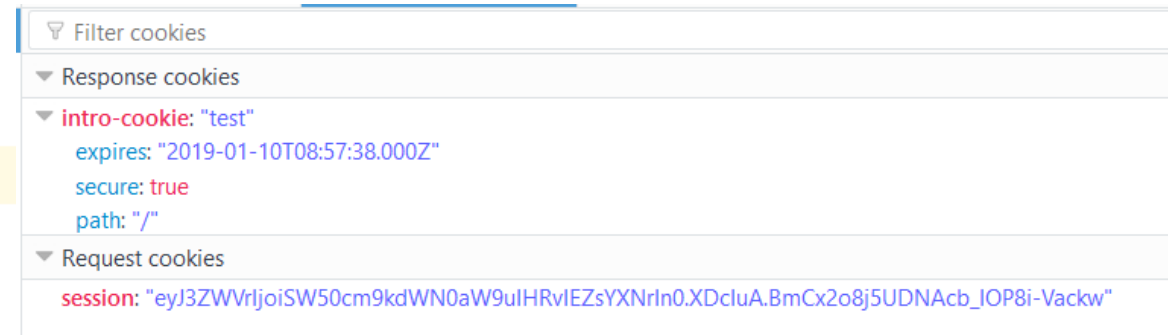
Available only till the browser is open

Encoded (not encrypted) – It cannot be modified by the client as it needs the secret key

```
@app.route('/setsession/<name>')
def set_session(name): #view function
    session['week']='Introduction to Flask'
    return '<h3>Set the session week</h3>'

@app.route('/getsession')
def get_session(): #view function
    value = session['week']
    return '<h3>Value is {}</h3>'.format(value)
```

Session viewed in a browser



Sessions in Flask

- Session encodes (not encrypts) the content
- Session data can be decoded without secret key
 - Do not store sensitive information in the session
- Session data cannot be modified as the signature requires information secret key

```
>>> import base64
>>> base64.urlsafe_b64decode('eyJ3ZWVrIjoiSW50cm9kdWN0aW9uIHRvIEZsYXNrIn0===')
b'{"week": "Introduction to Flask"}'
>>>
```

▼ Request cookies

session: "eyJ3ZWVrIjoiSW50cm9kdWN0aW9uIHRvIEZsYXNrIn0.XDcluA.BmCx2o8j5UDNAcb_IOP8i-Vackw"

If interested (optional): <https://blog.miguelgrinberg.com/post/how-secure-is-the-flask-user-session>

Rendering the HTML

- How do we use the HTML file created in Week2 and Week3
- Jinja Template engine can be used
- How does it work?
 - Templates are HTML files with extra syntax to handle data dynamically
 - Syntax for place holders in the template allow the dynamic data to be added.
 - **Rendering:** Template engine replaces the place holders with actual values and generates the final response

Example

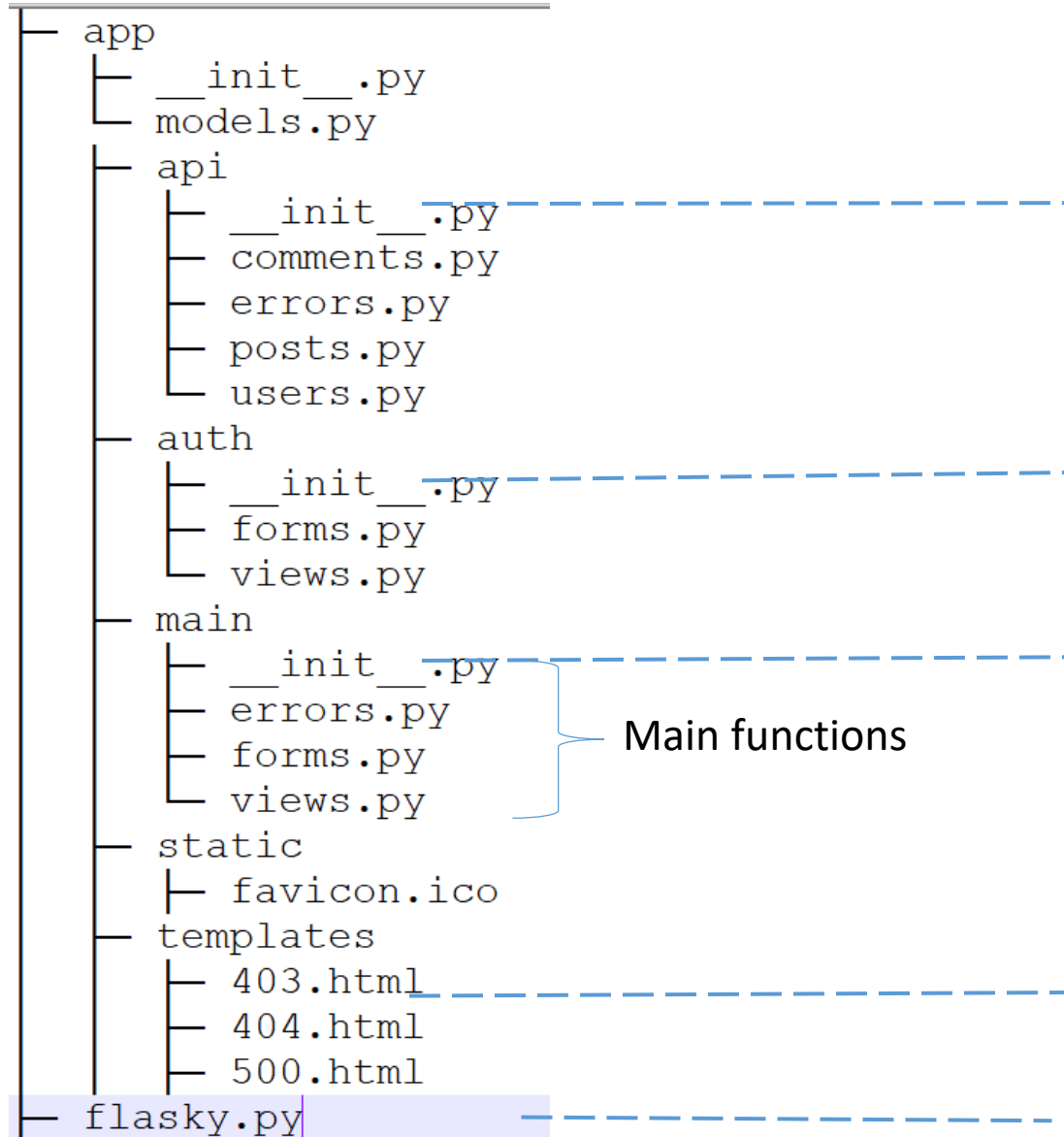
```
from flask import Flask, render_template

app = Flask(__name__)  # This is my flask app
app.debug=True

@app.route('/')
def index(): #view function
    return render_template('index.html')
```

- render_template is the function that is used to return html
- Flask looks for templates in the 'templates' directory

Organizing Flask Application



Flask application package

API package

Authentication package

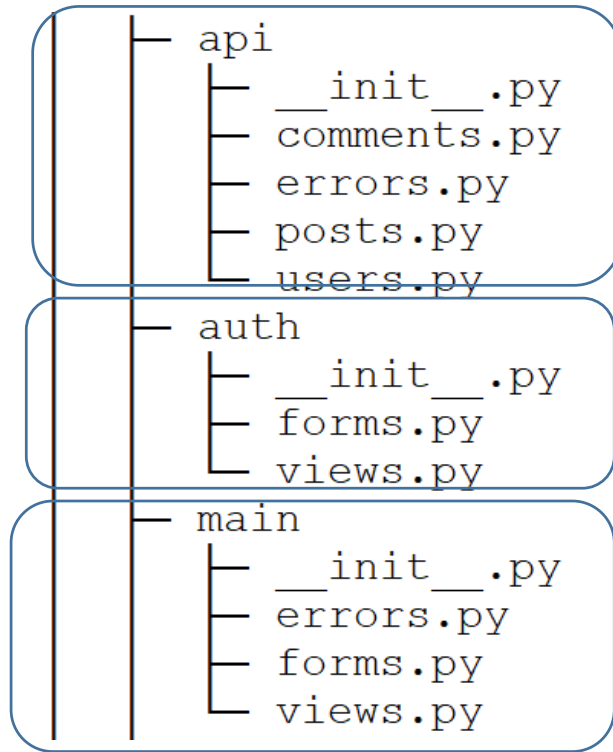
Main package

Static files

HTML files

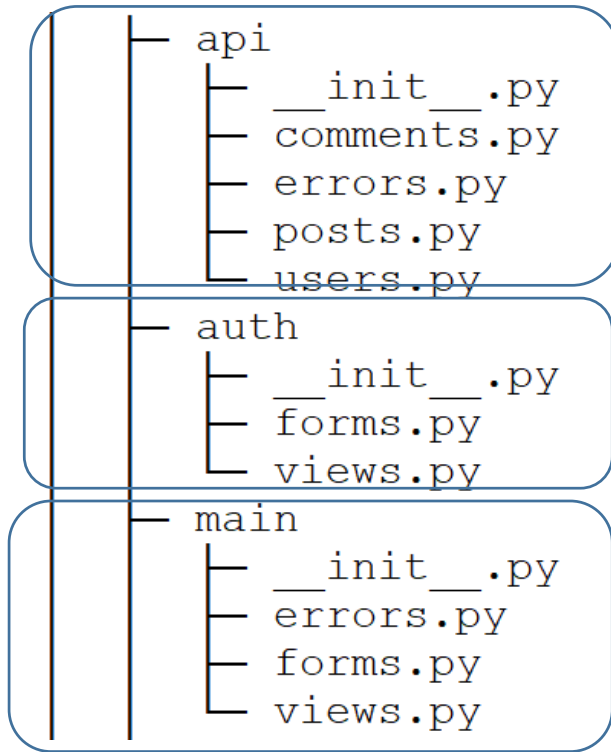
Python code that invokes Flask application

Blueprints



- Factor an application into a set of blueprints.
- Each Blueprint can have a collection of view functions, templates and static files
- Blueprint is similar to a Flask application object
- Blueprint is not a pluggable application
 - It cannot run independently

Blueprints



- Factor an application into a set of blueprints.
- Each Blueprint can have a collection of view functions, templates and static files
- Blueprint is similar to a Flask application object
- Blueprint is not a pluggable application
 - It cannot run independently

Blueprints

- Create a Blueprint

- Blueprint name
- Import name internally used to locate static and HTML files
- url_prefix – prefix to all routes of this blueprint (default is empty string)

```
#create a blueprint  
mybp = Blueprint('destination', __name__,  
url_prefix='/destinations')
```

```
#register routes with the blueprint
```

```
@mybp.route('/')  
def index():
```

- Register the Blueprint

```
app=Flask(__name__)  
app.register_blueprint(mybp)
```

Resources

- The mega flask tutorial (Miguel Grinberg)
 - First four chapters (<https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world>)
- Flask Web Development – Book by Miguel Grinberg

Summary

- Web application protocol
 - HTTP
 - Request types
- Frameworks and their functions
 - Inversion of control
- The **Flask** framework
 - Routes
 - Session
 - Rendering template
 - Blueprints

Reminder on Pulse Survey



**PULSE
SURVEY**
Your Feedback Matters

Evaluate and improve your unit with the Pulse survey

Feedback on your unit will enable you to review the design of curricula, learning activities and assessment, and reflect on the impact on students' learning. The survey closes on 26 August.

1. Feedback on the unit
2. Closes on August 26th

Thank you!