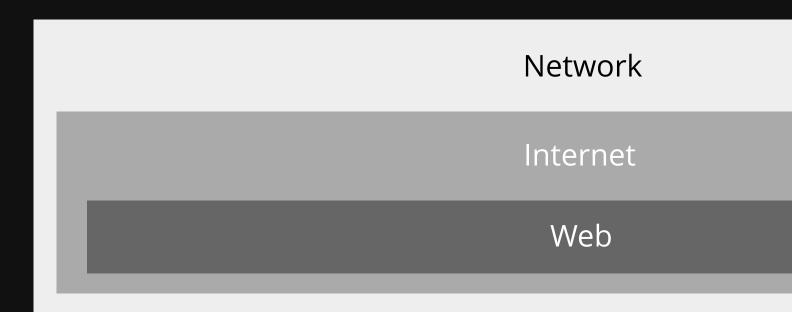
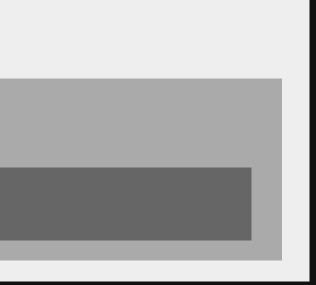
## COMP1531

#### 4.2 - Web - HTTP & Flask

### Computer Networks





### The network

This is not a networking course:

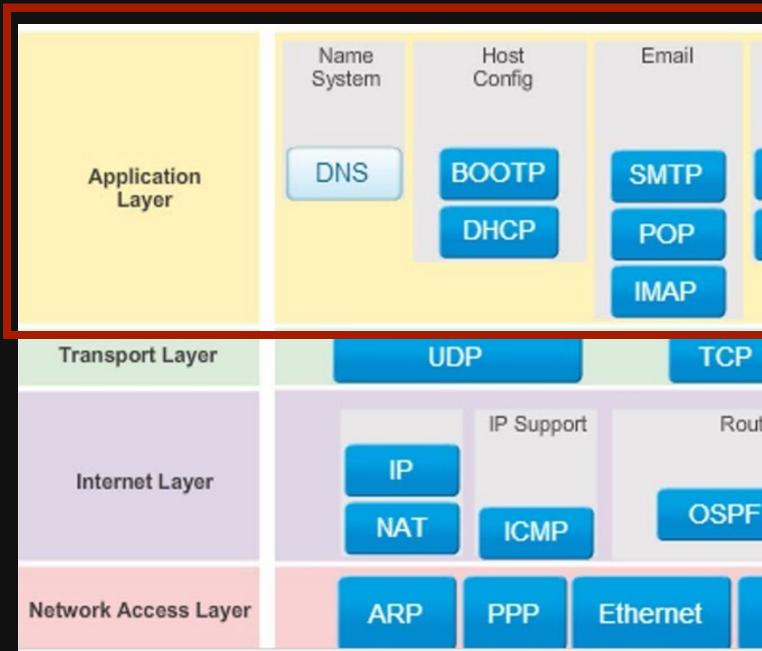
- Network: A group of interconnected computers that can communicate
- **Internet**: A global infrastructure for networking computers around the entire world together
- World Wide <u>Web</u>: A system of documents and resources linked together, accessible via URLs



### **Network Protocols**

- Communication over networks must have a certain "structure" so everyone can understand
- Different "structures" (protocols) are used for different types of communication

### Network Protocols



Source

File Transfer	Web			
FTP	нттр			
TFTP				
ting Protocols	3			
EK	GRP			
Interface Drivers				

## Examples?

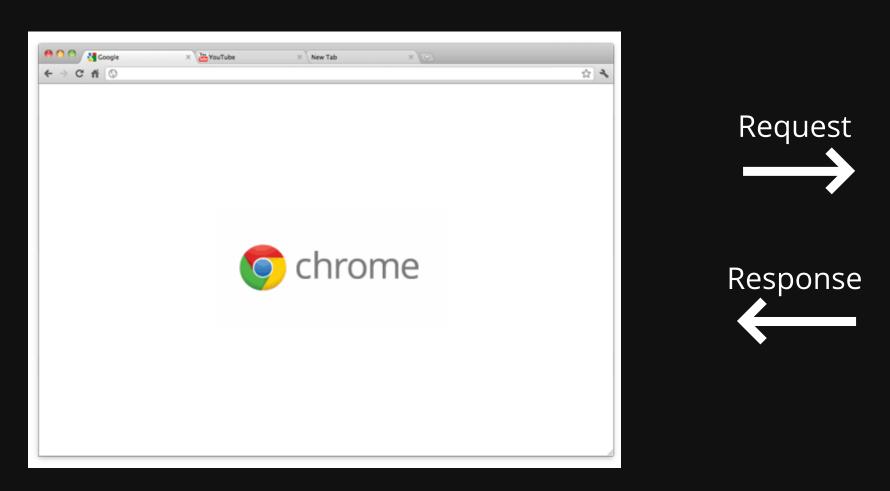
🔁 FileZilla			- 🗆 X			
File Edit View Transfer Server Bookmarks Help 	~ <b>A</b>					
Image: Second	ord: Port:	Quickconnect 💌		-		
Local site: C:\Users\smith\OneDrive\Documents\Adobe\	 ✓ Remote sit	ρ	·			
Filename     Filesize     Filesize         Filesize     Filesize	Mrite: (no subject					
File folder         29/10/2018 3:1           Adobe Medi         File folder         2/11/2018 3:57           dynamiclink         File folder         2/11/2018 3:57           Premiere Pro         File folder         19/09/2018 3:4	File Edit View	sert Format O <u>p</u> tions <u>I</u> pelling V 🎴 Security	ools <u>H</u> elp	and the second		
	F <u>r</u> om:	Hayden Smith <z3418003@< td=""><td>@ad.unsw.edu.au&gt;</td><td>Reputity Configuration</td><td></td><td>×</td></z3418003@<>	@ad.unsw.edu.au>	Reputity Configuration		×
	✓ To: 			Category: Session Logging Terminal Keyboard Bell	Options controlling session log Session logging: None Printable All session output SSH pace SSH packets and raw data	output
3 directories				iFeatures ⊡Window	Log file name:	
Server/Local file Direc Remote file	Paragraph V	ariable Width	▲ ▲ ▲ ▲	Appearance Behaviour Translation	putty.log (Log file name can contain &Y, &M, &D fo time, and &H for host name)	Browse r date, &T for
Queued files Failed transfers Successful transfers	 Hayden Smith ( <u>I</u>	<u>_inkedIn</u> )		Selection Colours Data Proxy Telnet	What to do if the log file already exists: Always overwrite it Always append to the end of it Ask the user every time Flush log file frequently	
	Lecturer-in-char School of Comp UNSW Sydney	ge COMP1531 uter Science and Eng	gineering	About	Options specific to SSH packet logging Omit known password fields Omit session data Open	Cancel



### HTTP

#### HTTP: Hypertext Transfer Protocol

I.E. Protocol for sending and receiving HTML documents (nowadays much more)



Web browsers are applications to request and receive HTTP





### **HTTP Request & Response**

#### HTTP Request

- 1 GET /hello HTTP/1.1
- 2 Host: 127.0.0.1:5000
- 3 Connection: keep-alive
- 4 Upgrade-Insecure-Requests: 1
- 5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90 Safari/537.36
- 6 Sec-Fetch-Mode: navigate
- 7 Sec-Fetch-User: ?1
- 8 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3
- 9 Sec-Fetch-Site: none
- 10 Accept-Encoding: gzip, deflate, br
- 11 Accept-Language: en-GB, en-US; q=0.9, en; q=0.8

#### HTTP Response

```
1 HTTP/1.0 200 OK
2 Content-Type: text/html; charset=utf-8
3 Content-Length: 12
4 Server: Werkzeug/0.16.0 Python/3.5.3
5 Date: Wed, 09 Oct 2019 13:21:51 GMT
6
7 Hello world!
```



#### Flask

#### Lightweight HTTP web server built in python

flask1.py

```
1 from flask import Flask
2 APP = Flask(___name___)
3
  @APP.route("/")
4
5 def hello():
      return "Hello World!"
6
7
8 if _____ == "____main___":
      APP.run()
9
```

1 \$ python3 flask1.py

## Server an image

Time to serve an image via a flask server...

#### flask2.py

```
1 from flask import Flask, send_file
2 APP = Flask(___name___)
3
4 @APP.route("/img")
5 def img():
      return send_file('./cat.jpg', mimetype='image/jpg')
6
7
8 if name == " main ":
      APP.run()
9
```

1 \$ python3 flask2.py

### Flask Reloading

Lightweight HTTP web server built in python

flask1.py



```
1 $ export FLASK_APP=flask1.py
2 $ export FLASK_DEBUG=1
3 $ export FLASK_RUN_PORT=0
4 $ python3 -m flask run
```

### Learn More

Some tutorials include:

1. https://pythonspot.com/flask-web-app-with-python/ 2. https://blog.miguelgrinberg.com/post/designing-a-restful-api-with-

python-and-flask

When it comes to online tutorials, note that:

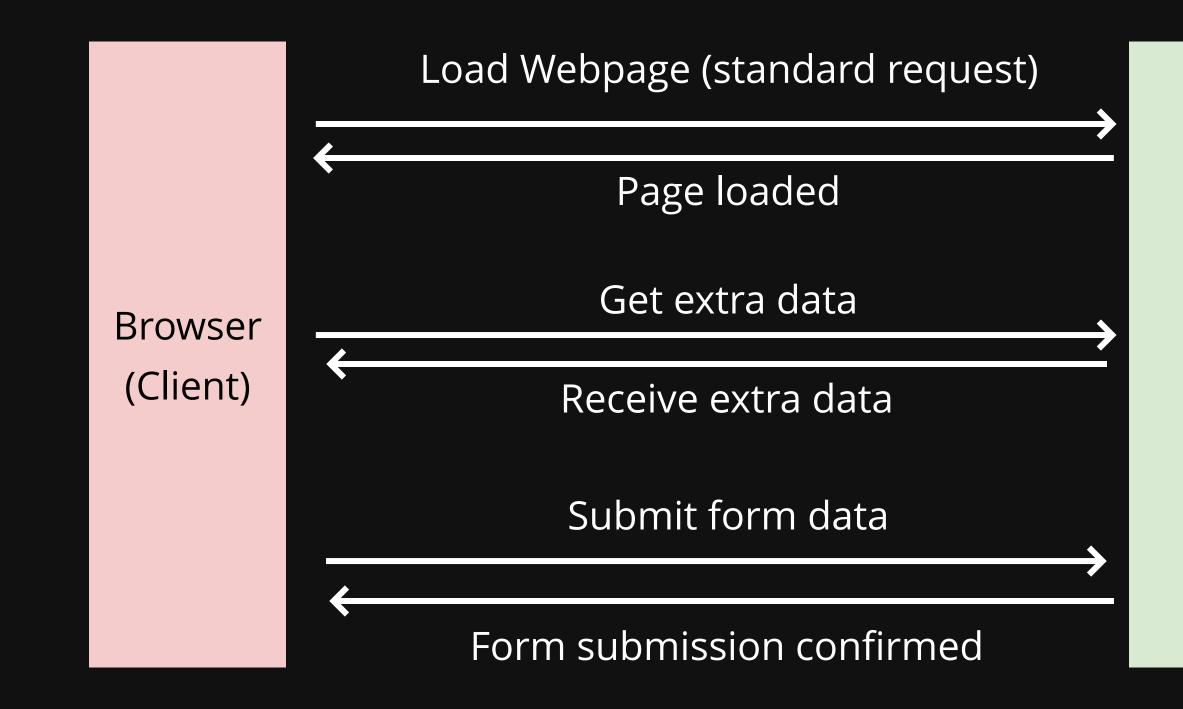
- Each "tutorial" may be using different python versions
- Each "tutorial" may have different aims in mind

#### API

An API (Application Programming Interface) refers to an interface exposed by a particular piece of software.

The most common usage of "API" is for Web APIs, which refer to a "contract" that a particular service provides. The interface of the service acts as a black box and indicates that for particular endpoints, and given particular input, the client can expect to receive particular output.

### Web API



#### Server

## Restful API & "CRUD"

A *RESTful API* is an application program interface (*API*) that uses HTTP requests to GET, PUT, POST and DELETE data. These 4 methods describe the "nature" of different API requests.

GET, PUT, POST, DELETE are HTTP Methods

Method	Operation
POST	Create
GET	Read
PUT	<b>U</b> pdate
DELETE	Delete



## Input & Output

Different CRUD properties require different approaches for input. All output are the same.

	1	from
	2	from
Innute are aither	3	
Inputs are either:	4	APP =
	5	0
<ul> <li>GET: via URL and "request.args"</li> </ul>	6	@APP.
	/	def o
<ul> <li>PUT   POST   DELETE: via post-data</li> </ul>	8 9	r
and via "request.get_json()"	9 10	
and via request.get_json()	11	٦
<ul> <li>All outputs should be packaged up as</li> </ul>	12	ſ
	13	@APP.
ISON	$14^{-1}$	def t
	15	d
<ul> <li>(JSON discussed later)</li> </ul>	16	r
	17	
	18	
	19	}
	20	
	21	if
	22	Δ

#### crud.py

```
flask import Flask, request
 json import dumps
 Flask(___name___)
 route("/one", methods=['GET'])
 ne():
 eturn dumps({
    '1': request.args.get('data1'),
    '2': request.args.get('data1'),
 route("/two", methods=['POST'])
 WO():
 ata = request.get_json()
 eturn dumps({
   '1': data['data1'],
    '2': data['data2'],
 name == ' main ':
APP.run()
```

## Using CRUD and state

#### Task:

Create a web server that uses CRUD to allow you to create, update, read, and delete a point via HTTP requests

Use a global variable to manage the state.

#### point.py

## Talking to Flask

How can we talk to flask? 1. API client 2. Web Browser 3. URLLib via python

## API Client (ARC/Postman/Insomnia)

How to download/install postman:

- Open google chrome
- Google "ARC client"
- Install the addon and open it
- Follow the demo in the lectures

You may need to use a tool like this in the final exam.

it ures

## API Client (A R C)

$\equiv$ ARC	Request	(i)	:
HTTP request	Method GET • Request URL •	SEND	•
Socket	An URL is required. Parameters	_	
History ^	Headers Variables		
Send a request and recall it from here	Toggle source mode + Insert headers set		
Once you made a request it will appear in this place.	Header name Header value     ADD HEADER     Apple Headers are valid	Headers size: by	<b>X</b> ytes
Saved ^			
Save a request and recall it from here			
place.			

Install new ARC with new features!

- (j

### Web Browser

S 127.0.0.1:5000/hello	× +						
← → C ③ 127.0.0.1	:5000/hello						
Hello World!		Elements Co	nsole	Source	s Netwo	rk Perfor	mance
		🌄 Q   🗆 Pr	reserv	ve log 🔲 🛙	Disable cache	e Online	•
	Filter			Hide data	URLs All	XHR JS CS	S Img I
	10 ms	20 m s	30 ms	40 ms	50 ms	60 ms	70
	Name		×	Headers	Preview	Response	Timing
	📃 hello		<b>▼</b> G	eneral			
				Request UI	RL: http://	/127.0.0.1:	:5000/he
					ethod: GET		
					e: 🔵 200 0	к .0.0.1:5000	
						ferrer-whe	
			▼ R	esponse He	eaders	view source	
				Content-Le	ength: 12		
				Content-Ty	/ <b>pe:</b> text/h	tml; chars	et=utf-
				Date: Wed,	09 Oct 20	019 13:26:0	05 GMT
				Server: We	rkzeug/0.1	.6.0 Python	/3.5.3
			▼ R	equest Hea	i <b>ders</b> vi	ew source	
				Accept: te	xt/html,ap	oplication/	/xhtml+x
					-	,applicati	_
				-		p, deflate	
	1 requests	166 B transferre	-	-	<b>trol:</b> max-a	-GB,en-US;o ge=0	4=0.9,en
					and the second second	0	

							—				×	
						☆	9	•		H	:	
М	1em	ory	A	Appl	licatio	on	»			:	×	
<u></u>	4	Ł									ф	;
Med	dia	Font	D	oc	WS	Ma	anifes	st	Oth	er		
'0 ms		80 n	ns		90	ms		10	0 ms		11	01
g												
grad												*
-xml,	, app	olicat	ti	on/	xml;	q=0	.9,i	ma	ge/1	webp	),i	

ned-exchange;v=b3

n;q=0.8

### requests - Python

**requests** is a python3 library that allows you to programmatically make HTTP requests to a web server.

You will use this extensively in iteration 2.

### requests - Python

#### echo.py

```
1 from flask import Flask, request
 2 from json import dumps
 4 APP = Flask(_______)
 5
 6 @APP.route("/echo", methods=['GET'])
 7 def echo():
       return dumps({'data': request.args.get('data')})
 8
 9
10 if _____ == '___main___':
       APP.run()
11
```

#### echo\_main.py

```
1 import json
2 import requests
3
4 def get payload():
      response = requests.get('http://127.0.0.1:5000/echo', params={'data': 'hi'})
5
      payload = response.json()
6
      print(payload)
8
9 if name == ' main ':
      get payload()
10
```

We expect you to do your own research for POST

### Web server as a wrapper

Because you've written so many **integration** tests for iteration 1, it makes sense to: 1. Implement all of the functions from iteration one 2. Then wrap them in a flask server

### Web server as a wrapper

#### iter2example/search.py

1	def	<pre>search(token, query_str):</pre>
2		return {
3		'messages' : [
4		'Hello ' + token + ' ' + query
5		<pre># Not the right structure</pre>
6		]
7		}

#### iter2example/server.py

```
1 from json import dumps
 2 from flask import Flask, request
   from search import search_fn
 4
 5
 6 APP = Flask( name )
 7
 8 @APP.route('/search', methods=['GET'])
 9 def search flask():
       return dumps(search(requests.args.get('token'), request.args.get('query str')))
10
11
12 if _____ == '___main__':
13
       APP.run()
```

y str,

# (Bonus) interesting question

How do companies track whether or not you've read an email they've sent you?