FALMOUTH UNIVERSITY

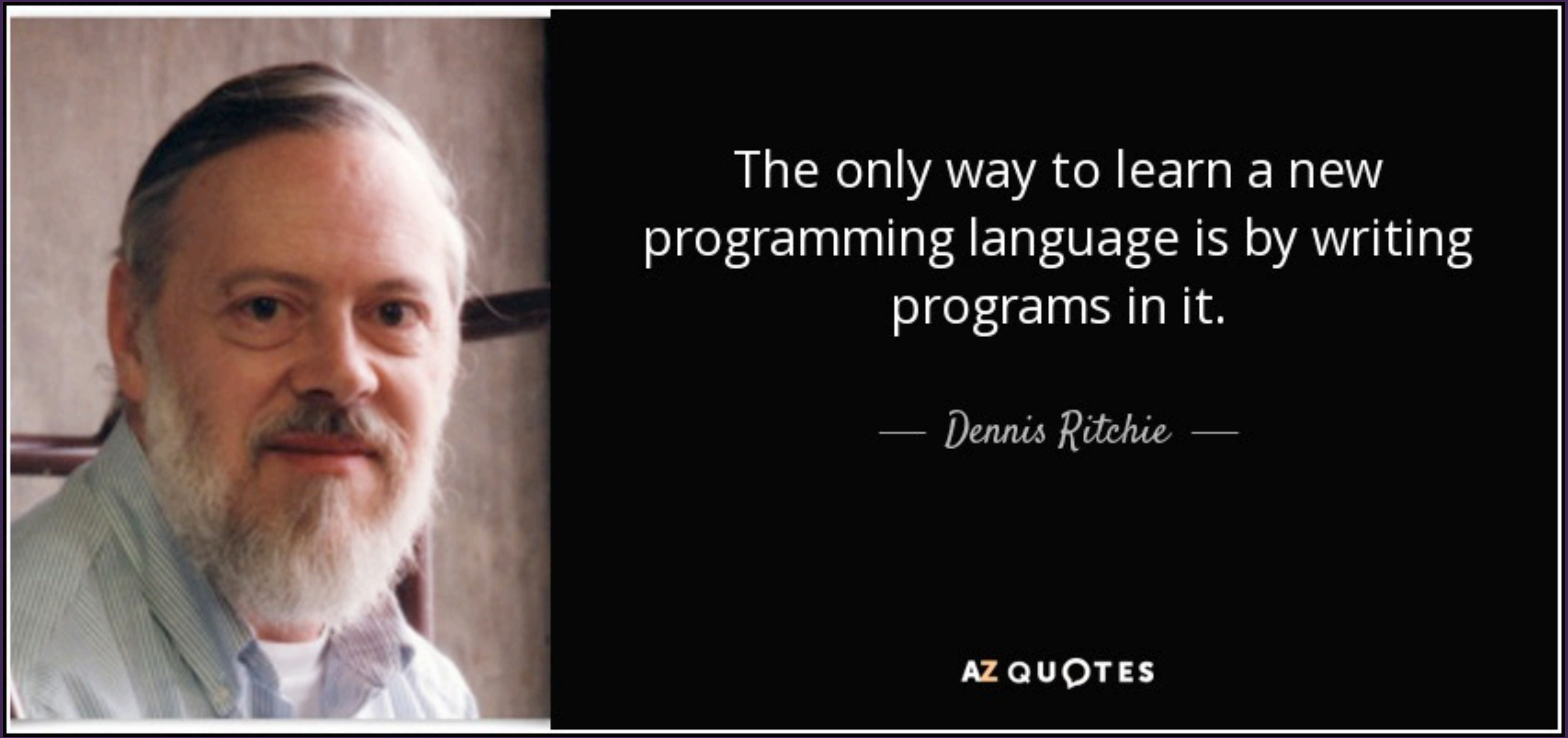
Lecture 4: Networking in Python

COMP280: Creative Computing
BSc(Hons) Computing for Games
BA(Hons) Game Development: Programming

F

- Learning outcomes
  - **Understand** how the http server framework is implemented in Python
  - **Develop** suitable JSON packets to transmit data between client & server and server & client application
  - **Create** simple relational databases using sqlite
  - **Program** queries in sql to CRUD (create, retrieve, update and delete) data in a sqlite database

- Worksheet and learning outcomes
  - https://learningspace.falmouth.ac.uk/course/view.php?id=4355
  - Use key full stack technologies to build applications
    - Python client
    - Unreal client
    - Python Server

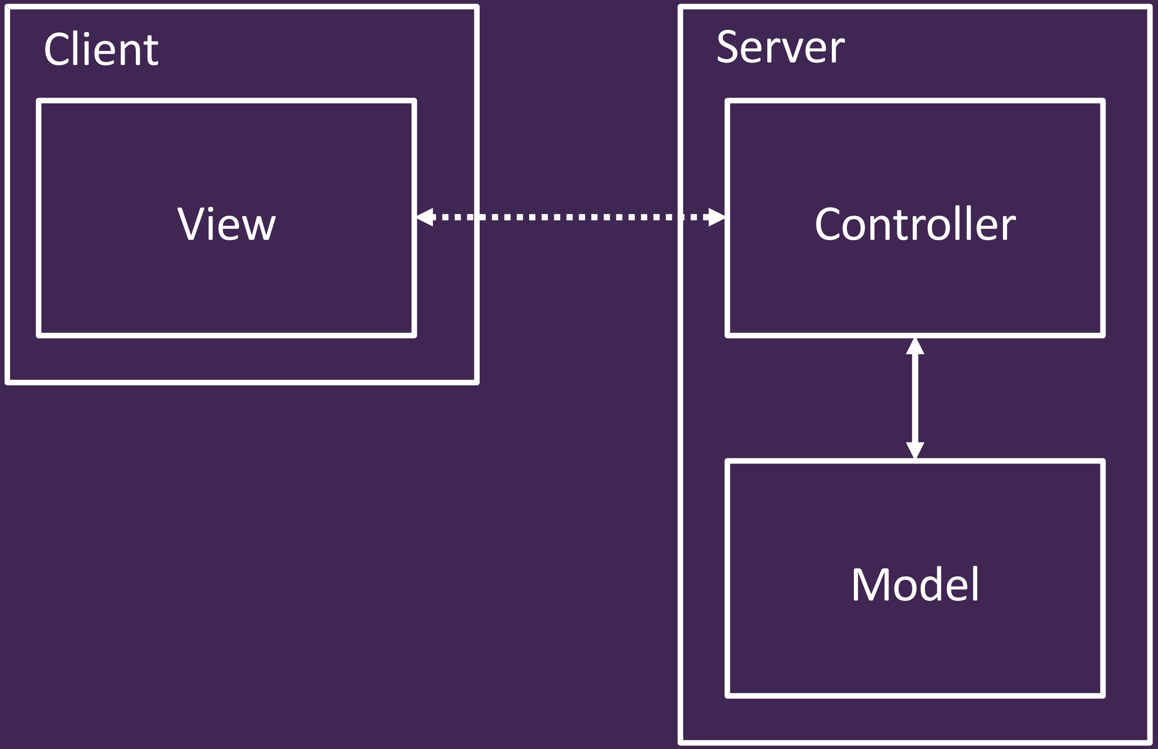


The only way to learn a new programming language is by writing programs in it.

— Dennis Ritchie —

AZ QUOTES

- Key Full stack Technologies
  - When we talk about full stack, we are talking about client-server architecture
    - And model, view, controller (MVC) architecture



- MVC
  - Client(s)
    - **View** allows user to receive data and enter input
  - Server
    - **Controller** takes user input and sanitises it before updating model
    - **Model** stores all data

- Key Full stack Technologies
  - Server programming language & development frameworks (Controller)
  - Model language and tools
  - Data communications protocols between client and server
  - Client programming language & development frameworks (View)

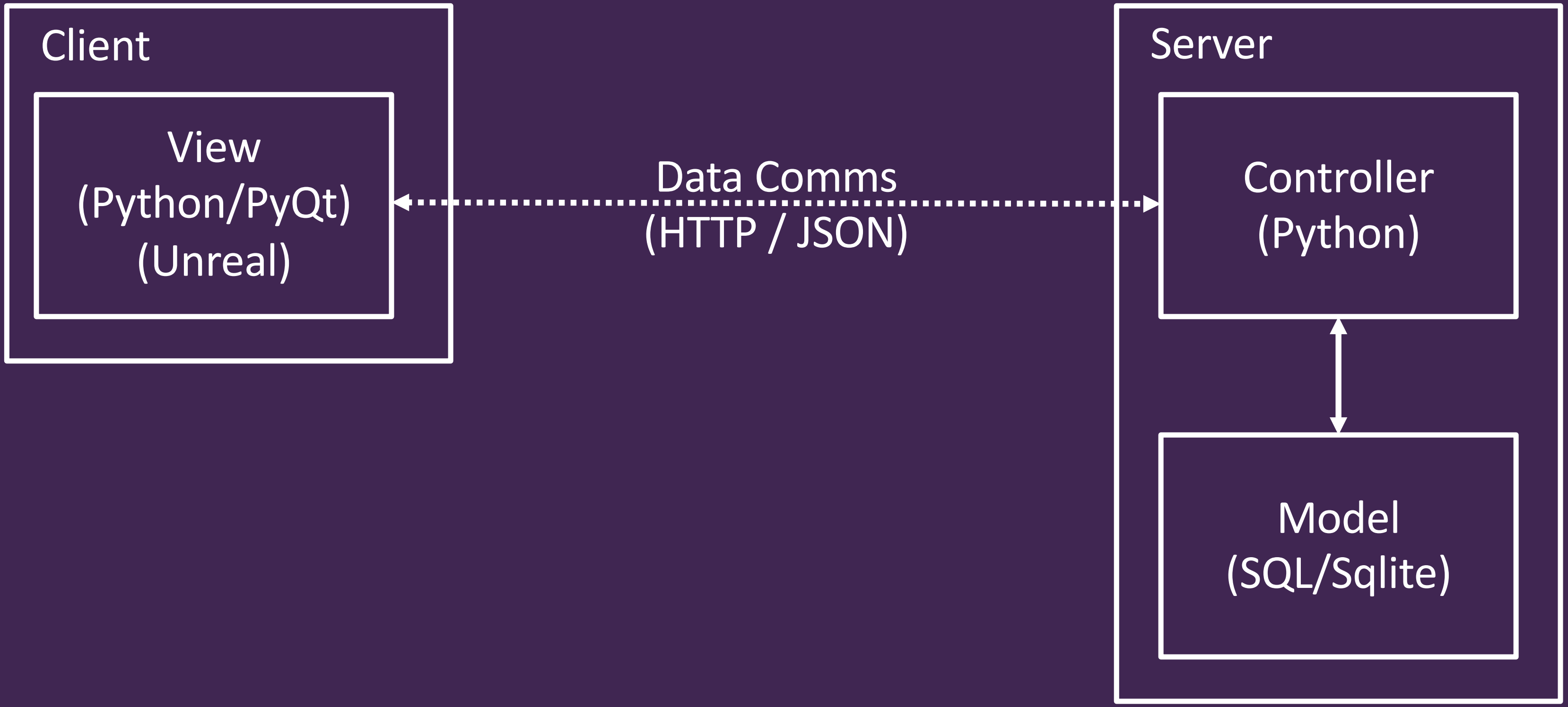
- Key Full stack Technologies
  - Server programming language & development frameworks
    - Python
      - Cross-platform language with lots of technical support
      - Wide range of tools and support libraries
      - Geared around rapid development
      - Strong support for different environments and locales
      - Simple threading architecture
  - Model language and tools
  - Data communications protocols between client and server
  - Client programming language & development frameworks

- Key Full stack Technologies
  - Server programming language & development frameworks
  - Model language and tools
    - SQL
      - Industry standard database description and programming language
    - Sqlite
      - Cross-platform implementation of SQL reference
  - Data communications protocols between client and server
  - Client programming language & development frameworks

- Key Full stack Technologies
  - Server programming language & development frameworks
  - Model language and tools
  - Data communications protocols between client and server
    - HTTP (Hypertext Transmission Protocol)
      - Industry-standard and mature communications protocol
      - Yes, it’s how all the browsers work
    - JSON (JavaScript Object Notation)
      - Industry-standard data transformation library
      - Well-suited to different platforms and languages
  - Client programming language & development frameworks

- Key Full stack Technologies
  - Server programming language & development frameworks
  - Model language and tools
  - Data communications protocols between client and server
  - Client programming language & development frameworks
    - Python
      - See above
    - PyQt
      - UX/UI framework for Python to create attractive application UI
    - Unreal
      - Industry-leading game engine

- Key Full stack Technologies



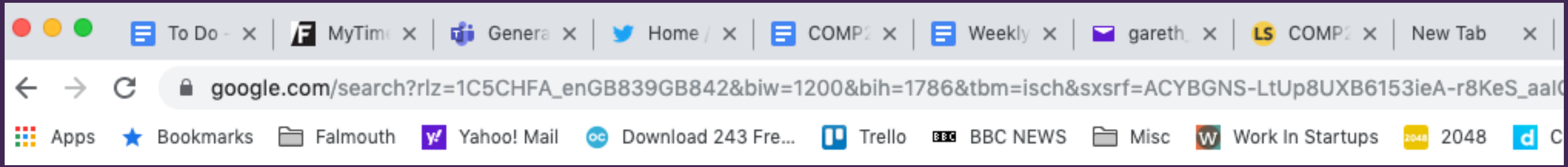
- HTTP Server
  - HTTP is part of the WWW protocols
    - Hypertext Transmission Protocol
      - As a TP it defines how data is transferred between nodes on the internet
        - » Internet –> inter-network, a network of nodes
        - » Sits on TCP/IP
          - IP is the Internet Protocol (how nodes are named (IP address) and connected
          - TCP is the Transmission Control Protocol
            - Make sure packets of data turn up
            - & in the correct order
            - (not like datagams)

- HTTP Server
  - HTTP is part of the WWW protocols
    - Defines client and server roles
      - Client will send messages (requests) to a server
        - » OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT & PATCH
        - » Generally we will use POST & GET
          - POST – send something to the server
          - GET – ask the server for some data
      - Server will respond to them

- HTTP Server
  - HTTP is part of the WWW protocols
    - Like using a browser, this is client driven
      - Client sends a request, server responds
      - Client deals with response
      - Server can't originate communications
    - Client-centric nature makes it useful for:
      - Requesting data from a server (persistent data, balance data etc)
      - Sending data to a server (high scores, analytics, persistent data)
      - Need to consider sockets for 'proper' network gaming (multiplayer FPS etc)

FALMOUTH UNIVERSITY

- HTTP Server
  - HTTP is part of the WWW protocols
    - Generally a text (hypertext) protocol
    - You will often see HTTP command & data in web addresses


To Do · MyTim · Genera · Home / · COMP · Weekly · gareth · COMP · New Tab

google.com/search?rlz=1C5CHFA_enGB839GB842&biw=1200&bih=1786&tbm=isch&sxsrf=ACYBGNS-LtUp8UXB6153ieA-r8KeS_aal

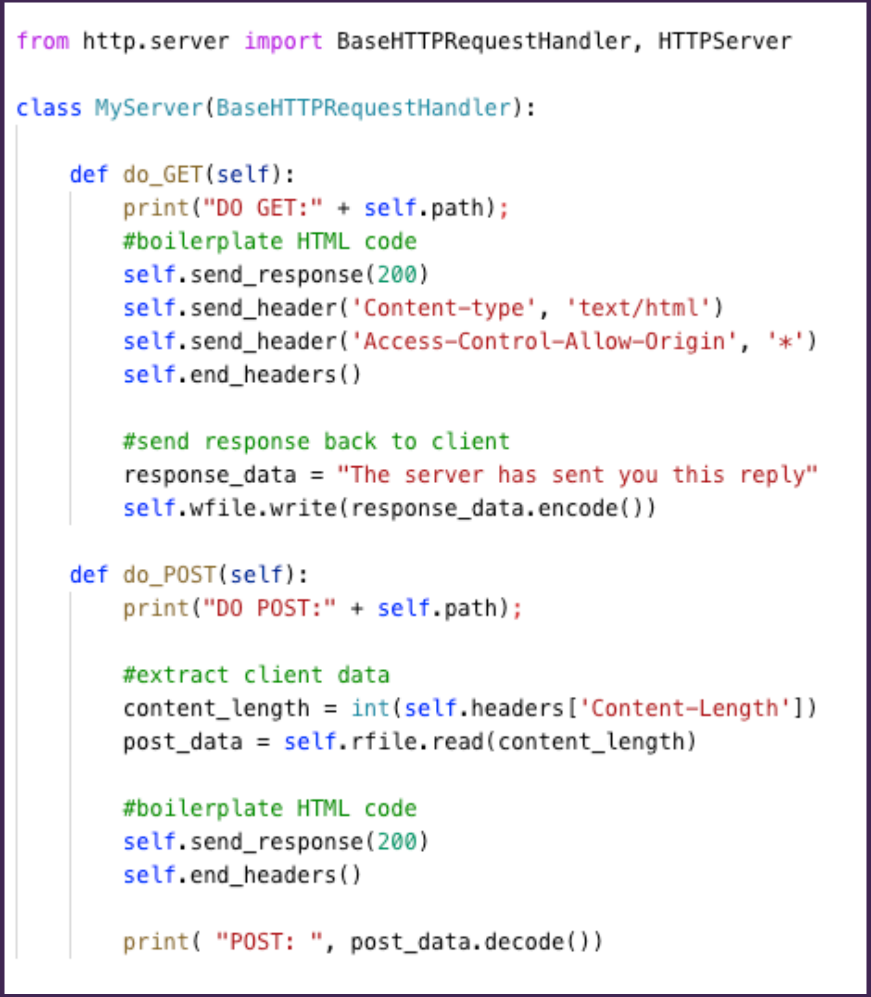
Apps · Bookmarks · Falmouth · Yahoo! Mail · Download 243 Fre... · Trello · BBC NEWS · Misc · Work In Startups · 2048

- As we saw from cybersecurity, this can be extremely hackable
  - SQL injections!
  - HTTPS (s- secure) should prevent that

- HTTP Server in Python
  - Python provides:
    - (Server-side) Abstract HTTPServer class
      - Make your own servers by overloading (re-writing) do_POST and do_GET methods
      - Create an instance of your server and serve_forever()
    - (Client-side) HTTPConnection class
      - Create an instance of an HTTPConnection()
      - Use
        - » request(<command>, <data>) to send a request to the server
        - » getresponse() to see if it worked
        - » response.read() to read the response data

FALMOUTH UNIVERSITY

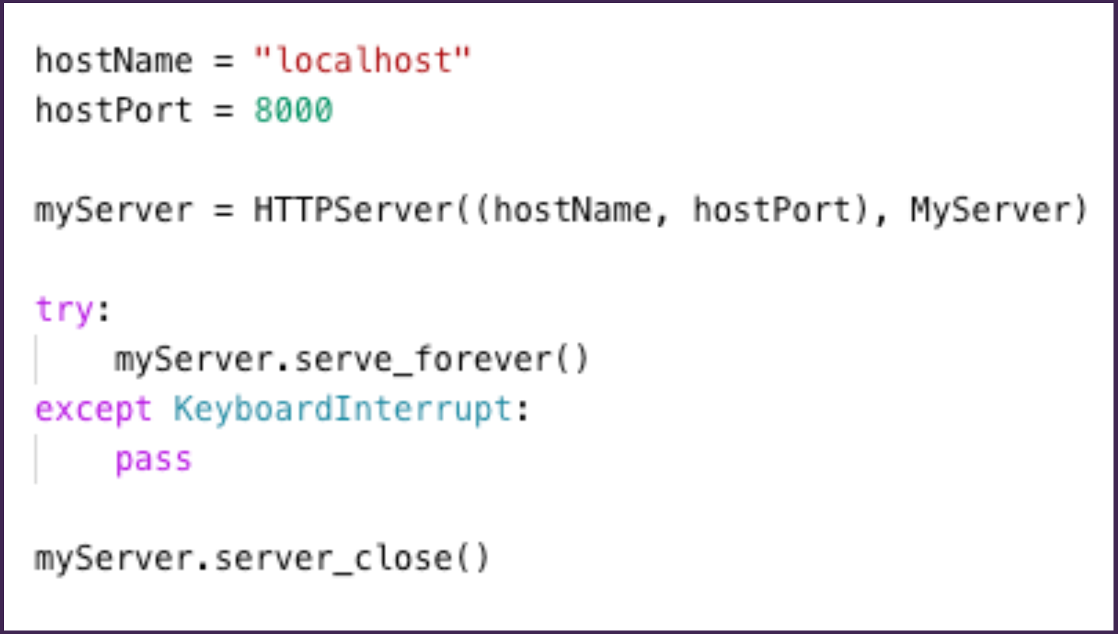
- HTTP Server in Python
  - The server app: Boilerplater HTMPL Server class



```python
from http.server import BaseHTTPRequestHandler, HTTPServer

class MyServer(BaseHTTPRequestHandler):

    def do_GET(self):
        print("DO GET:" + self.path);
        #boilerplate HTML code
        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.send_header('Access-Control-Allow-Origin', '*')
        self.end_headers()

        #send response back to client
        response_data = "The server has sent you this reply"
        self.wfile.write(response_data.encode())

    def do_POST(self):
        print("DO POST:" + self.path);

        #extract client data
        content_length = int(self.headers['Content-Length'])
        post_data = self.rfile.read(content_length)

        #boilerplate HTML code
        self.send_response(200)
        self.end_headers()

        print( "POST: ", post_data.decode())
```

FALMOUTH UNIVERSITY

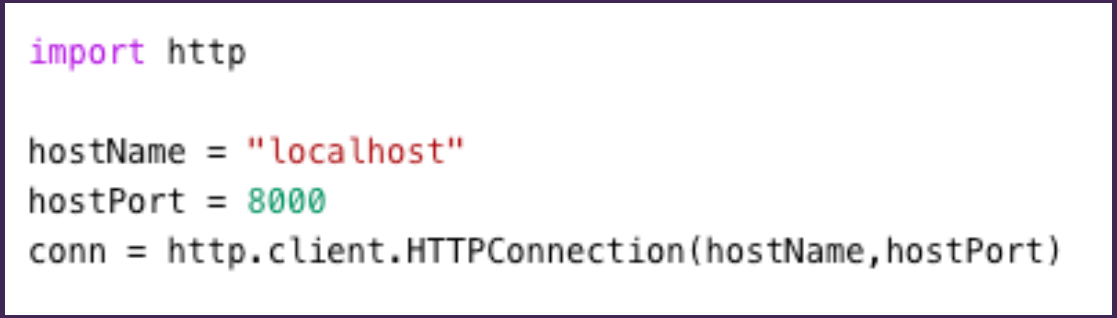
- HTTP Server in Python
  - The server app: Instantiate & Launch



```
hostName = "localhost"
hostPort = 8000

myServer = HTTPServer((hostName, hostPort), MyServer)

try:
    myServer.serve_forever()
except KeyboardInterrupt:
    pass

myServer.server_close()
```

- This will just sit waiting for clients to serve (as designed)

- HTTP Server in Python
  - The client app: Create an HTTPConnection

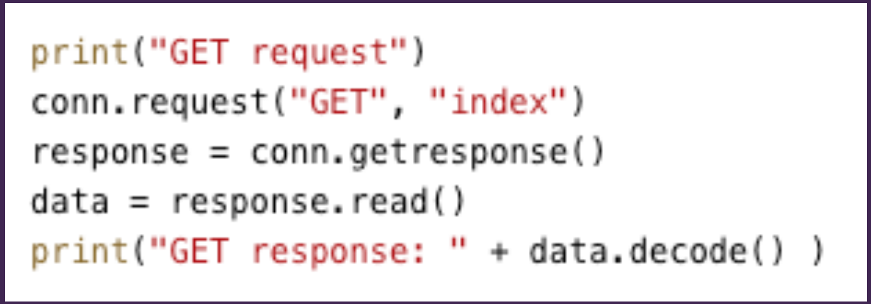


```
import http

hostName = "localhost"
hostPort = 8000
conn = http.client.HTTPConnection(hostName,hostPort)
```

- NB
  - Make sure the hostname and port address match between client and server
  - OS can be sniffy on port IDs, check stackoverflow if your apps fail on instantiation

FALMOUTH UNIVERSITY

- HTTP Server in Python
  - The client app : Boilerplater GET request

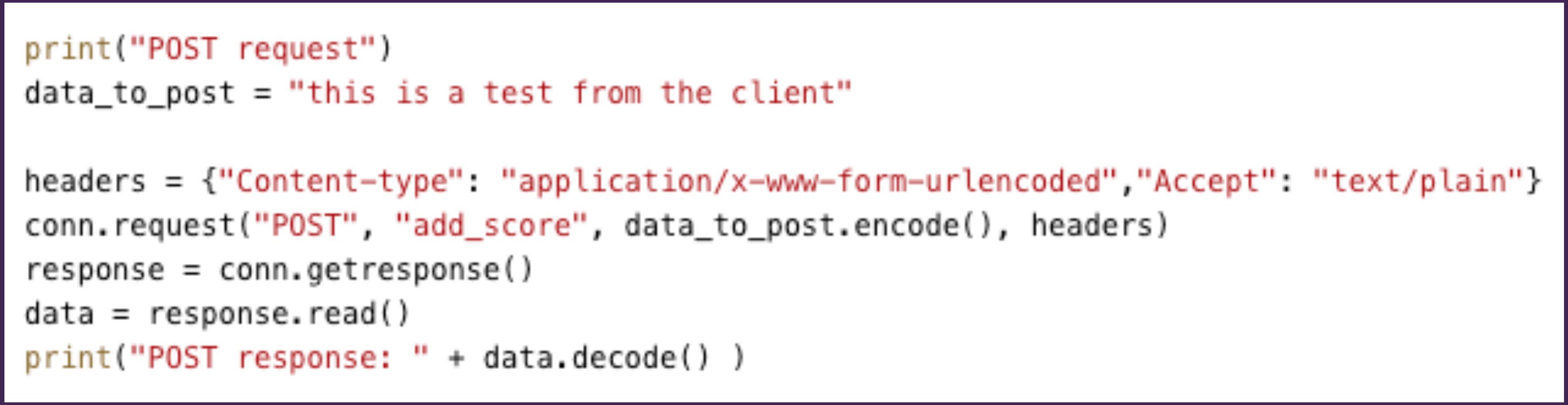


```
print("GET request")
conn.request("GET", "index")
response = conn.getresponse()
data = response.read()
print("GET response: " + data.decode() )
```

- NB
  - Arguments and commands can embedded into the ‘index’ field of the conn.request (ripe for injections)
    - » I would recommend JSON for this as it’s more flexible
  - Server will respond with bytes, so decode back to text

FALMOUTH UNIVERSITY

- HTTP Server in Python
  - The client app : Boilerplater POST request



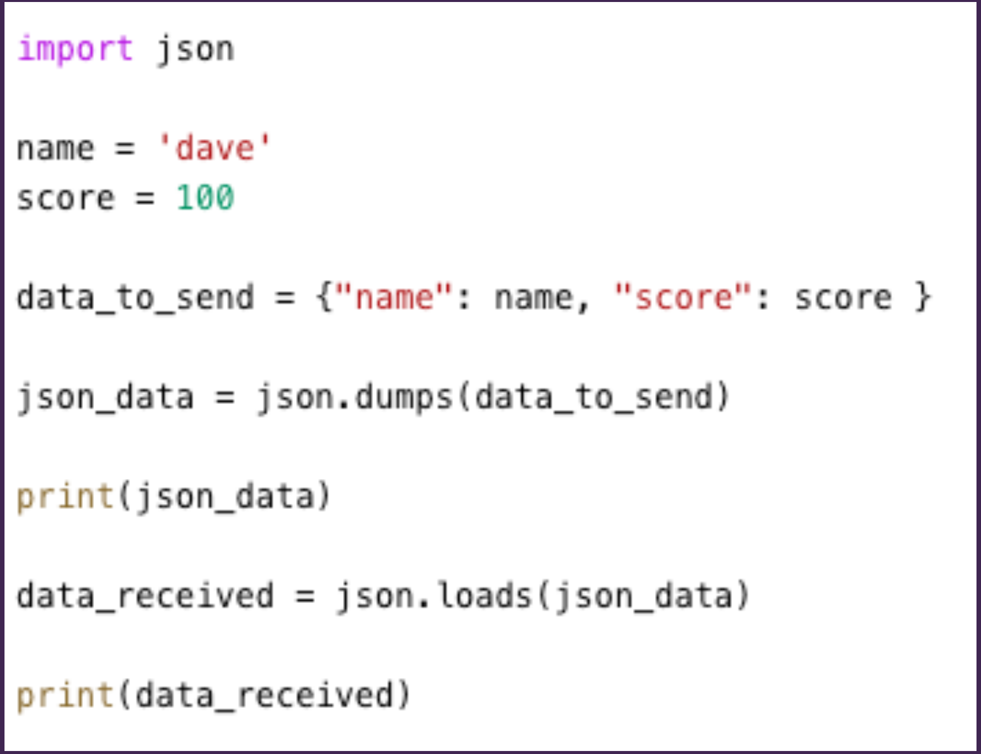
```
print("POST request")
data_to_post = "this is a test from the client"

headers = {"Content-type": "application/x-www-form-urlencoded","Accept": "text/plain"}
conn.request("POST", "add_score", data_to_post.encode(), headers)
response = conn.getresponse()
data = response.read()
print("POST response: " + data.decode() )
```

- POST will take arguments as part of command as data_to_post
  - » Ideal for JSON encoding
  - » Also support for data in the 'add_score' argument
    - See GET issues
- However, need to pass in headers to define content type (generally just boilerplate)

- JSON as a data protocol
  - JSON (JavaScript Object Notation)
    - Open-standard format for serialising and deserialising JavaScript object
    - Has become adopted as an open-format on the internet
    - Because it works with *objects* can serialise sequences and hierarchies of dissimilar objects
    - Works really well, until it doesn’t
      - Interesting edge cases to consider, especially between different languages (JS<->Python <->C# <->C++)

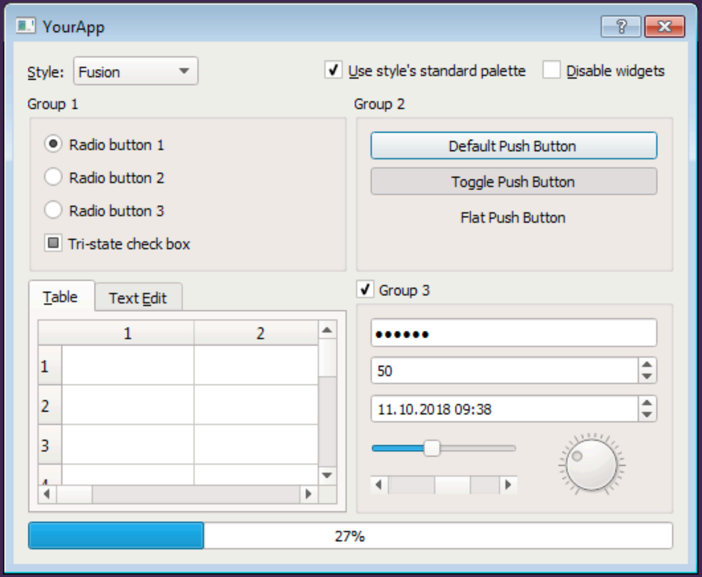
- JSON as a data protocol
  - JSON (JavaScript Object Notation)
    - Python supports JSON as a core language feature
      - `json class (import json)`
      - `json.dumps(object)` -> convert object to json string
      - `json.loads(string)` -> convert string to object
      - https://www.w3schools.com/python/python_json.asp
    - Process
      - Object->json->string->bytes -> HTTP
      - HTTP->bytes->string->json->Object
  - https://www.w3schools.com/python/ref_string_encode.asp

- JSON as a data protocol
  - JSON (JavaScript Object Notation)



```python
import json

name = 'dave'
score = 100

data_to_send = {"name": name, "score": score }

json_data = json.dumps(data_to_send)

print(json_data)

data_received = json.loads(json_data)

print(data_received)
```

- PyQt as a view framework
  - Qt is an industry standard multi-platform UI framework
    - PyQt is the Python wrapper for it
    - Has a UI design tool (QtDesigner)
      - Projects can be loaded into a python app and Python’s reflection will sort out all the names
    - Import into your PyCharm projects through Package Manager



- SQL / sqlite
  - Python supports sqlite as a core part of the language framework
    - `import sqlite`
  - Ideal for managing persistent data on server (or client)
  - SQL queries allow you to perform complex queries on your data
    - Also ideal for high score tables, simple analytics, game balance and so on.

- SQL
  - We can think of a database as collection of data that is stored in tables
    - Ideally, each entry will have a unique identifier to make possible to identify that record and no other

CustomerID	Name	Age	Email
0	Dave	32	dave@dave.com
1	Barry	94	barry@barrysworld.org
2	Joe	22	john@facebook.co.uk

- SQL
  - We can think of a database as collection of data that is stored in tables
    - We can use unique references to link data between tables
      - Purchase records link to customers (last slide) and items (some other table)

Purchase ID	Customer ID	Delivered	Items
76	0	Y	[754,281,998]
77	0	N	[4]
78	1	Y	[123,556,4]

- SQL

- Let’s make a kick-ass telephone book as a client-server application
  - Make a client-server application that will let you CRUD telephone numbers of your friends (whilst making sure you follow GDPR compliance)
    - Store the data in an SQL database on a server
    - Use a PyQt python client to access data
    - Use HTTP requests and JSON to manage data transmissions
  - It’s really small, but it will test all the aspects of full stack development
    - This will cover the Python & SQL criteria of the worksheet
    - Feel free to add more complexity if you fancy

- Questions