Edinburgh Napier
UNIVERSITY

ADVANCED WEB TECHNOLOGIES
SET09103
TOPIC #01

HELLO WORLD (WIDE WEB)

Dr Simon Wells
s.wells@napier.ac.uk
http://www.simonwells.org

Edinburgh Napier UNIVERSITY

OVERVIEW

- Part# 1: The architecture of the Web
- Part#2: Using Python & Flask to say "Hello World"

Edinburgh Napier
UNIVERSITY

PART #1: BACKGROUND

Edinburgh Napier
UNIVERSITY

TL/DR

The World Wide Web is predicated upon an architecture (clients & servers) and core protocols (URLs, HTTP, HTML)

Edinburgh Napier
UNIVERSITY

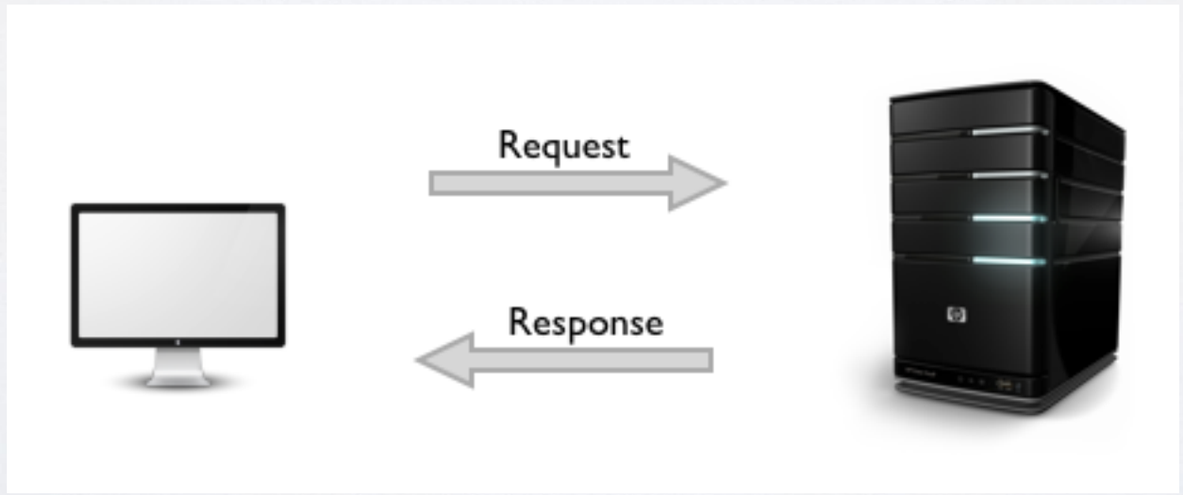
DEVELOPMENT OF THE MODERN WEB

- 1980 —1991 (Web1.0) - Birth of the Web (invention of HTTP, HTML, URL, Browser)
- 1992 —1999 (Web1.0) - Static HTML websites (text, links, inline styles)
- 2000 —2006 (Web2.0) - Dynamic Websites (blogs, forums, wikis)
- 2007 — 2011 (Web2.0) - Interactive Websites (social networks, AJAX, CRUD, Multi language support)
- 2012 — today (Web2+/rich Web) - Rich Applications (HTML5, CSS3, Mobile/ SmartPhone)
- today — Future (Web3.0) - Decentralised? Intelligent? Permanent? Realtime? Dark?

Edinburgh Napier UNIVERSITY

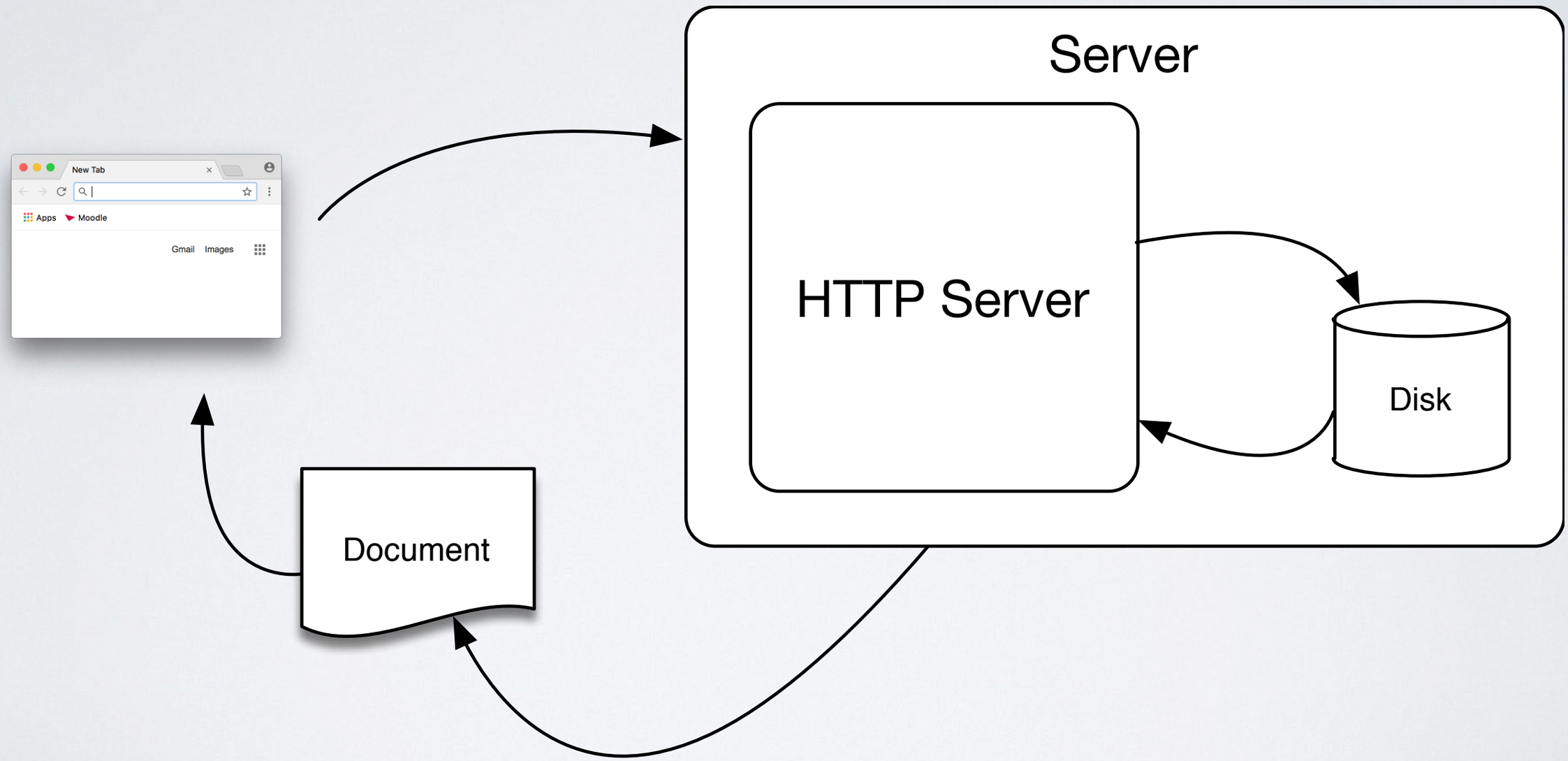
CLIENTS & SERVERS

- A server waits for a client to communicate with it.
- The client initiates communication with the server.
- The server interprets the clients **request** and sends an appropriate **response**.
- This is the **request/response** pair
- The client initiates an HTTP **request** message and the host *services* the request and returns a **response** message



CLIENT SERVER ARCHITECTURES

Edinburgh Napier UNIVERSITY



Edinburgh Napier UNIVERSITY

HTTP

- Clients, servers, requests and responses are all part of the HTTP protocol
- HTTP is an agreement for text-based communication between clients & servers for retrieving, altering, & deleting resources (amongst other things)
- HTTP is *the* basic core protocol that underpins the Web
  - Defines:
    - how messages (primarily text) are formatted and transmitted
    - what actions web-servers & browsers should perform in response to a given message
- Stateless, application layer protocol for communicating between distributed systems

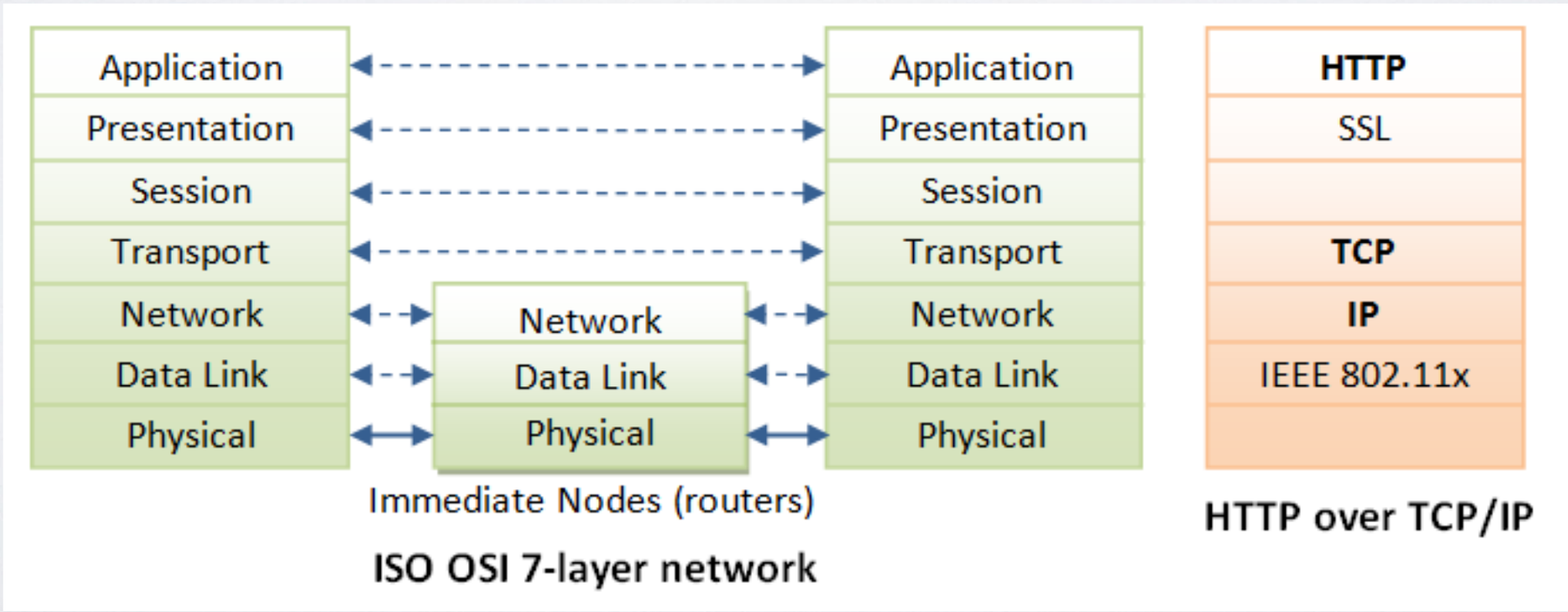
Edinburgh Napier UNIVERSITY

HTTP BASICS

- Allows for communication between hosts and clients, supports many network configurations, and is robust.
  - How? By not assuming very much about any particular underlying system & not keeping state (stateless)
- Communication uses TCP/IP (but can work over any reliable transport) and runs, by default, over port **80**
  - *NB. Other ports can be used, e.g. we are using 5000 for our web-apps during development*

RELATIONSHIP TO WIDER NETWORK STACK

- HTTP operates at the Application layer (the top layer)
- Doesn't care too much about what is below so long as it can transmit data and is reliable



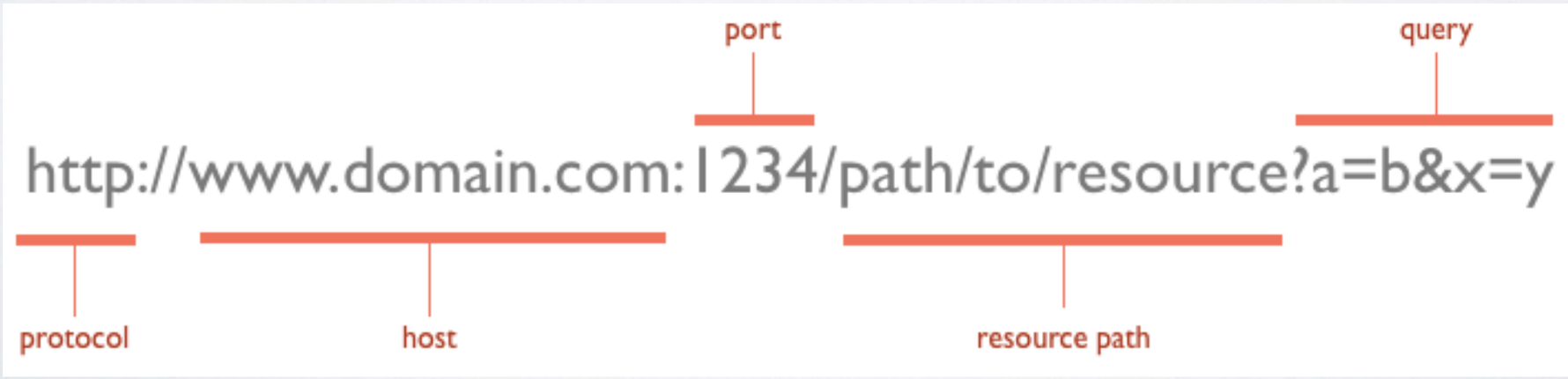
FINDING THINGS

- We find things by using their address.
- A web address is a **Uniform Resource Locator (URL)**
  - e.g. https://www.simonwells.org/publications/

Edinburgh Napier UNIVERSITY

URLS

- The request message is at the heart of communications via the web
- A request will be sent to an HTTP server with respect to a URL
- The URL has structure (which we will exploit later)
- **Protocol** is usually http (but can also be https)
- **Port** defaults to 80 but can be set explicitly in the URL
- **Resource Path** is the local path to the resource on the server



Edinburgh Napier UNIVERSITY

HTML

- We should all be familiar with the H**yperText Markup Language** by now.

- Another text based tool. This time, for describing the structure of documents (with style via css & interactivity va JS).

```
<!doctype html>
<head></head>
<body>
    <h1>Hello World</h1>
</body>
</html>
```

Edinburgh Napier UNIVERSITY

SUMMARY

- Protocols & languages form the basis for the Web: HTTP, URLs, HTML, CSS, JS

- These all build on existing infrastructure for networking: the wider Internet

- Important Takeaway: HTTP is a client-server protocol based on requests & responses from clients to servers & focussed upon specific URLs. If the request came from a browser then the response usually returns a document which is rendered for the user to view.

Edinburgh Napier
UNIVERSITY

PART #2: PRACTICAL WORK

Edinburgh Napier
UNIVERSITY

IMPLEMENTING WEBSITES

- Not going to write an entire HTTP server or client at this point.
- Will use a Python micro-framework called Flask:
- Let's us build dynamic Websites with a focus on the elements of HTTP
- Includes a development server to run our code
- Keeps things simple but doesn't hide the underlying HTTP so we can understand what we're doing (hopefully ;)

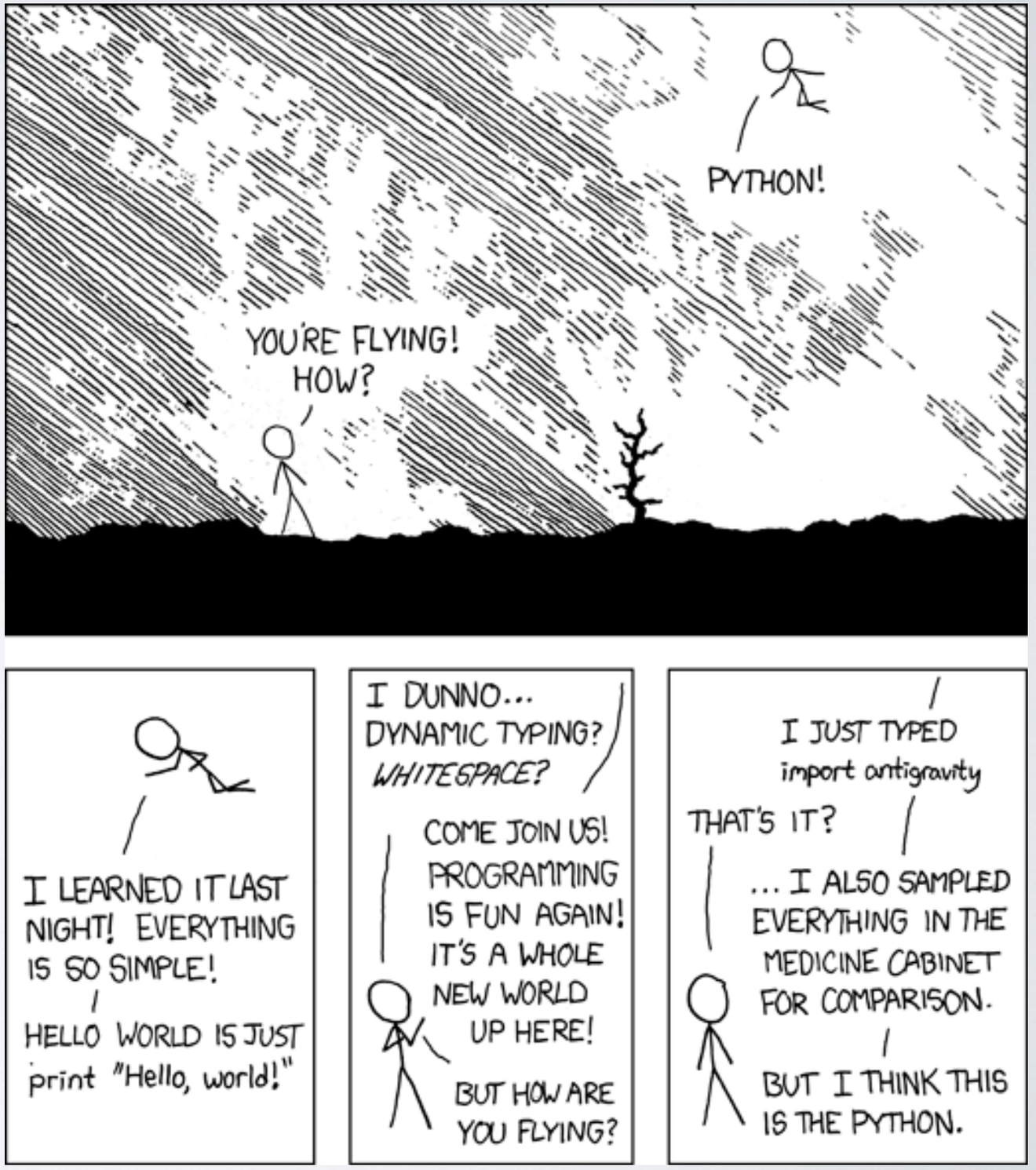
Edinburgh Napier UNIVERSITY

PYTHON

- There are many languages that you can do server-side web development in. Challenge: *Is there a language that couldn't be used to build an HTTP server?*

- Python is a really popular programming language: multi-paradigm, lots of libraries. Fairly stable. Used a lot in data engineering, web devlopment, science, research, industry…

- Has an awful lot of libraries that enable you to do pretty much everything you might need to.



Edinburgh Napier UNIVERSITY

PYTHON-FLASK

- Enables you to create a Web application according to the WSGI standard.
- Includes a web server with hot reload (debug mode) to make development easier
- Exposes much of HTTP so that you can do things like:
  - Build hierarchies of URLs (where each URL is the address of a **resource**)
  - Interact with those resources in different ways from a client
  - Build dynamic web pages (HTML, CSS, JS) and use templates to reduce work where pages are similar
  - Create error pages, return status codes, manage sessions, …
- Via the host language (Python):
  - Connect to databases, Run background tasks, Encrypt data, Hash passwords, + anything Python can do.

Edinburgh Napier UNIVERSITY

UV

- uv is a tool to make working with Python and Python libraries easier: https://docs.astral.sh/uv/

- Enables you to use different versions of Python and different versions of the libraries you need.

- Isolates each app into a project with its own Python and libraries.

Edinburgh Napier
UNIVERSITY

“HELLO NAPIER”

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"
```

WRAPPING UP

- We've taken a whistle-stop tour of modern, dynamic, We development from a programmers perspective. Including:
  - HTTP, URLs, HTML, CSS, JS, Python, Flask, uv
- There's lot's more to see & We'll be returning to some of these topics in more detail later.