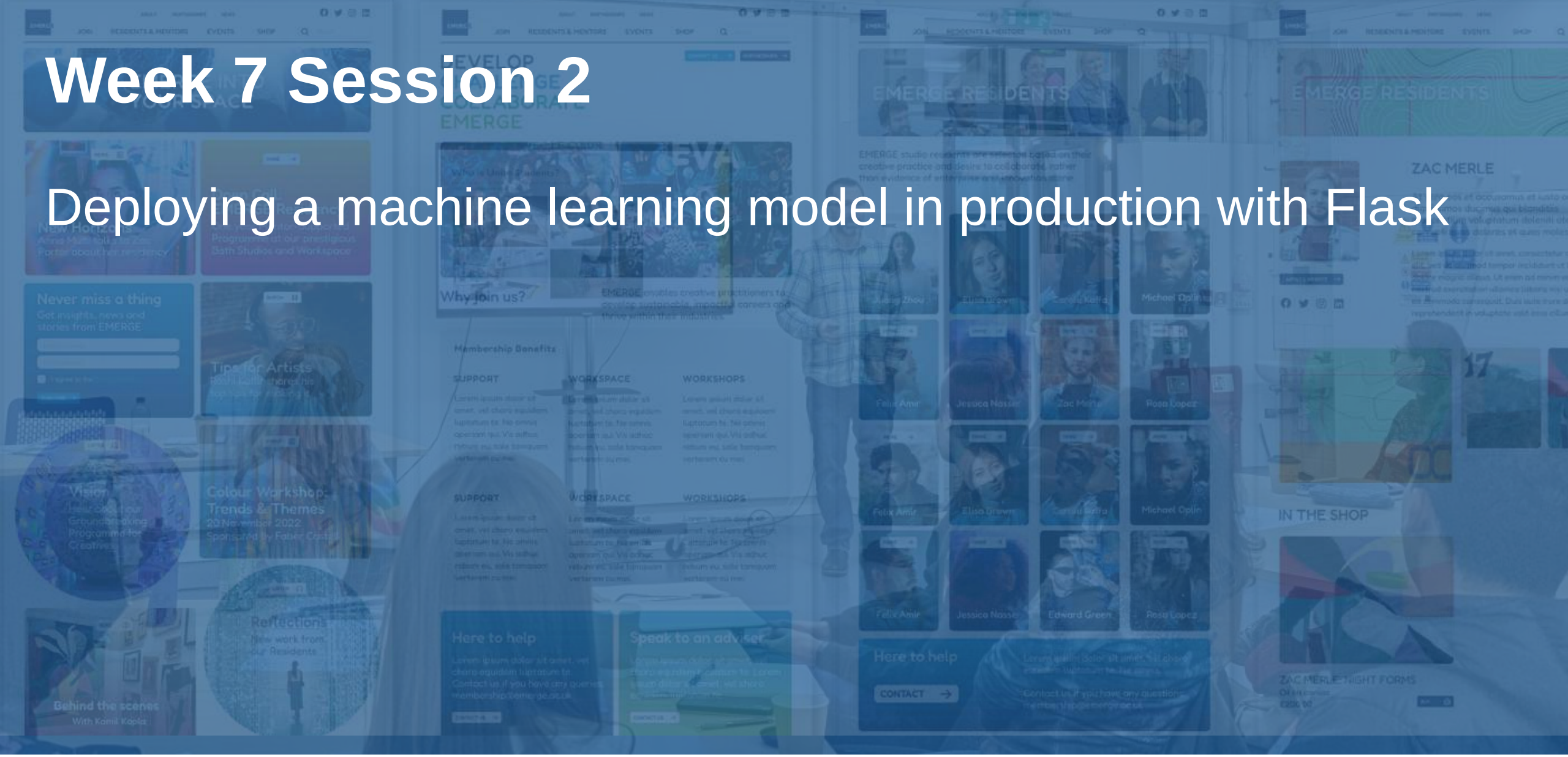
Week 7 Session 2

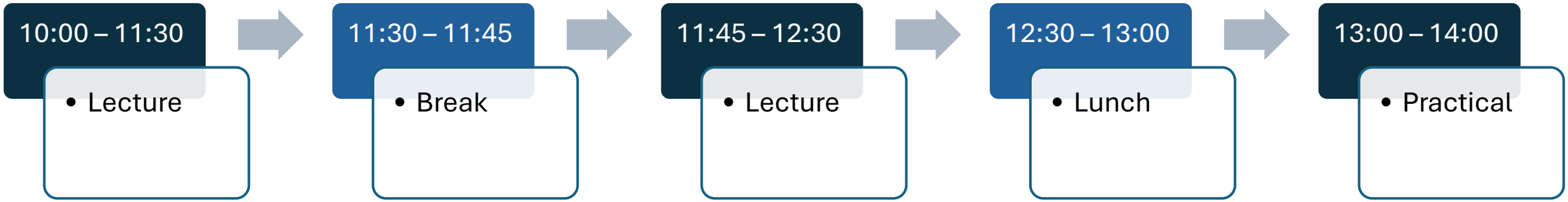
Deploying a machine learning model in production with Flask



Topics

- The mechanics behind how the internet works, and the world wide web.
- Anatomy of a basic web page using HTML and CSS.
- How a web application works.
- Using Flask with Python to operationalise a trained ML model.

Timing


10:00 – 11:30
• Lecture
11:30 – 11:45
• Break
11:45 – 12:30
• Lecture
12:30 – 13:00
• Lunch
13:00 – 14:00
• Practical

Keyword	Description
WWW	World Wide Web: a combination of all resources and users on the Internet that are using the Hypertext Transfer Protocol (HTTP). A broader definition comes from the World Wide Web Consortium (W3C): "The World Wide Web is the universe of network-accessible information, an embodiment of human knowledge."
HTML	HTML (HyperText Markup Language) is the standard markup language used to create web pages and web applications, structuring content on the web.
Tags	An HTML tag is a piece of markup language used to indicate the beginning and end of an HTML element in an HTML document. As part of an HTML element, HTML tags help web browsers convert HTML documents into web pages.
Stylesheet	A stylesheet, specifically a Cascading Style Sheet (CSS), is a file used to control the look and formatting of a document written in HTML, such as layout, colours, and fonts.
CSS	CSS stands for Cascading Style Sheets · CSS describes how HTML elements are to be displayed on screen, paper, or in other media
Pickle File	A pickle file in Python is a serialised file used for storing the binary representation of Python objects for later use or transmission over a network.
GET	The most common method. A GET message is sent, and the server returns data
POST	Used to send HTML form data to the server. The data received by the POST method is not cached by the server.

The Internet

World’s most popular computer network

- Created in 1969 as an academic project.
- Transformed into a commercial network in the 1990s.
- Over 2 billion users worldwide.

Decentralised nature of the internet

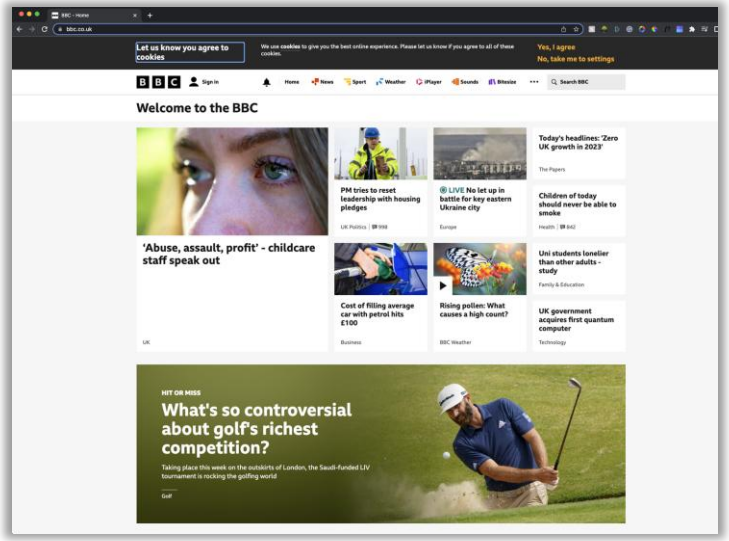
- No single entity has ownership or control.
- Operated by numerous organisations through voluntary agreements.

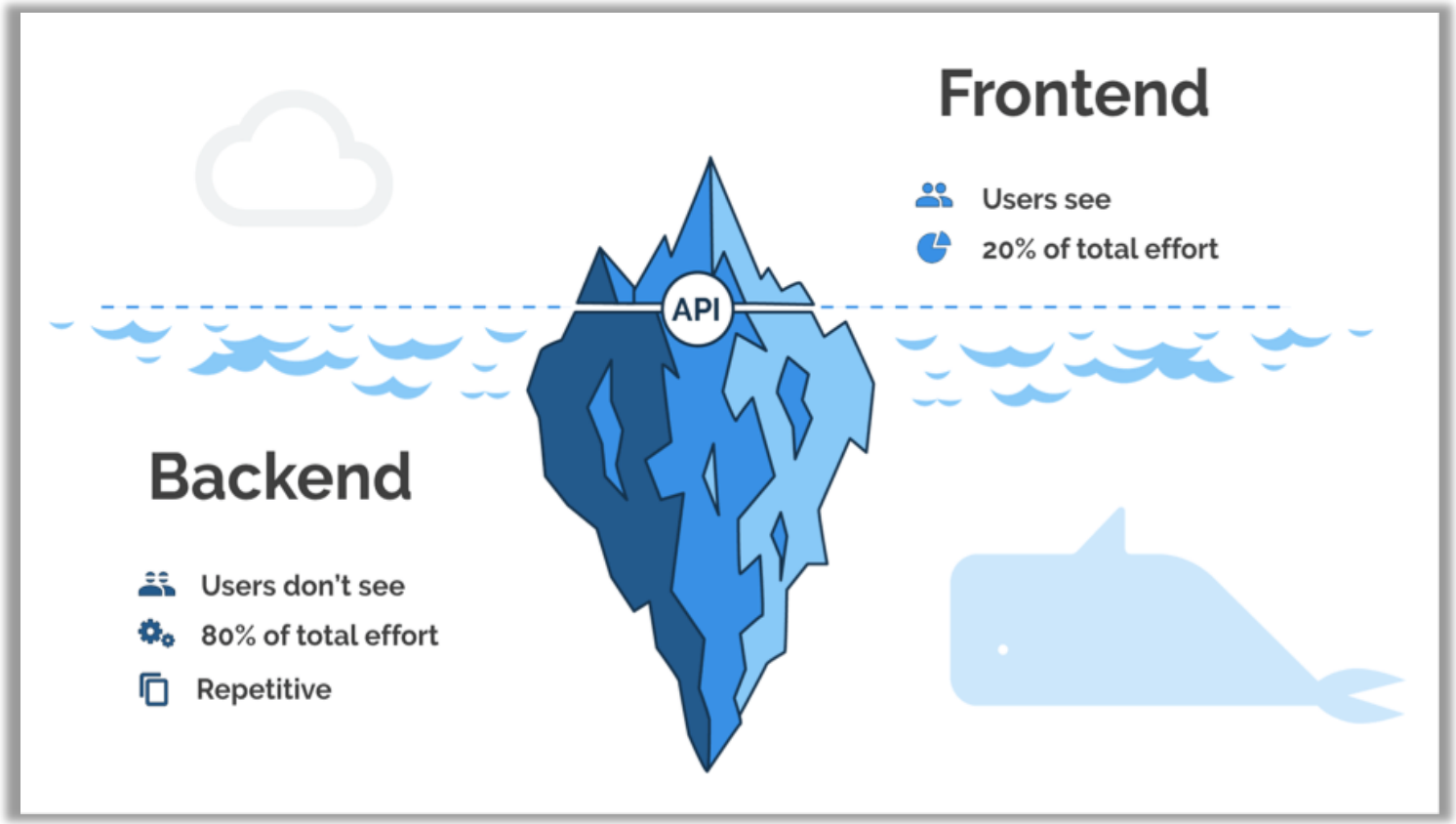
Web browsers are commonly used for accessing content.

- The web is one among many applications on the internet.

Displaying a Webpage

- Web browsers
- Front end pages
- Backend code
- Databases





Front end web pages

HTML

- Not a programming language.
- Creates and structures a webpage.
- First thing a browser loads.

CSS

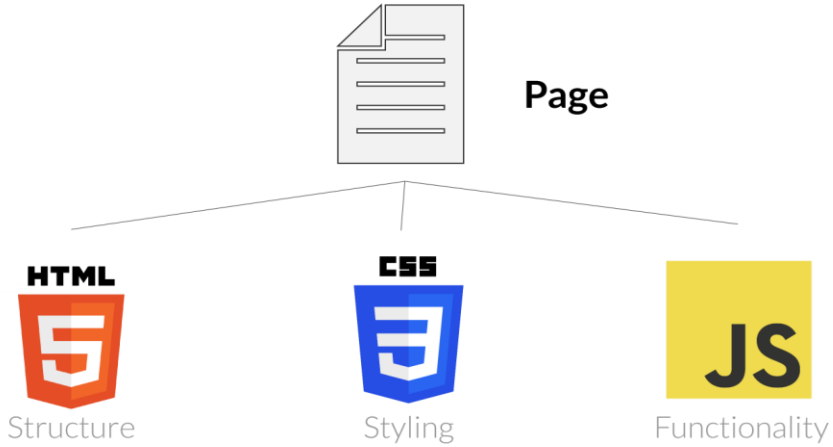
- Describes how HTML elements are displayed on screen.
- Different screen layouts e.g. desktop, tablet, mobile phone.
- Controls the layout of multiple elements.

JavaScript

- Provides interactivity to webpages.

Frameworks

- Combine features into one package.
- Bundle up HTML, CSS, & JS & provide a way to build a website that is more efficient and effective.
- Flask is a python library that enables us to do this, even though traditionally python is a back-end programming language.



Building a web page

HTML

HTML

1. Go to https://codepen.io/jargonaut/pen/RNwLERK
2. Copy the HTML and paste into a new file in VS Code.
3. Save the file on your Desktop as `portfolio.html`.
4. Navigate to the file, right-click, and open with your usual web browser.
5. In VS Code, edit the Achievements section to make it about you, and what you have learned on the bootcamp.
6. Refresh the browser window to see the changes.

Building a web page

CSS

CSS

1. Go to https://codepen.io/jargonaut/pen/abrovzM
2. Copy the CSS and paste into a new file in VS Code.
3. Save the file on your Desktop as `portfolio.css`.
4. In `portfolio.html`, add these lines at the top:

```
<head>
<link rel="stylesheet" type="text/css"
href="portfolio.css">
</head>
```

5. Refresh the browser window to see the changes.
6. Try editing some of the options in `portfolio.css`.
7. Refresh the browser window to see how your changes look.

HTTP: HyperText Transfer Protocol

The basis for data communication on the World Wide Web

Request	Purpose
GET	The most common method. A GET message is send, and the server returns data
POST	Used to send HTML form data to the server. The data received by the POST method is not cached by the server.
HEAD	Same as GET method, but no response body.
PUT	Replace all current representations of the target resource with uploaded content.
DELETE	Deletes all current representations of the target resource given by the URL.

An HTTP request is a message sent by a client (usually a web browser) to a server to request a specific resource (like a web page, an image, a video, etc.) from the server.

HTTP requests can be of various types, including GET (retrieve data), POST (submit data), PUT (update data), DELETE (remove data), and others. Each type has a specific purpose.

These requests are typically initiated by a user interacting with a web page or by a script running on a web page.

BREAK - 15 minutes

Using a framework

Flask

Flask

Flask is a micro web framework:

- Written in Python,
- Light and easy to use,
- Can include interactivity.

For our purposes, the framework provides a way to share our ML models.

Further Information:

https://www.geeksforgeeks.org/flask-tutorial/

By Armin Ronacher - http://flask.pocoo.org/static/logo/flask.svg
Copyrighted free use, https://commons.wikimedia.org/w/index.php?curid=19501815

Flask – Decorators

- Decorators are a significant part of Flask.
  - Route HTTP requests to specific sections of code.
  - Allow Flask to handle different URLs and HTTP methods n a clean and efficient way.
- Flask decorator is app.route() method.
  - Change decorator method as required.

Request	Purpose
GET	The most common method. A GET message is send, and the server returns data
POST	Used to send HTML form data to the server. The data received by the POST method is not cached by the server.
HEAD	Same as GET method, but no response body.
PUT	Replace all current representations of the target resource with uploaded content.
DELETE	Deletes all current representations of the target resource given by the URL.

```
@app.route('/predict',methods=['POST'])
```

```
@app.route('/predict',methods=['GET'])
```

Flask – HTTP Requests

- HTTP protocol defines different methods for retrieving data from a specified URL.
- GET, POST, PUT, DELETE, and others
- Each method performs a different action on the specified URL.
- Decorators route HTTP requests to specific sections of code.
  - E.g. @app.route('/')
    - Route a GET request:
      - To the root URL,
      - And to its decorated function.

Request	Purpose
GET	The most common method. A GET message is send, and the server returns data
POST	Used to send HTML form data to the server. The data received by the POST method is not cached by the server.
HEAD	Same as GET method, but no response body.
PUT	Replace all current representations of the target resource with uploaded content.
DELETE	Deletes all current representations of the target resource given by the URL.

```
10  # Define home route and its corresponding request handler
11  @app.route('/')
12  def home():
13      return render_template('index.html')
```

Practical

1. Setting up Flask
2. Creating a simple app
3. Creating an interactive chart element
4. Training a machine learning model
5. Deploying a machine learning model

Practical

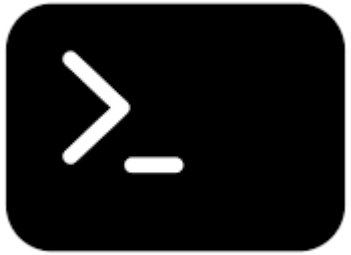
1. Setting up Flask
2. Creating a simple app
3. Creating an interactive chart element
4. Training a machine learning model
5. Deploying a machine learning model

Setting up Flask

We're not working only in Jupyter for this exercise, because we will be running Python scripts (programmes) from the command line.

It's good practice to have different environments for projects that may need different versions of modules and dependencies.

1. Open a Terminal or Command Line window and run the following commands:
    - `pip install virtualenv`
    - `mkdir newproj`
    - `cd newproj`
    - `python –m virtualenv venv`
    - `venv\scripts\activate`
    - `pip install Flask`
    - `pip install plotly`



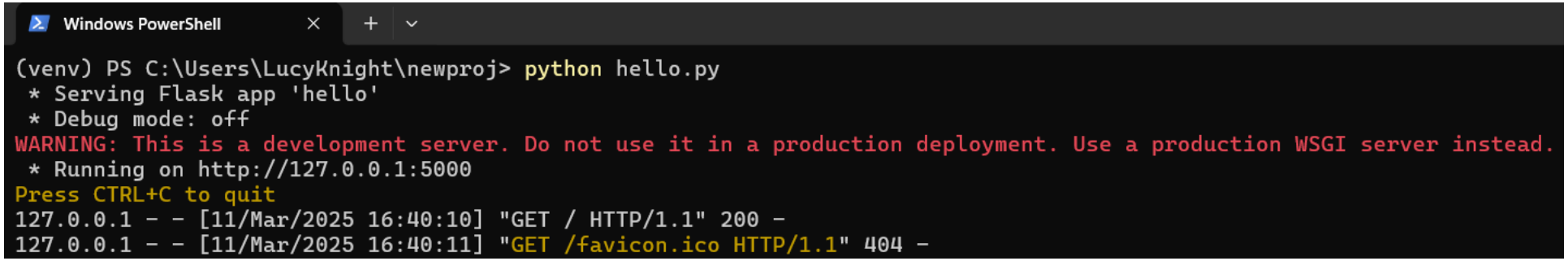
NOTE you may also need to reinstall other modules in this virtual environment, but we will deal with those as they come up!

Practical

1. Setting up Flask
2. Creating a simple app
3. Creating an interactive chart element
4. Training a machine learning model
5. Deploying a machine learning model

Creating a simple app

1. Fetch the file `hello.py` from the GitHub repository.
2. Save this file to the `newproj` folder you created in the last step.
3. Return to the terminal window and type the following:
   - `python hello.py`



4. Open a browser window and type this URL into the address bar:
   - `127.0.0.1:5000`

Practical

1. Setting up Flask
2. Creating a simple app
3. Creating an interactive chart element
4. Training a machine learning model
5. Deploying a machine learning model

Creating an interactive chart element

1. Fetch the files `chart.py`, `bar.html` and `employees.csv` from the GitHub repository.
2. Save `chart.py` and `employees.csv` to your `newproj` folder.
3. Inside the `newproj` folder, create a new folder named `templates`.
4. Save the `bar.html` file inside the `templates` folder.
5. Return to the terminal window and type the following:
   - `Ctrl-C`
   - `python chart.py`
6. Refresh the browser window that you opened for the last example:
   - `127.0.0.1:5000`
7. You should now see a bar chart.

LUNCH - 30 minutes

Practical

1. Setting up Flask
2. Creating a simple app
3. Creating an interactive chart element
4. Training a machine learning model
5. Deploying a machine learning model

Training a machine learning model

1. Fetch the `model.ipynb` Jupyter notebook from the GitHub repository.
2. Save the notebook in your `newproj` folder.
3. Open the notebook and review the code.
4. Run the cells, and check that you now have a new file in your `newproj` folder named `model.pkl`.

Practical

1. Setting up Flask
2. Creating a simple app
3. Creating an interactive chart element
4. Training a machine learning model
5. Deploying a machine learning model

Deploying a machine learning model

1. Fetch three more files from the GitHub repository:
   - `app.py`
   - `request.py`
   - `Salary_data.csv`
   - `index.html`
2. Save the Python files and the CSV to your `newproj` folder, and the HTML file to your `templates` folder.
3. Return to the terminal window and type the following:
   - `Ctrl-C`
   - `python app.py`
4. Refresh the browser window that you opened for the last example:
   - `127.0.0.1:5000`

Thank you!

Any questions?

Learner journal (2 minutes)

- What was the highlight of my week? What was meaningful for me?
- What is one thing I can do to set myself up for success next week?
- What new skill or idea excited me the most this week, and how can I apply it further?
- What strategies or tools helped me to stay focused and motivated?
- What am I now curious about?

Next week:

- Tuesday: Working on your final presentations.
- Wednesday: Presentations!
  - Sion Hill Campus
  - Room T1.11
  - 10:00am to 3:00pm