SECTION 15: WEB SCRAPING WITH PYTHON - 1 hour 40 minutes, 9 parts

1/9 Introduction to Web Scraping

- **automating the gathering of data from a website**
  - -> techniques for automating the gathering of data from a website
  - -> copying and pasting in information from a website is too tedious
  - -> we want to use Python modules to gather this data
  - -> web scraping tasks <- downloading images off of a website
  - -> understanding the basic concepts of how a website works
  - -> how the front-end of a website delivers information to a website <- HTML, CSS and JavaScript
  - -> we need to be able to understand what the browser is telling us

  - **-> how the browser loads a website**
    - **-> wikipedia**
      - -> open the computer
      - -> connect to the internet
      - -> then the browser connects and converts the information in a human readable format
      - -> the server is sending HTML to the browser
    - -> reading the HTML / CSS / JS using a web program to grab what we want
    - **-> searching an HTML document and finding parts of it which we want to use**
    - -> we can convert these to different Python objects
  - -> to perform web scraping effectively

  - **-> outline**
    - -> web scraping rules
    - -> limitations of web scraping
    - -> basic HTML and CSS
  - -> we want to be able to find what we are looking for in the code

- **web scraping rules**
  - -> get permission before scraping
  - -> your IP could get blocked if you make too many requests
  - -> check the permissions / rules / guideline pages before scraping
  - -> check the laws / licences for the website before scraping a certain webpage etc

- **limitations**
  - -> every web scraping script is unique
  - -> every websites HTML is unique to that website
  - **-> so we need to adapt the script to fit one website**
  - **-> web scraping scripts are static because they apply to one HTML page**
  - -> generalising the skillset of using Python to perform the scraping
  - -> looking up information and generalising the process of web scraping for unique situations

- **front end components of a website**
  - -> HTML
  - -> CSS
  - -> JavaScript
  - -> the browser is showing us something human readable
  - -> the browser doesn't show us the source code
  - -> extracting the raw code from those documents
  - -> this shows us the HTML / source code

- -> HTML is used to create the structure and content of a webpage
  - -> for the information on the website
- -> CSS <- Cascading Style Sheets
  - -> for the design and style of a webpage
- -> JS
  - -> for the interactivity of the webpage

- **for effective basic web scraping**
  - **-> we are scraping the HTML / CSS / JS documents for the webpage**
  - -> we are looking for the HMTL and we can use CSS to help us find the information
  - -> we need the HTML and CSS to scrape the page
  - **-> Python extracts information from the HTML and CSS for webpages**
  - -> we are extracting the information from those webpages

```
<!DOCTYPE html>
<html>
    <head>
     <title>Title on Browser Tab</title>
    </head>
    <body>
        <h1> Website Header </h1>
        <p> Some Paragraph </p>
    <body>
</html>
```

  - **-> HTML**
    - -> Hypertext Markup Language
    - -> views page source
    - -> this can be inspected in a console
    - -> he is going through an example HTML document ->
    - -> we have tags for different elements
    - -> the closing tags use /'s
    - -> there is also the head at the top of the html file
      - -> this contains information about the file
    - -> we also have the body <- this contains headers and paragraphs
    - -> we also have tags which we can search for
    - -> we have paragraph tags
    - -> opening and closing tags, and looking for information between the tags

Complete Python Bootcamp

```
<!DOCTYPE html>
  <html>
    <head>
    <link rel="stylesheet" href="styles.css">
    <title>Some Title</title>
    </head>
    <body>
        <p id='para2'> Some Text </p>
    <body>
</html>
```

  - **-> CSS**
    - -> to change the colours / fonts on the website
    - -> in this example, he's linked to a CSS file int he header of an HTML file
    - -> telling the HTML file where to find the CSS file
    - -> we also have IDs and classes
    - -> we need to know about IDs and classes when it comes to CSS
    - -> ids <- these are for single uses

Complete Python Bootcamp

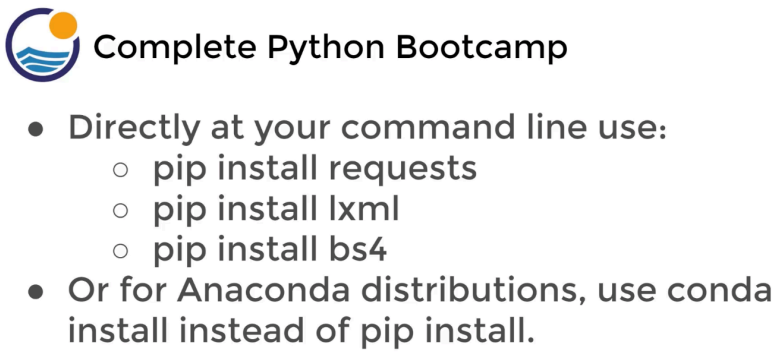
Example of the style.css file:

```
#para2 {
    color: red;
}
```

- **-> style.css example**
  - -> #'s are for IDs
  - -> these are used once for HTML documents
  - -> we have another example of this, which does not contain IDs
  - -> ids are called inside the tags


```
p{
    color: red;
    font-family: courier;
    font-size: 160%;
}
.someclass{
    color: green;
    font-family: verdana;
    font-size: 300%;
}
#someid{
    color: blue;
}
```

- -> it's a dictionary for the property and value
- -> this is denoting it's an ID
- -> the html is linking to the CSS
- -> everything which has that id is going to be coloured red
- -> ids are only used once for HTML documents
- -> classes are used multiple times (or once)
- -> .cool <- these classes are used when we want to add multiple uses across the elements
- -> in a larger CSS file, we can have several types of class



- -> the HTML file contains the information, CSS style
- -> and then the tags locate specific information on the page
- **-> to web scrape with Python, we use BeautifulSoup**
- -> to directly install the requests ->