SECTION 15: WEB SCRAPING WITH PYTHON - 1 hour 40 minutes, 9 parts
5/9 Python Web Scraping - Grabbing an Image

- **grabbing an image from a website**
  - -> grabbing images from a website
  - **-> images have URL links**
  - -> beautiful soup can scan for image tags
  - -> grabbing a list of URLs from a webpage
  - -> check the copyright permissions before grabbing images
  - -> most of the images on Wikipedia are from Wikipedia commons
  - -> it depends on if you are downloading the images and selling them as your own
  - -> he is in an ipynb file
  - -> he's on a Wikipedia page and there are two images on it
  - -> each of them will have URLs
  - **-> making a request to the page, turning it into a soup then inspecting the images and seeing what to look for (a class name, or an element tag)**
  - -> there is more than one way to scrape the information off of the page

  - **-> he is inspecting the images on the Wikipedia page**

```
In [43]: res = requests.get("https://en.wikipedia.org/wiki/Deep_Blue_(chess_computer)")

In [44]: soup = bs4.BeautifulSoup(res.text,'lxml')
```

    - -> one of them has an <img> tag
    - -> we can try and grab the element in Python using the img tag for this
    - -> this returns a list of everything that has an image tag associated with it
    - -> soup.select
    - -> src <- this scraped information contains the URL of the image
    - -> this can be copied and pasted into the webpage

- **if we only want the images in the article, rather than the ones which are on the page**
  - -> he searches for an image, and there is a class associated with it
  - -> the images in the article, rather than the ones which are just on the page
  - -> .thumbimage <- there are two links here
  - -> this returns two URLs, for the images on the page
  - -> we are grabbing the two images in the article
  - -> he grabs the image of the computer and sets it equal to a variable <- this is a specialised tag object
  - -> checking out the image information, and seeing if we can grab it
  - -> he stores the scraped image into a variable, this is a tag object
  - -> we can do a call, which returns a dictionary
  - -> then checking the source for this returns a string
  - **-> he's stored the URL of the image in a markdown cell**
  - **-> running this prints out the scraped image from the URL in the notebook**

- **without using the image directly**
  - -> we can call a certain type of the dictionary / class
  - -> pasting the markdown into the image tags
  - **-> he's used <img> html tags in the markdown in an ipynb file, and this has inserted the image**
  - -> we are making requests, specifically on a URL
  - -> he's used the requests.get() method on the same URL and set it equal to a variable
  - -> the URL used for this is the URL for the browser
  - -> he has entered image_link.content <- this is a series of x00/x00's for example <- this is a

binary representation of the image
- -> we can save this onto the computer in the form of an image
- -> wb <- write with binary
- -> we are opening the jpg image
- -> f.write(image_link.content)
- -> then .close() for it
- -> this allows us to see the jpg file locally
- **-> this saves the file for the jpg in the same directory as the .ipynb file**
- -> downloading images
- -> we can make a request on a website, turn it into a soup and make a class call
- -> class / id calls for the results
- -> we can copy and paste the URL into the browser for this
- -> then an example project