**Section 15: Web Scaping with Python**

**17.03.**

**116. Introduction to Web Scraping**

Web scraping is a general term for automating the gathering of data from a website

Rules of scraping:

Get permission before scraping

If there are many scraping attempts, the IP address may get blocked

Some sites automatically block scraping software

Limitations of scraping:

For each site that is unique a different script is needed

Updates inside a site that are frequent may break the script

While scraping a site, the interaction is manly with HTML and CSS, just perhaps with JS

With HTML and CSS main info can be in a page

Web scaping in Python is done with BeautifulSoup and request libraries: requests, lxml, bs4

**117. Setting Up Web Scraping Libraries**

Before web scraping the elements of interest and the source code should be inspected

For this whole page source code can be viewed and particular elements inspected

**18.03.**

**118. Python Web Scraping - Grabbing a Title**

Get the content of a page with requests.get(url).text

We can parse a web page with soup = bs4.BeautifulSoup(result.text, ‘lxml’)

With soup.select(…) we can get elements from a web site, tags and elements with a css class/id

For example soup.select(‘p’) we get a list with all paragraph tags on which we can call getText()

**119. Python Web Scraping - Grabbing a Class**

soup.select(element) where element could be:

an html tag -> all elements with that tag

#some\_id -> all elements with that css id

.some\_class -> all elements with that class

Tag1 tag2 -> any tag2 inside tag1

Tag1 > tag 2 -> any tag2 directly inside tag1

**120. Python Web Scraping - Grabbing an Image**

In a bs4.element.Tag we can get the values of attributes as from a dictionary

An image can be displayed in a markdown Jupyter cell because it accepts html code

With requests.get(image\_url).content we can get directly the image from web

The image can be stored on the local computer if we write it as a wb file with an img extension