Python module:

In this module we have learned how to use Python in the Jupyter Notebook enviroment. We have learned about what data types there are, how to define variables and functions and which data structures exist such as Lists, Tuples, Dictionaries, Sets. We have also touched on object oriented programming by creating objects that contain both data and functions and can be leveraged for many different purposes, instead of writing a new code for each problem statement. Another important aspect of programming is also around best practice on how to write code, meaning empty spaces and indentation as well as relevant commentary are important for good code.

In the following section we have learned about libraries such as pandas and seaborn in order to process data in data frames and series, but also to apply data visualisation such as scatter, box, line and bar plots and other frameworks in order to communicate derived insights from the data sets.

Focus Data scraping with Python:

Module 5 covers the basics of API, which is the process of requesting and posting JSON as a client to the server, in order to retrieve specific information through the request. We covered the basics of HTML, HTTP commands (GET, POST, Head, PUT, DELETE, PATCH), how to web-scrape contents from any website by identifying the right HTML elements, as well as how connect to a database in PostgreSQL.

Some of the best practices are around web scraping are:

- Import the libraries such as requests, pandas, and BeautifulSoup
- Establish a connection with the URL link by using requests and checking if the response is ‘200’, which is a status message that the connection works.
- Next step is to extract tabular data, by using soup.find to identify the main table elements
- We then extract table headers by defining a rows variable which takes all ‘tr’ elements from the html_doc
- We then create an empty list and define the column_names similar to the table on the HTML website and use a loop statement to append all the relevant elements to our empty list and exclude any empty objects.
- The output is then inserted into a pandas dataframe, which can be further filtered for null-values and formatted
- We can then export the data as a csv of JSON file for future use cases

I personally have struggled the most with identifying the right table elements on the HTML documents as shown below. It is sometimes not as obvious which id element to use and to extract, which might impact the raw data imported:

```
# Extract the contents of the table with the table id.
table = soup.find('table', attrs={'id': 'main_table_countries_today'})
```

Depending on the ‘tr’ elements and how they are formatted, it is important to exclude empty spaces etc:

```
# Note: th = (table header), tr = (table row), and td = table column
rows = table.find_all('tr', attrs={'style': ""})
```

I personally struggle to read the HTML doc and find the relevant formatting and HTML elements in order to process the data correctly.

My personal solution is to try it out and see whichever formatting style or element ID I require to help myself with the Google Chrome website inspector.

While API requests I find much more structured and simple to the user given the available documentation etc. However, when web scraping one has to really understand the HTML elements and identify the elements of the required data sets.

Overall I find Python much more useful and easy to use than SQL based systems. Python libraries are very powerful and provide a variety of functions that can be applied to web scraping, data analysis and visualisation and up to integrating SQL databases. Python is very powerful when used in the right way as it can combine libraries to leverage the most for automation and working at scale.

The ease of exporting and importing files as well as website and API content, make Python in my opinion much more powerful than SQL, Excel or any other tool.