Project 1

Web scarping and data analysis using Python- A pandemic case study –

By Naserddine Hallam

1. Abstract:

In this project, you will apply Python coding skills in developing an application that involves web scraping, database management, and data exploration and analysis. The project consists of scraping Covid-19 data from the web and storing it in a database. The data then need to be analysed and charts/graphs be created.

A written one-page report documenting your analysis, findings, and conclusions drawn need to be produced.

2. Technical Requirements

Your project will consist of writing (loosely coupled) programs/modules that, when put together, allow the user to scrape/store/explore-analyse data.

1.1 Web data Scraping

1. Scrape data from https://www.worldometers.info/coronavirus/,
2. Avoid scraping the same url over and over again by saving the web page locally and scrape the local page instead.
   ~~Persist scraped data in flat files json **and/or** csv data formats~~
3. You should scrape data **for all countries** and for the **past 6 days**.
   a. The web page you are scraping contains three tables (one for today (current day, one for the day before, and the last one for 2 days before).
   b. This means you will need to scrape the website (and thus saving two local html pages) exactly **TWO times**:
   c. The first scraping/local saving will deal with data for current day and the two preceding days (day1, day2 and day3).
   d. Similarly, the second scraping/local saving will deal with data for day4, day5 and day6. (wait three days after the first scraping to perform the second one).

   For example, you can scrape/save data on a Wednesday then on Saturday. The Wed data scraping/saving gives you the past two days as well -Tue and Mon), and then the Sat data scraping/saving gives you the past two days as well -Thu and Fri). Thus, you have your data for 6 days (Mon to Sat).

1.2 Archive data in a Database

4. Create a database called **covid_corona_db_ABCD_EFGH** (where ABCD and EFGH are 4 initials of the surnames of the team members.

Deadline: Thu 22 Nov 2020, 23:59

5. Create a table called **country_borders_table**. Data are in the json file countries_json/country_neighbour_dist_file.json. A sample is show in appendix.
6. Create a table called **corona_table.** The schema for one of your tables is given in the **appendix** for your convenience (can also be extracted from the web site.). you may think about design options:
    - Storing each day in a spate table or all days in one table.
    - you may design and add other fields if you see a need be.
    - Choose wisely the primary key.

1.3 data science

In this part, you need to write modular codes (preferably OOP) to explore and analyse data using pandas and plotting the results using matplotlib.

Plotting process should be coded separately from data exploration to make the code loosely coupled.

1.3.1 Data exploration using pandas

Once your **corona_table** data and **country_borders_table** data is stored in the database, you need to perform data science analysis using **pandas** on the scraped (and stored) data. The process is as follows:

- The user (me) enters/chooses a **country**.
- For 7. And 8., use the **three key indicators** are [**NewCases**,**NewDeaths**,**NewRecovered**].

7. Compare and analyse the evolution through **the 6days** (of the **three key indicators)** for the **country** in question.

8. Compare and analyse the evolution through **the 6days** the scraped data (on the **key indicator NewCases**) for the **country** in question with the **neighbour** with **the longest border**.

9. Compare and analyse the difference between the current Deaths/1M pop for the first **3 days**, for the **country** in question and at **most 2 of its neighbours**. If a country has more than 5 neighbours, disregard those with shorter border distances.

1.3.2 Plotting data

Generate three plotting for the above previous 3 data analysis. You can use any type of plotting provided it make sense. This section has to be coded properly.

10. Use histogram plotting to plot data obtained in 7.
11. Use scatter plotting to plot data in 8.
12. Use histogram plotting to plot/compare results obtained in 9.

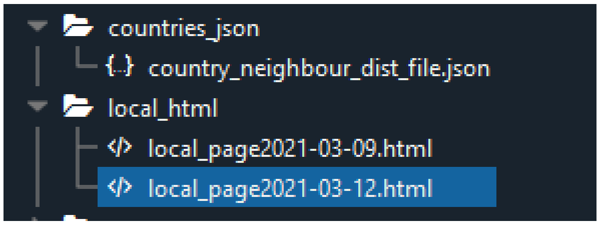
You will need to make the plots as clear as possible: pay attention to the plots titles, legends, colors, the font size, etc.

3. Report

- Produce a (word) report document about the analysis. no more than 2 pages (remember a picture worth 127 words)

4. Project Requirements

- Team of maximum 2 people
- Follow software engineering best practices:
    - Must use the OOP methodology
    - Use exception handling (try/except)
    - Divide your projects into modules
- Some suggestions for modules/classes:
    - CleanData class: mainly clean the numbers
    - DataBaseAPI module- for python mysql database API for SQL commands
    - Database_management module - creating actual database, creating the two tables; countries borders data should be inserted once only
    - Utility module -?
    - ScrapeClass class/module – you can have many useful methods. Should arrange for saving the html page into a local html page.
    - FileIO class/module - if desired to save the html page into a local file
    - Data_explore_analyse (dataScience) module - reading data from the two tables and storing them in dataframes, writing codes for section 1.3.1 and 1.3.2
- Remember, you should have the 2 local html files and json file stored and named as shown in the figure:



```
countries_json
  country_neighbour_dist_file.json
local_html
  local_page2021-03-09.html
  local_page2021-03-12.html
```

5. Submission and deadline

1. Project duration (15 days): deadline as stated in Moodle
2. **Submit to Moodle** a Zip file containing
    a. Word document for the analysis
    b. Required python files
    c. A Readme Text file detailing the content of your project and how to run your program(s)

6. Appendix

Main table headings

The table headings are given in the following list:

[‘date’, 'rank', 'Country,Other', 'TotalCases', 'NewCases', 'TotalDeaths', 'NewDeaths', 'TotalRecovered', 'NewRecovered', 'ActiveCases', 'Serious,Critical', 'Tot Cases/1M pop', 'Deaths/1M pop', 'TotalTests', 'Tests/1M pop', 'Population', 'Continent', '1 Caseevery X ppl', '1 Deathevery X ppl', '1 Testevery X ppl']

Countries and their neighbouring countries

Check the file **country_neighbour_dist_file.json**

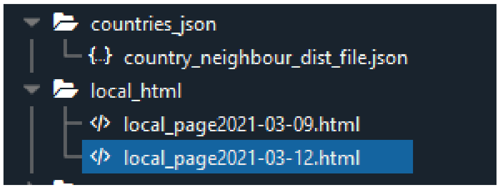
Hints

Where are the data?

All (~200 countries) data for **Now**, **Yesterday** and **day before yesterday** are already in the html page. You just need to locate them.

Avoid sending many requests (DoS attacks)

While doing the tests, it is better to scrape data from the website (once) and then store it in a local (html) files, as shown again in the figure below. Subsequent tests on the page should be directed to the local page rather than the website.


countries_json
country_neighbour_dist_file.json
local_html
local_page2021-03-09.html
local_page2021-03-12.html

Execution scenario:

***** enter your choice:

1: save to local web file

2: scrape from a saved local file

User enters 2

Enter day only (1-31)

User enters 9

Two local files **local_page2021-03-09.html** and **local_page2021-03-12**.html are scraped and data stored in the database.

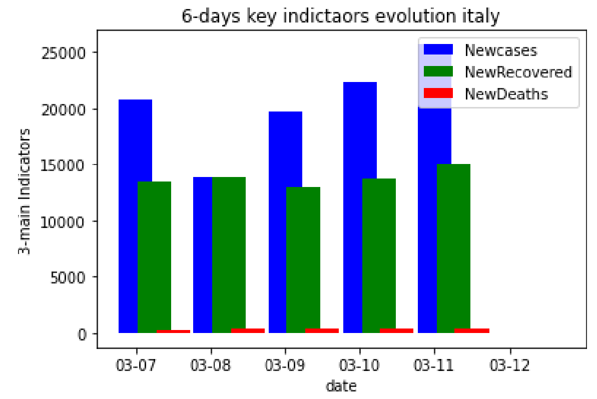
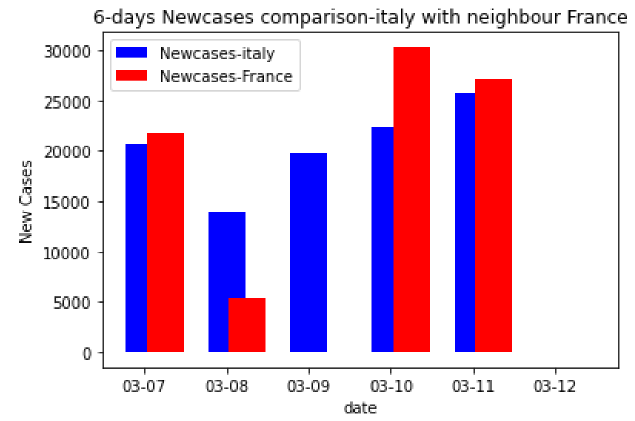
Also data in countries_json/country_neighbour_dist_file.json are stored in a table in the database.

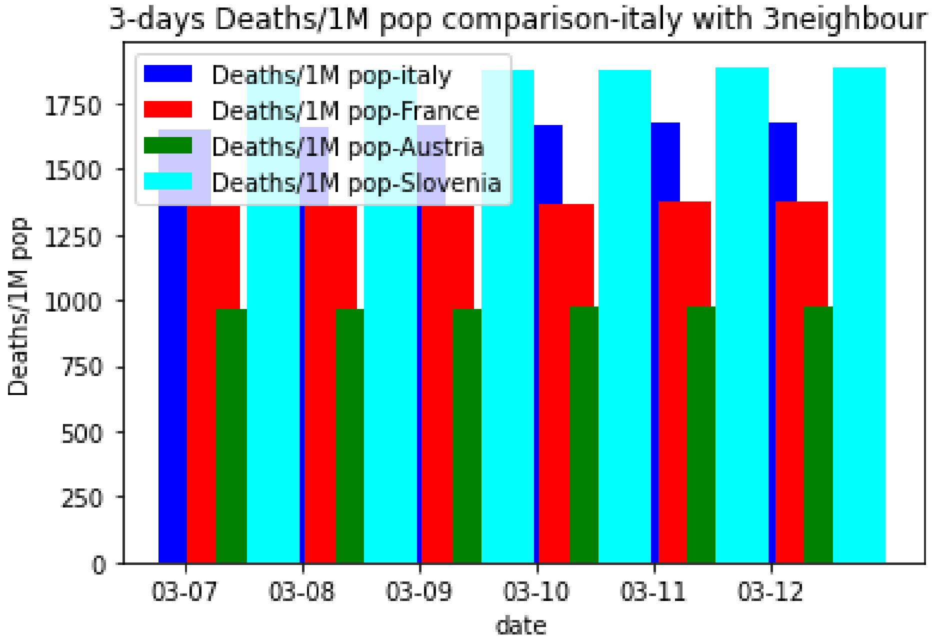
Then user shall be able to run the data_analyse_explore module:

Enter the country to analyse:

User enters: italy

Output

 	
	Italy is compared to France (longest border)



Italy is compared to its 3 neighbours (longest borders considered)