Computing Laboratory I (CS69011)

Autumn Semester 2023

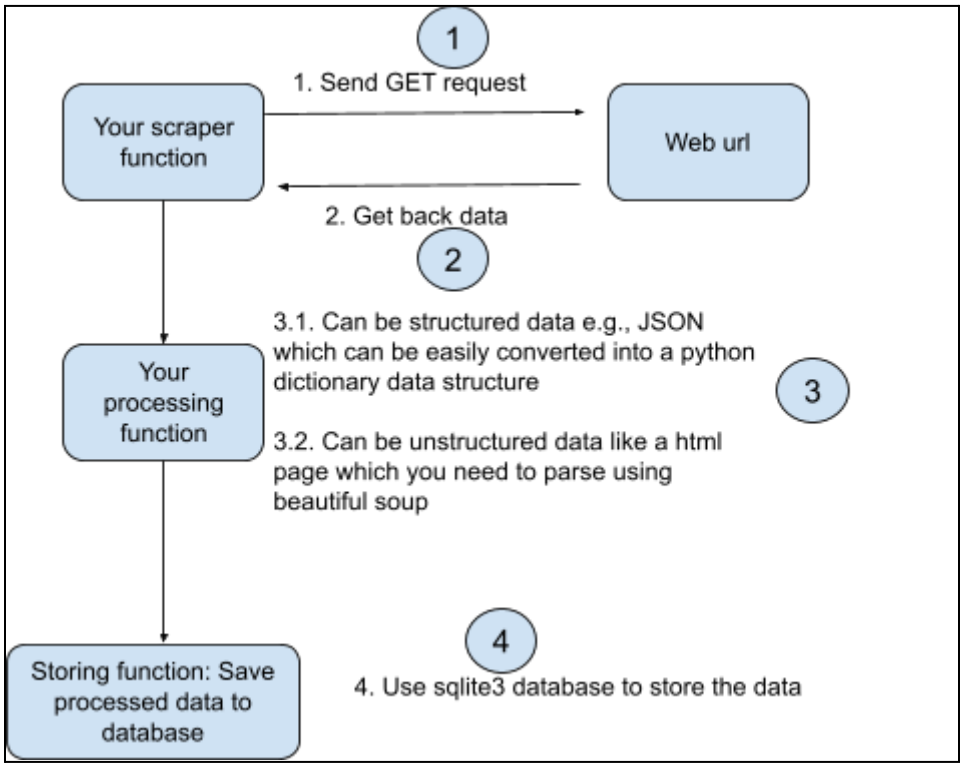
Assignment 6: Web scraping system with python

Assignment date: October 04, 2023

In this lab you will use python to create web scrapers. Web scrapers are programs that can collect data from websites, parse it and store the data (e.g., for analysis later). Recall that a web request can be GET or POST. Using GET requests you can get data from the world wide web (WWW).

So essentially using a GET request, you can get data given by a URL. The URL might be an API (which will send you ”structured data”) or it can be unstructured data (e.g., a HTML webpage). In the later case, you need to “parse” the webpage for collecting the desired data.

This assignment will teach you to use the following basic architecture of a scraper:



Setting up your programming environment

You need to implement the parts of the working scraper system Here are the libraries that you will need:

- For steps 1 and 2: sending GET request and receiving data – python **requests** library or **urllib3** library
- For processing structured data that you receive back: **json** library
- For processing unstructured data: **beautifulsoup4** library which is to be called as **bs4**
- For storing data **sqlite3** library
- For random sampling **random** library

Install each of the libraries as "**pip3 install <libraryname>**" if it's not on your desktop.

Example Code

Finally here is an example code example.py . It's fairly well documented, so first run it, then read it and ask questions. Now you are all set to write some scrapers:

TO BE UPLOADED IN MOODLE

- <rollno>_Assgn_6_1.py and <rollno>_assgn_6_2.py in class
- <rollno>_Assgn_6_3.py and <rollno>_Assgn_6_3.txt in home, by Saturday.

Problem 1: Collecting and Storing structured JSON data

Your task is to collect weather data from an online API and store it in a local SQLite database

1. Use the **requests** or **urllib3** library in Python to make API calls to the **OpenWeatherMap API** (https://openweathermap.org/api). First, you will need to sign up for a free API key. The API allows you to access current weather data for any city in the world (it simply means you need to include the API key in your url before sending to requests)
2. The API returns data in **JSON** format given a latitude and longitude. Extract the following information from the API response:
    - City name
    - Current temperature (in Kelvin)
    - Weather description
    - Humidity (%)
    - Wind speed (meter/sec)
3. Create a SQLite database named 'Weather.db' and a table named 'city_weather' with the following columns:
    - City
    - Temperature

    - Description
    - Humidity
    - WindSpeed
4. Insert the extracted data into the 'city_weather' table (note that you need to create the table only if it does not already exist)
5. Test your system with three to five different cities of your choice, included in a **test** function.

Problem 2: Collecting, storing and processing unstructured data

Your task is to collect information about the different summer olympics from its Wikipedia page and process the data using Python's **urllib3 / requests (to get data)** and **BeautifulSoup** library (for parsing/processing), and store the collected data in a SQLite database.

TASK:

1. Collect the main page of Summer Olympics Wikipedia for this task, the page is here: https://en.wikipedia.org/wiki/Summer_Olympic_Games . Note that you might need to use headers for fetching this page.
2. Now create a database Create a SQLite database named 'OlympicsData.db' and a table named **'SummerOlympics'** with the following columns:
    - Name (e.g. "2012 Summer Olympics", in title of respective wikipedia pages)
    - WikipediaURL
    - Year (the year when its conducted)
    - HostCity (the city where its hosted)
    - ParticipatingNations (List of the participating nations)
    - Athletes (number of athletes)
    - Sports (list of sports)
    - Rank_1_nation
    - Rank_2_nation
    - Rank_3_nation
3. Parse the html from step 1 and extract the individual summer olympics wiki page urls for random 2 olympics from the last 50 years, i.e., from 1968 to 2020. (hint: try to parse the "**List of Summer Olympic Games**" table to get the urls and use **random.sample** for random sampling)
4. For each of the pages of your two selected summer olympics, extract the data (with the help of **BeautifulSoup**) mentioned in step 2 and insert in the database.
5. Then using the database print answers to the following questions:
    - What are the years you chose?
    - What is the average number of countries participating in the two olympics?
    - Print the overlap (i.e., common nations) within <Rank_1_nation, Rank_2_nation and Rank_3_nation> for your chosen two years.

Problem 3: Using multiple processes to speed up

Now we will convert the above code to be run by multiple processes for speed up.

1. Write a handler function that will do the following (reuse the code from previous example)
    a. Collect the main page of Summer Olympics Wikipedia for this task, the page is here: https://en.wikipedia.org/wiki/Summer_Olympic_Games . Note that you might need to use headers for fetching this page.
    b. Now create a database Create a SQLite database named 'OlympicsData.db' and a table named **'SummerOlympics'** with the following columns:
        i. Name (e.g. "2012 Summer Olympics", in title of wikipedia pages)
        ii. WikipediaURL
        iii. Year (the year when its conducted)
        iv. HostCity (the city where its hosted)
        v. ParticipatingNations (List of the participating nations)
        vi. Athletes (number of athletes)
        vii. Sports (list of sports)
        viii. Rank_1_nation
        ix. Rank_2_nation
        x. Rank_3_nation
        xi. DONE_OR_NOT_DONE (a 1 or 0 variable signifying whether fetched or not respectively)
    c. Parse the html from step 1 and extract the individual summer olympics wiki page urls for **TEN** olympics from the last 50 years, i.e., from 1968 to 2020.
    d. insert the WikipediaURL for each row and set DONE_OR_NOT_DONE as 0 for all.
2. Now the handler code will spawn three processes using os.system call. Example of this call

    import os
    os.system("python3 scraper.py&")

    This will run "**python3 scraper.py**" in a separate process.

3. This is what **scraper.py** will do
    a. It will check the database for rows where DONE_OR_NOT_DONE flag is 0.
    b. It will pick a row where DONE_OR_NOT_DONE is 0 (if no such row, **scraper.py** will exit).
    c. For the row chosen, **scraper.py** will first set the DONE_OR_NOT_DONE to 1.
    d. Then it will fetch the wikipedia page using URL in the WikipediaURL column
    e. Next using beautifulSoup it will parse the page and populate the columns mentioned in step 1.b. corresponding row in the database
4. Write a **checker.py** code that can check the database and
    a. Report if all the database rows are populated, i.e., there is no DONE_OR_NOT_DONE which is set to 0 and no process is working (figure out how do you check that?)
    b. If all database rows are populated, then print answers to the following:

i. What are the years you chose?
ii. Which country was within top 3 for the maximum time in your database?
iii. What is the average number of athletes?

5. Finally write a small text document documenting what the percentage speed up in time you actually get by running multiple processes. What is your experiment set up and results?