INTE 21273
DATA SCIENCE

Project – 1

Group no.12

Group Members :-

IM/2021/007 - S.D.S.H. SAMARAKKODI

IM/2021/035 - R.L.L.T. SAMPATH

IM/2021/053 - N.W.I.M. PRASAN

IM/2021/086 - T.M.S.P. DHANAPALA

Task

The assignment involves collecting detailed information from the academic staff page of the Department of Industrial Management at the University of Kelaniya, including names, roles, and specialized subjects, and storing them in CSV files using Python and web scraping techniques.

Approach

The task starts by asking the user to enter a filename for saving the data. Then, a check is made to see if a file with the same name already exists. If it is found, the user is asked whether it should be overwritten or if a new filename should be chosen. Once the file naming is sorted, data is gathered from a specified URL. This is achieved by sending a GET request and using BeautifulSoup to parse the HTML content. Elements containing information about lecturers are looked for, and details such as their name, designation, room number, fax, phone number, email, and specialization(s) are extracted. Additionally, individual lecturer pages are visited to find more details about their specialization(s). Finally, all the gathered data is stored in a CSV file with the chosen filename. The script also includes error handling to deal with any issues that might occur during the scraping process, ensuring that helpful error messages are shown if needed.

Code Explanation

First, the necessary libraries for web scraping and handling CSV files are imported. 'Requests' is used to send HTTP requests to web pages. 'BeautifulSoup' is used to parse HTML content. 'CSV' is used for reading and writing CSV files. 'os' provides functions for interacting with the operating system, such as checking if a file exists. Two separate functions called “is_valid_specialization” and “format_designation” have been created. The “is_valid_specialization” function will check if a given specialization string is valid. It takes a string “txt” as input and checks if it does not contain any invalid strings listed in “invalid_strings”. The “format_designation” function will remove parentheses from a given string.

To incorporate users, the input field is used, and users are asked for a file name. Then, a check is made if a file with the same name already exists in the directory. If so, users are asked if they want to overwrite the existing file or not. If the user enters an invalid input, they are prompted to enter 'y' or 'n'. If the user chooses not to overwrite the existing file, they are prompted to enter a new filename. Otherwise, the loop is broken, and the existing file is overwritten. Global variables such as “page_links”, “all_specializations”, and “lecturers_info_list” are defined to store the extracted data.

A try-except block is incorporated to handle any exceptions that may occur during the web scraping and handling of the CSV file. Using the requests library, a GET request is sent to the URL, and the HTML content is parsed using BeautifulSoup. If there is an error in the request, an exception is raised using the "raise_for_status()" method, and the user is informed using the print statement in the except block. The page links of the lecturers are extracted from the main page and stored in the “page_links” list for further processing. The specializations of each lecturer from their pages are then extracted and stored in the “all_specializations” list. The “is_valid_specialization” function is used to validate the extracted specialization strings to ensure they are valid and do not contain any unwanted strings such as abbreviations or symbols.

The "lecturers" list contains every "div" or "span" HTML element with the class called "sppb-column-addons". Information such as name, designation, room, fax, phone, and email are extracted from each lecturer element using a for loop. The "format_designation" function is used to remove parentheses from the designation string. The 'find' method is used to extract the required information from the HTML content. Then, a check is made if any information is available for the lecturer, and if so, it is appended to the "lecturers_info_list". Lambda functions are used to filter the required information when extracting room, fax, and phone numbers. When extracting email addresses, the 'a' tag with the 'href' attribute containing 'mailto:' is used to filter out email addresses. Since there can be multiple email addresses for a lecturer, a list comprehension is used to extract all email addresses and join them using a comma. After extracting the information, a check is made if any information is available for the lecturer using the 'any' function. If there is any information available, the lecturer's information is appended to the 'lecturers_info_list' to ensure that there are no empty rows in the CSV file.

Then, a check is made if the data was extracted successfully by verifying if the 'lecturers_info_list' is not empty. If there is no data, a message is printed, and the program is exited using the 'exit()' function. If the data was extracted successfully, the data proceeded to be written to a CSV file. Again, a try-except block is used to handle any exceptions that may occur during the writing process. Then, the CSV file is opened in write mode, and a CSV writer object is created using the 'csv.writer' function. The headers are written to the CSV file using the 'writerow' method, which includes the column names such as 'Name', 'Designation', 'Room', 'Fax', 'Phone', 'Email', and 'Specialization(s)'. Then, each lecturer's information in the 'lecturers_info_list' is iterated over, and the information is written to the CSV file using the 'writerow' method. Since the specializations are stored in a separate list called 'all_specializations', they are joined using a comma and added to the row using the 'join' method to concatenate the specializations into a single string separated by commas. Finally, a message is printed to inform the user that the data has been successfully written to the CSV file. In the except block, the 'PermissionError' exception, which occurs when the file is already open in another program, is handled, and the user is informed to close the file and try again. Any other exceptions that may occur during the writing process are also handled, and an error message is printed to inform the user. Finally, the CSV file is closed using the 'close' method to release the resources.

Why this approach?

The given method for online scraping and data extraction is superior to other approaches for several reasons, which are listed below with examples demonstrating each,

1. *User Interaction and Flexibility*

- **Provided Code:** The script incorporates user interaction by allowing users to specify the filename for the CSV file that is produced. Because of its adaptability, users can modify the script to suit their own needs.

- **Alternative Solutions:** Some of the current solutions might not have user interface capabilities, therefore, to handle file conflicts or specify filenames, users would need to manually edit the code. This may not be as user-friendly and intuitive, particularly for non-technical people.

2. *Error Handling*

- **Provided Code:** During file operations and web scraping, the script has strong error-handling methods that use try-except blocks to handle exceptions gracefully. It offers helpful error messages to assist users in resolving possible problems.

- **Alternative Solutions:** Alternative solutions might not handle faults well enough, which could result in unexpected crashes or hard-to-diagnose issues. A program that lacks thorough error handling may be less stable and dependable.

Challenges and Solutions

1. ***The correct element containing information about the lecturers had to be identified.***
   - **Solution:** This was addressed by inspecting the HTML content of the page and identifying the class name of the element that contained the required information.

2. ***Extracting the specializations of each lecturer from their pages posed a challenge.***
   - **Solution:** This was addressed by visiting each lecturer's page and extracting the specializations using the 'is_valid_specialization' function to ensure the extracted strings were valid.

3. ***Some lecturers did not have all the information available, such as room number, fax, or phone number.***
   - **Solution:** This was addressed by checking if the information was available before appending it to the 'lecturers_info_list' to avoid empty rows in the CSV file.

4. ***Information such as "Room", "Fax", and "Phone" did not have a specific class name in the HTML content.***
   - **Solution:** This was addressed by using lambda functions to filter the required information when extracting room, fax, and phone numbers.

5. ***Some lecturers had multiple email addresses or specializations.***
   - **Solution:** This was addressed by using list comprehensions to extract all email addresses and specializations and joining them using a comma to concatenate them into a single string separated by commas.

6. ***Lines with only specializations were present in the CSV file.***
   - **Solution:** This was addressed by checking if any information was available for the lecturer before appending it to the 'lecturers_info_list' to ensure there were no empty rows in the CSV file.

7. ***Same filename conflict occurred.***
   - **Solution**: This was addressed by prompting the user to enter 'y' or 'n' if they wanted to overwrite the existing file. If the user chose not to overwrite, they were prompted to enter a new filename.

8. ***The program crashed if a file was opened in another program.***
   - **Solution:** This was addressed by handling the 'PermissionError' exception, which occurs when the file is already open in another program and informing the user to close the file and try again.

9. ***The program crashed if there was no internet connection or if the URL was incorrect.***
   - **Solution:** This was addressed by using the 'try-except' block to handle exceptions that may occur during the web scraping process and displaying helpful error messages to the user.

10. ***Errors could occur during the writing process, such as the file being open in another program or other exceptions.***
    - **Solution:** This was addressed by using the 'try-except' block to handle any exceptions that may occur during the writing process and display error messages to the user.