To accomplish the objective of extracting textual data articles from the given URLs and performing text analysis,
i follows these detailed instructions:

Data Extraction from URLs:

- Use Python libraries like BeautifulSoup, requests for web scraping to extract textual data from the provided URLs.
- Read the URLs from the "Input.xlsx" file using a library like pandas, numpy.
- Iterate through each URL, extract the article text (excluding headers, footers, etc.), and save it to a text file with the URL_ID as its filename.

Text Analysis:

- Once the textual data is extracted and saved, perform text analysis on each article.
- Calculate the required variables such as Positive Score, Negative Score, Polarity Score, Subjectivity Score, etc., as defined in the "Text Analysis.docx" file.

Output Data Structure:

- Create an output spreadsheet similar to the "Output Data Structure.xlsx" file provided.
- Ensure that the output spreadsheet contains all the input variables from "Input.xlsx" along with the computed variables from the text analysis.

Instructions for Running the .py File for Generating Output:

Dependencies Required:

- Ensuring that I had installed the necessary Python libraries such as pandas, BeautifulSoup, requests, and any other libraries used in the solution.

Code Implementation:

- Writing Python code to implement the data extraction and text analysis tasks as per the approach mentioned above.
- Organized my code into functions or classes for better modularity and readability.

Reading Input Data:

- Use pandas to read the input data from the "Input.xlsx" file.

Data Extraction:

- Implement functions or methods to extract textual data from the URLs provided in the input file.
- Ensure that only the article text is extracted, excluding any headers, footers, or irrelevant content.

Text Analysis:

- Implement functions to perform text analysis on the extracted articles.
- Calculate the required variables according to the definitions provided in the "Text Analysis.docx" file.

Saving Output:

- Save the computed variables along with the input variables into an output spreadsheet, following the format specified in the "Output Data Structure.xlsx" file.

Running the .py File:

- Once the code implementation is complete, save it as a .py file.
- Open a terminal or command prompt and navigate to the directory containing the .py file.
- Run the .py file using Python: BlackCoffer.py

Verification of Output:

- After running the script, verify that the output spreadsheet ("Output Data Structure.xlsx") is generated with the computed variables accurately filled for each article.

List of Dependencies Used in the Python Script:
The script utilizes the following libraries:
1. External Libraries:

- requests: Used for making HTTP requests to retrieve webpages.
- beautifulsoup4: Used for parsing HTML content and extracting data.
- numpy: Used for numerical operations (not strictly necessary in this script).
- io: Used for working with file streams and buffers.
- json: Used for working with JSON data (not used in this script).
- syllables: Used for estimating the syllable count of words.
- nltk: Used for various natural language processing tasks (text normalization, POS tagging, etc.).
- regex: Used for regular expression pattern matching (used for finding personal pronouns).

2. Built-in Libraries:

- pandas: Used for data manipulation and analysis (reading/writing Excel files, creating DataFrames).
- re: Built-in regular expression module (alternative to regex library, not used in this script).

Note:

- The script includes multiple installations of numpy and requests. This is likely redundant and can be optimized by removing duplicate installations.
- The script also downloads resources from NLTK (omw-1.4) for WordNet functionality.