Instruction:

1. Import the required libraries: pandas, requests, BeautifulSoup, numpy, nltk, and string.

2. Read the data from an Excel file named "Input.xlsx" into a DataFrame called 'df'. Drop the 'URL\_ID' column from the DataFrame.

3. Create an empty list called 'data\_list' to store the extracted data from web pages.

4. Iterate through each row in DataFrame 'df', fetch the web page content, extract the title and content text, and merge them together to create a new 'text' column. Append this data to the 'data\_list' list.

5. Create a new DataFrame 'df\_output' using the 'data\_list' with columns 'URL\_ID', 'URL', 'ARTICLE\_TITLE', and 'ARTICLE\_CONTENT'. Save this DataFrame to a CSV file named "output.csv".

6. Read the content from the "output.csv" file into a new DataFrame 'txt'. Drop the 'URL\_ID' and 'URL' columns from 'txt', and convert all columns to strings.

7. Define a function 'text\_process' to process text by removing special Unicode characters, punctuation, and stop words from various categories.

8. Apply the 'text\_process' function to the 'SENTENCE' column of DataFrame 'c' (the processed 'txt' DataFrame).

9. Read positive and negative word dictionaries from the provided text files. Remove punctuation and apply text processing to the words in both dictionaries.

10. Tokenize the 'SENTENCE' column in DataFrame 'c' into a list of words for each row.

11. Calculate the positive score for each sentence by counting the positive words in each sentence and add it as a new column 'POSITIVE SCORE' in DataFrame 'c'.

12. Calculate the negative score for each sentence by counting the negative words in each sentence and add it as a new column 'NEGATIVE SCORE' in DataFrame 'c'.

13. Calculate the polarity score for each sentence using the positive and negative scores and add it as a new column 'POLARITY SCORE' in DataFrame 'c'.

14. Calculate the subjectivity score for each sentence using the positive and negative scores and add it as a new column 'SUBJECTIVITY SCORE' in DataFrame 'c'.

15. Calculate the average sentence length for each sentence and add it as a new column 'AVG SENTENCE LENGTH' in DataFrame 'c'.

16. Calculate the percentage of complex words for each sentence and add it as a new column 'PERCENTAGE OF COMPLEX WORDS' in DataFrame 'c'.

17. Calculate the Fog Index for each sentence using the average sentence length and percentage of complex words, and add it as a new column 'FOG INDEX' in DataFrame 'c'.

18. Calculate the average number of words per sentence for each sentence and add it as a new column 'AVG NUMBER OF WORDS PER SENTENCE' in DataFrame 'c'.

19. Calculate the complex word count for each sentence and add it as a new column 'COMPLEX WORD COUNT' in DataFrame 'c'.

20. Calculate the word count for each sentence and add it as a new column 'WORD COUNT' in DataFrame 'c'.

21. Calculate the syllable count per word for each sentence and add it as a new column 'SYLLABLE PER WORD' in DataFrame 'c'.

22. Calculate the personal pronoun count for each sentence and add it as a new column 'PERSONAL PRONOUNS' in DataFrame 'c'.

23. Calculate the average word length for each sentence and add it as a new column 'AVG WORD LENGTH' in DataFrame 'c'.

24. Save the final DataFrame 'c' to a new DataFrame 'output'.

25. The script will provide the DataFrame 'output' containing the processed data for each sentence, including various calculated metrics and scores.