INSTRUCTIONS

The code might take time during webscraping.

Step 1: Installing dependencies/Importing Libraries:

THE LIBRARIES USED IN THIS PROJECT ARE AS FOLLOWS:

Python version: `Python 3.10.9`

- **Requests** `Requests version: 2.31.0`
- **BeautifulSoup** `Beautiful Soup version: 4.12.3`
- **String**

For NLTK (Natural Language Toolkit), you will need to install the packages separately using NLTK's download utility after installing NLTK itself. You can do this in Python as follows: `NLTK version: 3.8.1`

- **`nltk.download('punkt')`**
- **`nltk.download('stopwords')`**
- **`nltk.download('cmudict')`**
- **`nltk.download('wordnet'`**
- OS
- Regex (re)
- Pandas: `Pandas version: 1.5.3`

```
import requests
from bs4 import BeautifulSoup
import string
import nltk
nltk.download('punkt')
nltk.download('stopwords')
# Download the CMU Pronouncing Dictionary for syllable counting

from nltk.corpus import cmudict
nltk.download('cmudict')

from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')

from nltk.tokenize import word_tokenize,sent_tokenize
from nltk.corpus import stopwords
nltk.download('stopwords')
import os
import re
import pandas as pd
```

My Approach to solving this problem was:

I first did Webscraping using BeautifulSoup on the url's provided in the input excel file and extracted the article content and titles and made a dataframe out of it. I saved the article's content in the dataframe and started to make functions for calculating and saving the output variables(word count,syllable per word etc) in that dataframe . After that I saved the article's content in text files. I performed the text processing steps that were stated in the objective document after that I started appending then in the dataframe and later made a final dataframe which was converted to an excel file.

During the Textprocessing phase I made the functions whose names are as follows:

- **filetextmaker :** It generates text files of the names the text file according to the url_id and the content of the file contains the article
- **remove_url:** It takes text as input and outputs the text without url
- **removingns:** It takes text as input and outputs the text without '\n'
- **remove_html:** It takes text as input and outputs the text without html tags if any
- **remove_punc:** It takes text as input and outputs the text without any punctuation marks
- **remove_stopword:** It takes text as input and outputs the text without the specified stopwords that were given
- **lemmatization:** It takes text as input and outputs lammatized text

- **positivity:** It takes text as input and returns the count of number of positive words in the text
- **negativity:** It takes text as input and returns the count of number of negative words in the text
- **subjectivity:** It takes text as input and returns the subjectivity score of the text
- polarity: It takes text as input and returns the polariity score of the text
- **syllable_count:** It takes text as an input and returns the syllable count in that text
- **sentence_count:** It takes count as an input and returns the count of sentences in the text
- **is_complex:** It takes word as an input and returns the syllable count if it is greater than 3 that is it verifies if a word is complex or not
- **readability:** It takes text as input and makes lists of output variables such as fog index, complex word counts, average sentence list
- **syllableperword:** It takes word as input and returns the count of syllable in word
- **count_pronouns: I**t takes the counts of pronouns in the text and returns the count of pronouns
- **nltstopwordremover:** It takes text as input and outputs the text without stopwords by using the nltk stopwords library for english language
- **word_count:** It takes text as an input and calaculates the number of words in text and returns it
- **avgwordlen:** It takes text as an input and return the output average word length of the text