```
import numpy as np
import pandas as pd


df=pd.read_excel("Input.xlsx")

import requests
from bs4 import BeautifulSoup
import os

def extract_article(url):

    response = requests.get(url)


    if response.status_code == 200:

        soup = BeautifulSoup(response.content, 'html.parser')


        title = soup.find('title').get_text()


        article_content = soup.find('article')


        article_text = ''
        if article_content:
            for paragraph in article_content.find_all('p'):
                article_text += paragraph.get_text() + '\n'

        return title, article_text
    else:
        return None, None

def save_article_to_file(url_id, title, article_text):

    if not os.path.exists('article_texts'):
        os.makedirs('article_texts')


    filename = f'article_texts/{url_id}.txt'


    with open(filename, 'w', encoding='utf-8') as file:
        file.write(title + '\n\n')
        file.write(article_text)


url = 'https://insights.blackcoffer.com/rising-it-cities-and-their-
impact-on-the-economy-environment-infrastructure-and-city-life-in-
future/'
```

```
url_id = 'insights.blackcoffer_rising-it-cities-and-their-impact-on-
the-economy-environment-infrastructure-and-city-life-in-future'

title, article_text = extract_article(url)
if title and article_text:
    save_article_to_file(url_id, title, article_text)
    print(f'Article "{title}" saved successfully.')
else:
    print('Failed to fetch the webpage or extract article content.')
```

```
Article "Rising IT Cities and Their Impact on the Economy,
Environment, Infrastructure, and City Life in Future - Blackcoffer
Insights" saved successfully.
```

```
!pip install afinn
```

```
Collecting afinn
  Downloading afinn-0.1.tar.gz (52 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0.0/52.6 kB ? eta -:--:--
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 30.7/52.6 kB 1.6 MB/s eta
0:00:01 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 52.6/52.6 kB 957.6
kB/s eta 0:00:00
etadata (setup.py) ... e=afinn-0.1-py3-none-any.whl size=53429
sha256=8ea0b7ca6c2b2c6e588485a78fb8e0495079583c86e6e937a2e131c119d636a
5
  Stored in directory:
/root/.cache/pip/wheels/b0/05/90/43f79196199a138fb486902fceca30a2d1b52
28e6d2db8eb90
Successfully built afinn
Installing collected packages: afinn
Successfully installed afinn-0.1
```

```
import requests
from bs4 import BeautifulSoup
from afinn import Afinn


def extract_text_from_url(url):

    response = requests.get(url)


    if response.status_code == 200:

        soup = BeautifulSoup(response.text, 'html.parser')


        text_content = ' '.join([p.get_text() for p in
soup.find_all('p')])

        return text_content
```

```
    else:
        print("Failed to fetch the webpage.")
        return None


def calculate_positive_score(text):
    afinn = Afinn()
    return afinn.score(text)

url = 'https://insights.blackcoffer.com/rising-it-cities-and-their-
impact-on-the-economy-environment-infrastructure-and-city-life-in-
future/'
text_content = extract_text_from_url(url)

if text_content:
    positive_score = calculate_positive_score(text_content)
    print("Positive Score:", positive_score)
else:
    print("No text content extracted from the webpage.")
```

```
Positive Score: 97.0
```

```
import requests
from bs4 import BeautifulSoup
from afinn import Afinn

def extract_text_from_url(url):
    response = requests.get(url)


    if response.status_code == 200:

        soup = BeautifulSoup(response.text, 'html.parser')


        text_content = ' '.join([p.get_text() for p in
soup.find_all('p')])

        return text_content
    else:
        print("Failed to fetch the webpage.")
        return None


def calculate_negative_score(text):
    afinn = Afinn()
    return afinn.score(text)


url = 'https://insights.blackcoffer.com/rising-it-cities-and-their-
```

```
impact-on-the-economy-environment-infrastructure-and-city-life-in-
future/'
text_content = extract_text_from_url(url)

if text_content:
    negative_score = calculate_negative_score(text_content)
    print("Negative Score:", negative_score)
else:
    print("No text content extracted from the webpage.")
```

```
Negative Score: 97.0
```

```


def extract_text_from_url(url):

    response = requests.get(url)


    if response.status_code == 200:

        soup = BeautifulSoup(response.text, 'html.parser')


        text_content = ' '.join([p.get_text() for p in
soup.find_all('p')])

        return text_content
    else:
        print("Failed to fetch the webpage.")
        return None


def calculate_polarity_score(text):
    afinn = Afinn()

    positive_score = afinn.score(text)
    negative_score = afinn.score(text)

    polarity_score = (positive_score - negative_score) / max(1,
positive_score + negative_score)
    return polarity_score


url='https://insights.blackcoffer.com/rising-it-cities-and-their-
impact-on-the-economy-environment-infrastructure-and-city-life-in-
future/'
text_content = extract_text_from_url(url)

if text_content:
```

```
    polarity_score = calculate_polarity_score(text_content)
    print("Polarity Score:", polarity_score)
else:
    print("No text content extracted from the webpage.")
```

```
Polarity Score: 0.0
```

```
import requests
from bs4 import BeautifulSoup
import re


def extract_text_from_url(url):

    response = requests.get(url)


    if response.status_code == 200:

        soup = BeautifulSoup(response.text, 'html.parser')


        text_content = ' '.join([p.get_text() for p in
soup.find_all('p')])

        return text_content
    else:
        print("Failed to fetch the webpage.")
        return None


def clean_text(text):

    cleaned_text = re.sub(r'[^a-zA-Z\s]', '', text)
    cleaned_text = re.sub(r'\s+', ' ', cleaned_text)
    return cleaned_text.strip()


def count_total_words(text):
    words = text.split()
    return len(words)


url = 'https://insights.blackcoffer.com/rising-it-cities-and-their-
impact-on-the-economy-environment-infrastructure-and-city-life-in-
future/'
text_content = extract_text_from_url(url)

if text_content:
    cleaned_text = clean_text(text_content)
    total_words = count_total_words(cleaned_text)
```

```
    print("Total Words After Cleaning:", total_words)
else:
    print("No text content extracted from the webpage.")
```

```
Total Words After Cleaning: 1583
```

```
!pip install afinn
```

```
Collecting afinn
  Downloading afinn-0.1.tar.gz (52 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0.0/52.6 kB ? eta -:--:--
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 52.6/52.6 kB 1.7 MB/s eta
0:00:00
etadata (setup.py) ... e=afinn-0.1-py3-none-any.whl size=53429
sha256=11dde09d3ac7ae57a494d9f93eeca41a32b8eb8adf807e82a466dc9aad73aa7
9
  Stored in directory:
/root/.cache/pip/wheels/b0/05/90/43f79196199a138fb486902fceca30a2d1b52
28e6d2db8eb90
Successfully built afinn
Installing collected packages: afinn
Successfully installed afinn-0.1
```

```
import requests
from bs4 import BeautifulSoup
import re
from afinn import Afinn


def extract_text_from_url(url):

    response = requests.get(url)


    if response.status_code == 200:

        soup = BeautifulSoup(response.text, 'html.parser')


        text_content = ' '.join([p.get_text() for p in
soup.find_all('p')])

        return text_content
    else:
        print("Failed to fetch the webpage.")
        return None


def calculate_subjectivity_score(text):
    afinn = Afinn()
    positive_score = afinn.score(text)
```

```
    negative_score = afinn.score(text)
    total_words=len(text.split())
    subjectivity_score= (positive_score + negative_score) /
((total_words)+0.000001)
    return subjectivity_score


url='https://insights.blackcoffer.com/rising-it-cities-and-their-
impact-on-the-economy-environment-infrastructure-and-city-life-in-
future/'
text_content = extract_text_from_url(url)

if text_content:
    subjectivity_score = calculate_subjectivity_score(text_content)
    print("Subjectivity Score:", subjectivity_score)
else:
    print("No text content extracted from the webpage.")
```

```
Subjectivity Score: 0.12132582856702574
```

```
import requests
from bs4 import BeautifulSoup
import re


def extract_text_from_url(url):

    response = requests.get(url)


    if response.status_code == 200:

        soup = BeautifulSoup(response.text, 'html.parser')


        text_content = ' '.join([p.get_text() for p in
soup.find_all('p')])

        return text_content
    else:
        print("Failed to fetch the webpage.")
        return None


def calculate_average_sentence_length(text):

    sentences = re.split(r'[.!?]', text)


    num_sentences = len(sentences)
```

```
    total_words = sum(len(sentence.split()) for sentence in sentences)


    if num_sentences > 0:
        average_sentence_length = total_words / num_sentences
        return average_sentence_length
    else:
        return 0


url = 'https://insights.blackcoffer.com/rising-it-cities-and-their-impact-on-the-economy-environment-infrastructure-and-city-life-in-future/'
text_content = extract_text_from_url(url)

if text_content:
    average_sentence_length = calculate_average_sentence_length(text_content)
    print("Average Sentence Length:", average_sentence_length)
else:
    print("No text content extracted from the webpage.")
```

```
Average Sentence Length: 18.847058823529412
```

```
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023.12.25)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.2)
```

```
import nltk
nltk.download('punkt')
nltk.download('cmudict')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package cmudict to /root/nltk_data...
[nltk_data]   Unzipping corpora/cmudict.zip.
```

```
True
```

```
import requests
from bs4 import BeautifulSoup
```

```
import re
from nltk.tokenize import word_tokenize
from nltk.corpus import cmudict


def extract_text_from_url(url):

    response = requests.get(url)


    if response.status_code == 200:

        soup = BeautifulSoup(response.text, 'html.parser')


        text_content = ' '.join([p.get_text() for p in
soup.find_all('p')])

        return text_content
    else:
        print("Failed to fetch the webpage.")
        return None


def calculate_percentage_complex_words(text):

    words = word_tokenize(text)


    cmu_dict = cmudict.dict()


    num_complex_words = sum(1 for word in words if
len(cmu_dict.get(word.lower(), [])) > 2)


    total_words = len(words)


    if total_words > 0:
        percentage_complex_words = (num_complex_words / total_words) *
100
        return percentage_complex_words
    else:
        return 0


url = 'https://insights.blackcoffer.com/rising-it-cities-and-their-
impact-on-the-economy-environment-infrastructure-and-city-life-in-
future/'
text_content = extract_text_from_url(url)
```

```
if text_content:
    percentage_complex_words =
calculate_percentage_complex_words(text_content)
    print("Percentage of Complex Words:", percentage_complex_words)
else:
    print("No text content extracted from the webpage.")
```

```
Percentage of Complex Words: 9.838709677419356
```

```
import requests
from bs4 import BeautifulSoup
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import cmudict


def extract_text_from_url(url):

    response = requests.get(url)


    if response.status_code == 200:

        soup = BeautifulSoup(response.text, 'html.parser')


        text_content = ' '.join([p.get_text() for p in
soup.find_all('p')])

        return text_content
    else:
        print("Failed to fetch the webpage.")
        return None


def calculate_fog_index(text):

    sentences = sent_tokenize(text)


    words = word_tokenize(text)


    avg_words_per_sentence = len(words) / len(sentences)


    cmu_dict = cmudict.dict()


    num_complex_words = sum(1 for word in words if
len(cmu_dict.get(word.lower(), [])) >= 3)
```

```
    percentage_complex_words = (num_complex_words / len(words)) * 100


    fog_index = 0.4 * (avg_words_per_sentence +
percentage_complex_words)

    return fog_index


url = 'https://insights.blackcoffer.com/rising-it-cities-and-their-
impact-on-the-economy-environment-infrastructure-and-city-life-in-
future/'
text_content = extract_text_from_url(url)

if text_content:
    fog_index = calculate_fog_index(text_content)
    print("Fog Index:", fog_index)
else:
    print("No text content extracted from the webpage.")
```

```
Fog Index: 13.00865460267506
```

```
import requests
from bs4 import BeautifulSoup
from nltk.tokenize import sent_tokenize, word_tokenize


def extract_text_from_url(url):

    response = requests.get(url)


    if response.status_code == 200:

        soup = BeautifulSoup(response.text, 'html.parser')


        text_content = ' '.join([p.get_text() for p in
soup.find_all('p')])

        return text_content
    else:
        print("Failed to fetch the webpage.")
        return None


def calculate_avg_words_per_sentence(text):

    sentences = sent_tokenize(text)
```

```
    total_words = sum(len(word_tokenize(sentence)) for sentence in
sentences)


    avg_words_per_sentence = total_words / len(sentences) if
len(sentences) > 0 else 0

    return avg_words_per_sentence


url = 'https://insights.blackcoffer.com/rising-it-cities-and-their-
impact-on-the-economy-environment-infrastructure-and-city-life-in-
future/'
text_content = extract_text_from_url(url)

if text_content:
    avg_words_per_sentence =
calculate_avg_words_per_sentence(text_content)
    print("Average Number of Words Per Sentence:",
avg_words_per_sentence)
else:
    print("No text content extracted from the webpage.")
```

```
Average Number of Words Per Sentence: 22.682926829268293
```

```
import requests
from bs4 import BeautifulSoup
from nltk.tokenize import word_tokenize
from nltk.corpus import cmudict


def extract_text_from_url(url):

    response = requests.get(url)


    if response.status_code == 200:

        soup = BeautifulSoup(response.text, 'html.parser')


        text_content = ' '.join([p.get_text() for p in
soup.find_all('p')])

        return text_content
    else:
        print("Failed to fetch the webpage.")
        return None


def calculate_complex_word_count(text):
```

```
    words = word_tokenize(text)


    cmu_dict = cmudict.dict()


    complex_word_count = sum(1 for word in words if
len(cmu_dict.get(word.lower(), [])) >= 3)

    return complex_word_count


url = 'https://insights.blackcoffer.com/rising-it-cities-and-their-
impact-on-the-economy-environment-infrastructure-and-city-life-in-
future/'
text_content = extract_text_from_url(url)

if text_content:
    complex_word_count = calculate_complex_word_count(text_content)
    print("Complex Word Count:", complex_word_count)
else:
    print("No text content extracted from the webpage.")
```

```
Complex Word Count: 183
```

```
import requests
from bs4 import BeautifulSoup
from nltk.tokenize import word_tokenize


def extract_text_from_url(url):

    response = requests.get(url)


    if response.status_code == 200:

        soup = BeautifulSoup(response.text, 'html.parser')


        text_content = ' '.join([p.get_text() for p in
soup.find_all('p')])

        return text_content
    else:
        print("Failed to fetch the webpage.")
        return None


def calculate_word_count(text):
```

```
    words = word_tokenize(text)


    word_count = len(words)

    return word_count


url = 'https://insights.blackcoffer.com/rising-it-cities-and-their-
impact-on-the-economy-environment-infrastructure-and-city-life-in-
future/'
text_content = extract_text_from_url(url)

if text_content:
    word_count = calculate_word_count(text_content)
    print("Word Count:", word_count)
else:
    print("No text content extracted from the webpage.")
```

```
Word Count: 1860
```

```
import requests
from bs4 import BeautifulSoup
from nltk.tokenize import word_tokenize
from nltk.corpus import cmudict


def extract_text_from_url(url):

    response = requests.get(url)


    if response.status_code == 200:

        soup = BeautifulSoup(response.text, 'html.parser')


        text_content = ' '.join([p.get_text() for p in
soup.find_all('p')])

        return text_content
    else:
        print("Failed to fetch the webpage.")
        return None


def count_syllables(word, cmu_dict):
    if word.lower() in cmu_dict:
        return [len(list(y for y in x if y[-1].isdigit())) for x in
cmu_dict[word.lower()]][0]
```

```
    else:
        return 0


def calculate_syllables_per_word(text):

    words = word_tokenize(text)


    cmu_dict = cmudict.dict()


    total_syllables = sum(count_syllables(word, cmu_dict) for word in
words)


    syllables_per_word = total_syllables / len(words) if len(words) >
0 else 0

    return syllables_per_word


url = 'https://insights.blackcoffer.com/rising-it-cities-and-their-
impact-on-the-economy-environment-infrastructure-and-city-life-in-
future/'
text_content = extract_text_from_url(url)

if text_content:
    syllables_per_word = calculate_syllables_per_word(text_content)
    print("Syllables Per Word:", syllables_per_word)
else:
    print("No text content extracted from the webpage.")
```

```
Syllables Per Word: 1.549462365591398
```

```
import requests
from bs4 import BeautifulSoup
from nltk.tokenize import word_tokenize


def extract_text_from_url(url):

    response = requests.get(url)


    if response.status_code == 200:

        soup = BeautifulSoup(response.text, 'html.parser')


        text_content = ' '.join([p.get_text() for p in
soup.find_all('p')])
```

```
        return text_content
    else:
        print("Failed to fetch the webpage.")
        return None


def find_personal_pronouns(text):

    words = word_tokenize(text)


    personal_pronouns = ["i", "me", "my", "mine", "myself", "you",
"your", "yours", "yourself", "he", "him", "his", "himself", "she",
"her", "hers", "herself", "it", "its", "itself", "we", "us", "our",
"ours", "ourselves", "you", "your", "yours", "yourselves", "they",
"them", "their", "theirs", "themselves"]


    personal_pronoun_count = sum(1 for word in words if word.lower()
in personal_pronouns)

    return personal_pronoun_count

url = 'https://insights.blackcoffer.com/rising-it-cities-and-their-
impact-on-the-economy-environment-infrastructure-and-city-life-in-
future/'
text_content = extract_text_from_url(url)

if text_content:
    personal_pronoun_count = find_personal_pronouns(text_content)
    print("Personal Pronoun Count:", personal_pronoun_count)
else:
    print("No text content extracted from the webpage.")
```

```
Personal Pronoun Count: 38
```

```
import requests
from bs4 import BeautifulSoup
from nltk.tokenize import word_tokenize


def extract_text_from_url(url):

    response = requests.get(url)


    if response.status_code == 200:

        soup = BeautifulSoup(response.text, 'html.parser')
```

```
        text_content = ' '.join([p.get_text() for p in
soup.find_all('p')])

        return text_content
    else:
        print("Failed to fetch the webpage.")
        return None


def calculate_avg_word_length(text):

    words = word_tokenize(text)


    total_length = sum(len(word) for word in words)


    avg_word_length = total_length / len(words) if len(words) > 0 else
0

    return avg_word_length


url = 'https://insights.blackcoffer.com/rising-it-cities-and-their-
impact-on-the-economy-environment-infrastructure-and-city-life-in-
future/'
text_content = extract_text_from_url(url)

if text_content:
    avg_word_length = calculate_avg_word_length(text_content)
    print("Average Word Length:", avg_word_length)
else:
    print("No text content extracted from the webpage.")
```

```
Average Word Length: 4.897849462365591
```

```
import pandas as pd


data = {
    "URL_ID": ["https://insights.blackcoffer.com/rising-it-cities-and-
their-impact-on-the-economy-environment-infrastructure-and-city-life-
in-future/"],
    "POSITIVE SCORE": [97.0],
    "NEGATIVE SCORE": [97.0],
    "POLARITY SCORE": [0.0],
    "SUBJECTIVITY SCORE": [0.12132582856702574],
    "AVG SENTENCE LENGTH": [18.847058823529412],
    "PERCENTAGE OF COMPLEX WORDS": [9.838709677419356],
    "FOG INDEX": [13.00865460267506],
```

```
    "AVG NUMBER OF WORDS PER SENTENCE": [22.682926829268293],
    "COMPLEX WORD COUNT": [183],
    "WORD COUNT": [1860],
    "SYLLABLE PER WORD": [1.5494623655913978],
    "PERSONAL PRONOUNS": [38],
    "AVG WORD LENGTH": [4.897849462365591]
}

df = pd.DataFrame(data)


output_datafile = "Output Data Structure.xlsx"
df.to_excel(output_datafile, index=False)
print("Output file saved as:", output_datafile)
```

```
Output file saved as: Output Data Structure.xlsx
```