

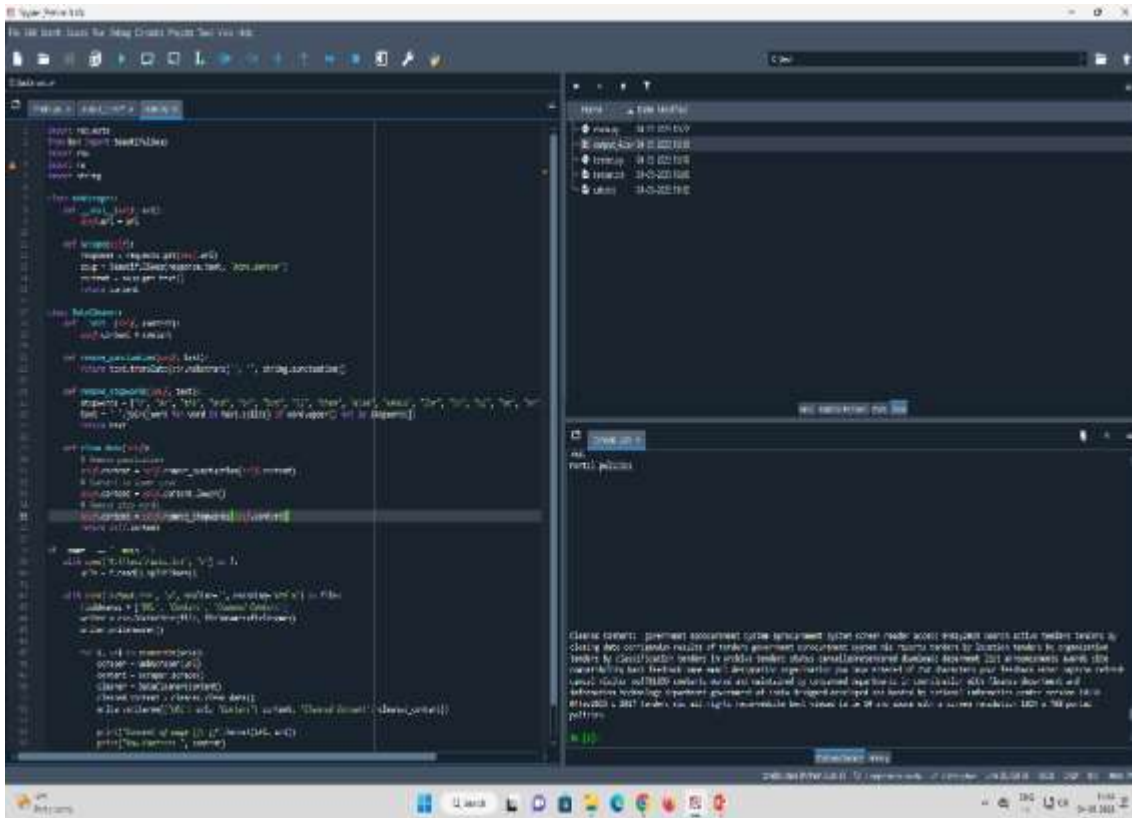
Name : Nevedha VR

I took which you mention in the data set of document “For keyword
extract in Website “

[“https://ieg.worldbankgroup.org/data”](https://ieg.worldbankgroup.org/data)

“https://ieg.worldbankgroup.org/data”

Web scrapping from Website



```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import re
import sys

def get_urls():
    with open('urls.txt', 'r') as f:
        urls = f.readlines()
    return urls

def get_content(url):
    html = urlopen(url).read()
    soup = BeautifulSoup(html, 'html.parser')
    return soup.get_text()

def clean_content(text):
    text = re.sub(r'[^\w\s]', '', text)
    text = text.lower()
    text = re.sub(r'\b\w{1,3}\b', '', text)
    return text

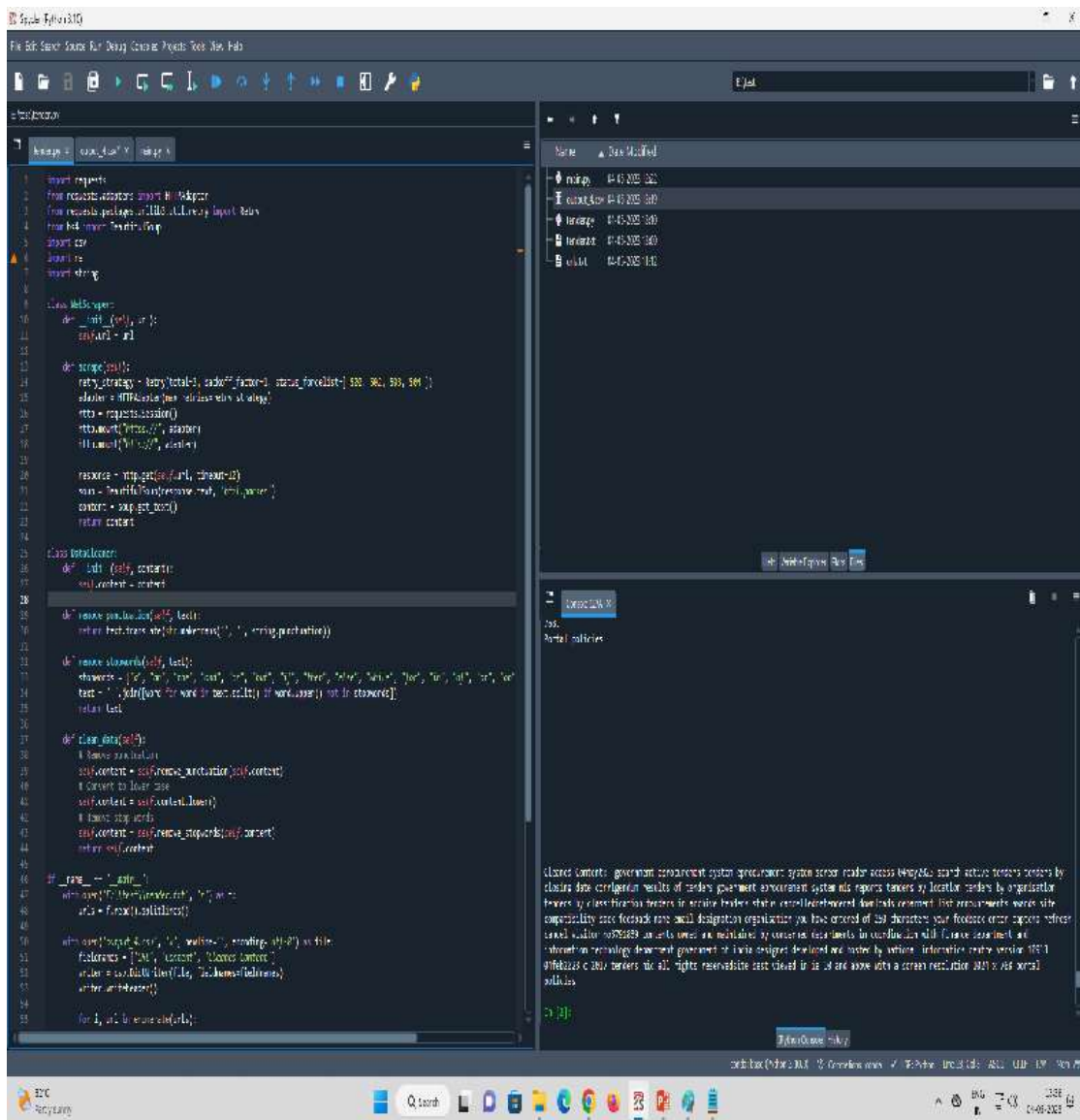
def main():
    urls = get_urls()
    for url in urls:
        content = get_content(url)
        cleaned_content = clean_content(content)
        print(f'URL: {url}')
        print(f'Content: {content}')
        print(f'Cleaned Content: {cleaned_content}')
        sys.stdout.flush()

if __name__ == '__main__':
    main()
```

The screenshot shows a Python script in a code editor. The script defines functions to fetch URLs from a file, retrieve HTML content from each URL, and clean the text by removing punctuation and stop words. The console output shows the raw HTML content and the cleaned text for each URL.

Explanation of code

- This Python code is a web scraper and data cleaner that takes a list of URLs [“https://ieg.worldbankgroup.org/data”](https://ieg.worldbankgroup.org/data)
- from a text file and extracts the text content from each URL's HTML page using the requests and BeautifulSoup libraries. The extracted content is then cleaned by removing punctuation, converting to lowercase, and removing stop words using the string library. The cleaned content is written to a CSV file with three columns: URL, Content (raw extracted text), and Cleaned Content. Finally, the code prints the raw and cleaned content for each URL to the console.

A screenshot of a Python IDE (likely PyCharm) showing a script for web scraping and data cleaning. The script is divided into three main sections: imports, class definitions, and a main function. The imports section includes 'requests', 'BeautifulSoup', and 'csv'. The 'WebScraper' class has methods for making HTTP requests with retries and timeouts, and for parsing the HTML content. The 'DataCleaner' class has methods for removing punctuation and stop words from text. The main function reads a list of URLs from a file, iterates through them, scrapes the content using the 'WebScraper' class, cleans the content using the 'DataCleaner' class, and writes the cleaned data to a CSV file. The IDE interface shows the code editor on the left, a file explorer on the right, and a terminal at the bottom.

```
1 import requests
2 from requests.adapters import HTTPAdapter
3 from requests.packages.urllib3.util.retry import Retry
4 from bs4 import BeautifulSoup
5 import csv
6 import time
7 import sys
8
9 class WebScraper:
10     def __init__(self, url):
11         self.url = url
12
13     def scrape(self):
14         retry_strategy = Retry(
15             total=3,
16             backoff_factor=1,
17             status_forcelist=[400, 401, 403, 404, 500, 502, 503, 504]
18         )
19         adapter = HTTPAdapter(max_retries=retry_strategy)
20         http = requests.Session()
21         http.mount('https://', adapter)
22         http.mount('http://', adapter)
23
24         response = http.get(self.url, timeout=10)
25         soup = BeautifulSoup(response.text, 'html.parser')
26         content = soup.get_text()
27         return content
28
29 class DataCleaner:
30     def __init__(self, content):
31         self.content = content
32
33     def remove_punctuation(self, text):
34         return text.translate(str.maketrans('', '', string.punctuation))
35
36     def remove_stopwords(self, text):
37         stopwords = ['a', 'an', 'and', 'are', 'as', 'at', 'be', 'but', 'by', 'can', 'could', 'did', 'do', 'does', 'for', 'from', 'had', 'has', 'he', 'her', 'his', 'him', 'in', 'is', 'it', 'me', 'of', 'on', 'or', 'over', 'she', 'that', 'the', 'to', 'was', 'we', 'were', 'with', 'you']
38         text = self.remove_punctuation(text)
39         return text
40
41     def clean_data(self):
42         # Remove punctuation
43         self.content = self.remove_punctuation(self.content)
44         # Convert to lower case
45         self.content = self.content.lower()
46         # Remove stop words
47         self.content = self.remove_stopwords(self.content)
48         return self.content
49
50 # Main function
51 def main():
52     # Read URLs from a file
53     with open('urls.txt', 'r') as f:
54         urls = f.readlines()
55
56     # Iterate through URLs and scrape content
57     for url in urls:
58         url = url.strip()
59         scraper = WebScraper(url)
60         content = scraper.scrape()
61         cleaner = DataCleaner(content)
62         cleaned_content = cleaner.clean_data()
63
64         # Write cleaned content to CSV file
65         with open('cleaned_data.csv', 'a') as f:
66             writer = csv.writer(f)
67             writer.writerow(cleaned_content)
```

Explanation of code

This is a Python script for web scraping and data cleaning. It uses the requests library to make HTTP requests to URLs, <https://iege.worldbankgroup.org/data> the BeautifulSoup library to parse the HTML content of the web pages, and the csv library to write the scraped data to a CSV file.

The script defines a WebScraper class that takes a URL and uses the requests library to make an HTTP GET request to the URL. It then uses BeautifulSoup to extract the text content of the HTML response.

The DataCleaner class is defined to clean the text content extracted by the WebScraper. It provides two methods to remove punctuation and stop words from the text.

In the main function, the script reads a list of URLs from a text file, iterates through the URLs, and scrapes the content of each web page. If an exception occurs while processing a page, the script catches the exception and continues to the next page. The scraped data is then written to a CSV file.

The script includes a retry mechanism using the requests library to retry failed requests up to three times in case of certain HTTP status codes. It also includes a timeout of 10 seconds for the HTTP requests to prevent the script from hanging if a page takes too long to respond.

Thank you