Web Scraping with BeautifulSoup: WikiHow Articles

INTRODUCTION:

In this document, we present a Python-based web scraping solution that extracts article titles, subheadings, and paragraphs from WikiHow. The web scraping project involves using libraries like 'requests' to handle HTTP requests and 'BeautifulSoup' for parsing HTML content. The extracted data is then saved into a CSV file, which can be used for further analysis.

CODE BREAKDOWN:

1. IMPORTING LIBRARIES:

We begin by importing the necessary libraries.

'requests' is used to send HTTP requests, and 'BeautifulSoup' from the 'bs4' library helps parse the HTML content we receive from the web pages. 'csv' is used to write the extracted data into a

CSV file. Lastly, ‘re’ is used for cleaning and processing text.

2. FETCHING DATA FROM WIKIHOW:

We loop through 4000 random articles from WikiHow using a loop and the URL 'https://www.wikihow.com/Special:Randomizer', which redirects us to a random article each time it's accessed. We use the ‘requests.get()’ method to retrieve the HTML content of each article.

3. PARSING HTML CONTENT:

Once we have the HTML content, we use BeautifulSoup to parse it. The title of the article is extracted using `soup.find('title')’. The subheadings and paragraphs are extracted from the div tags that have the class 'step', where subheadings are found in the <b> tag, and the remaining text in paragraphs.

4. DATA CLEANING:

To clean the data, we remove any non-ASCII characters and unwanted symbols using regular expressions (‘re’).

This ensures that the data stored in the CSV is clean and well-formatted for analysis.

5. SAVING DATA INTO CSV:

Finally, the extracted data is written to a CSV file. Each row in the CSV contains the article title, a subheading, and the corresponding paragraph.

6. PYTHON CODE:

Below is the Python code used in this project for scraping WikiHow articles and storing the extracted data in a CSV file.

```
import requests
from bs4 import BeautifulSoup
import re
import csv

csv_file_path = '/content/wikiHow.csv'

for count in range(4000):
url="https://www.wikihow.com/Special:Randomizer"
response = requests.get(url)
```

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

    article_title = soup.find('title').text.strip()
    print({article_title})

    subheadings = []
    paragraphs = []

    steps = soup.find_all('div', {'class': 'step'})
    for step in steps:
        subheading_element = step.find('b')
        if subheading_element is not None:

subheading_text=subheading_element.text.strip().
replace('\n', '')
subheading_text = subheading_text.encode('ascii',
errors='ignore').decode()
 subheading_text = re.sub(r'', '', subheading_text)
            subheadings.append(subheading_text)
            subheading_element.extract()
```

```
  for span_tag in step.find_all('span'):
        span_tag.extract()

paragraph_text =
step.text.strip().replace('\n','').replace(',', '')
paragraph_text =paragraph_text.encode('ascii',
errors='ignore').decode()
paragraph_text = re.sub(r'', '', paragraph_text)
         paragraphs.append(paragraph_text)
   if len(subheadings):
      with open(csv_file_path, mode='a', newline='',
encoding='utf-8') as csv_file:
       writer = csv.writer(csv_file)
        for i in range(len(subheadings)):
            writer.writerow([article_title,
subheadings[i], paragraphs[i]])
```

CONCLUSION:

This project demonstrates how to scrape dynamic content from a website like WikiHow using Python, BeautifulSoup, and requests. The data is effectively stored in a CSV file for easy accessibility.

This process can be adapted to scrape data from other websites by adjusting the parsing rules to match the structure of the target site.