Wikipedia Webside WebScrapping

A Project Submitted to the

IT VEDANT Institute, Thane.

Data Science & Data Analytics With AI

Python-Web-Scrapping Project

BY

Omkar Hase

Under the Guidance of

Mr. Sameer Warsolkar

Outline:

1. Choose the Website and Webpage URL

2. Inspect the Website

3. Install the Important Libraries

4. Write the Python Source Code

5. Export the Extracted Data

Steps:

Outline: Steps to Scrape https://en.wikipedia.org/wiki/List_of_highest-grossing_Indian_films
Choose the Website and Webpage URL

Website: https://en.wikipedia.org/wiki/List_of_highest-grossing_Indian_films

Step-by-Step Guide:

1. Import Required Libraries

1. import requests:

- This line imports the `requests` library, which is used to send HTTP requests in Python.
- It allows you to fetch web pages and retrieve their HTML content, enabling you to interact with websites and download their data.

2. from bs4 import BeautifulSoup:

- This line imports the `BeautifulSoup` class from the `bs4` (BeautifulSoup) library.
- `BeautifulSoup` is used to parse HTML or XML documents, making it easy to navigate, search, and extract data from web pages in a structured way.

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

Define the URL of the Website to Scrape

These lines of code send a GET request to the URL `"https://en.wikipedia.org/wiki/List_of_highest-grossing_Indian_films"` using the `requests` library and store the server's response in the `response` variable. The variable `response` will contain the HTTP response data, which can be used to access the webpage content or check request status.

```
url = "https://en.wikipedia.org/wiki/List_of_highest-grossing_Indian_films"
response = requests.get(url)
response
```

```
<Response [200]>
```

Define the Function to Scrape the Webpage

This code uses **BeautifulSoup** (a Python library) to parse HTML content. `response.content` retrieves the raw HTML, and `BeautifulSoup (response.content, "html.parser")` parses it, allowing easy navigation and extraction of specific elements within the HTML structure. The variable `soup` now holds this parsed HTML document

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

```
<!DOCTYPE html>

<html class="client-nojs vector-feature-language-in-header-enabled vector-feature-language-in-main-page-head
er-disabled vector-feature-sticky-header-disabled vector-feature-page-tools-pinned-disabled vector-feature-t
oc-pinned-clientpref-1 vector-feature-main-menu-pinned-disabled vector-feature-limited-width-clientpref-1 ve
ctor-feature-limited-width-content-enabled vector-feature-custom-font-size-clientpref-1 vector-feature-appea
rance-pinned-clientpref-1 vector-feature-night-mode-enabled skin-theme-clientpref-day vector-toc-available"
dir="ltr" lang="en">
<head>
<meta charset="utf-8"/>
<title>List of highest-grossing Indian films - Wikipedia</title>
<script>(function(){var className="client-js vector-feature-language-in-header-enabled vector-feature-langua
ge-in-main-page-header-disabled vector-feature-sticky-header-disabled vector-feature-page-tools-pinned-disab
```

STEP 1 :

This code retrieves all `<td>` elements from an HTML document using BeautifulSoup.

- `soup.find_all("td")` finds all `<td>` tags (table cells) in the HTML.
- `td_elements` stores the list of these `<td>` elements for further processing.

This is useful for extracting table data from a webpage.

```
td_elements = soup.find_all("td")
td_elements
```

```
[<td>1
 </td>,
 <td><i><a href="/wiki/Dangal_(2016_film)" title="Dangal (2016 film)">Dangal</a></i>
 </td>,
 <td align="right">₹1,968.03–₹2,200 crore
 </td>,
 <td style="text-align:center;">Hindi
 </td>,
 <td style="text-align:center;">2016
 </td>,
 <td style="text-align:center;"><sup class="reference" id="cite_ref-dgl_15-0"><a href="#cite_note-dgl-15"><s
pan class="cite-bracket">[</span>a<span class="cite-bracket">]</span></a></sup>
 </td>.
```

STEP 2 :

This code iterates through a list of `td_elements` (likely HTML table cells) and extracts the text content from each. It then splits each text by newline characters (`\n`), selects the first part (`text_1[0]`), and appends it to `list_general`. The final result is a list (`list_general`) containing the first line of text from each `td` element.

```
import re
list_general=[]
for td in td_elements:
    text = td.get_text()
    text_1=re.split("\n",text)
    a=text_1[0]
    list_general.append(a)

list_general
```

```
['1',
 'Dangal',
 '₹1,968.03–₹2,200 crore',
 'Hindi',
 '2016',
 '[a]',
 '2',
 'Baahubali 2: The Conclusion[α]',
 '₹1,810.60 crore',
 'Telugu',
 '2017',
```

STEP 3 :
This code creates a new list, `new_list`, by selecting elements from the `list_general` list. It iterates from index `0` to `46` (total 47 elements) and appends each element from `list_general` into `new_list`. After the loop, `new_list` contains the first 47 elements of `list_general`.

```
search_first_row_text='Dangal'
search_last_row_text='Sanju'
```

```
new_list=[]
for i in range(0,47):
    new_list.append(list_general[i])
print(new_list)
```

```
['1', 'Dangal', '₹1,968.03–₹2,200 crore', 'Hindi', '2016', '[a]', '2', 'Baahubali 2: The Conclusion[α]', '₹1,810.60 crore', 'Telugu', '2017', '[b]', '3', 'RRR', '₹1,300 crore', 'Telugu', '2022', '[c]', '4', 'KGF: Chapter 2', '₹1,200–1,250 crore', 'Kannada', '2022', '[d]', '5', 'Jawan', '₹1,148.32–1,159 crore', 'Hindi', '2023', '[e]', '6', 'Kalki 2898 AD', '₹1,100–1,200 crore', 'Telugu', '2024', '[f]', '7', 'Pathaan', '₹1,050.30–1,052.50 crore', 'Hindi', '2023', '[g]', '8', 'Animal', '₹917.82 crore', 'Hindi', '2023']
```

STEP 4 :
This code extracts every 6th element from `new_list`, starting from the first element, and appends each selected item to the `Rank` list.

Explanation:
1. `Rank=[]`: Initializes an empty list called `Rank`.
2. `for i in range(0, len(new_list), 6)`: Iterates through `new_list` with a step of 6, meaning `i` will take values `0, 6, 12, ...` until it reaches the length of `new_list`.
3. `Rank.append(new_list[i])`: Appends every 6th element of `new_list` (i.e., elements at indices 0, 6, 12, etc.) to `Rank`.
4. `print(Rank)`: Displays the final list of selected elements.

```
Rank=[]
for i in range(0,len(new_list),6):
   Rank.append(new_list[i])
print(Rank)
```

```
['1', '2', '3', '4', '5', '6', '7', '8']
```

```
Movie_Name =[]
for i in range(1,len(new_list),6):
   Movie_Name.append(new_list[i])
print(Movie_Name)
```

```
['Dangal', 'Baahubali 2: The Conclusion[α]', 'RRR', 'KGF: Chapter 2', 'Jawan', 'Kalki 2898 AD', 'Pathaan', 'Animal']
```

```
Worldwide_gross=[]
for i in range(2,len(new_list),6):
   Worldwide_gross.append(new_list[i])
print(Worldwide_gross)
```

```
['₹1,968.03–₹2,200 crore', '₹1,810.60 crore', '₹1,300 crore', '₹1,200–1,250 crore', '₹1,148.32–1,159 crore', '₹1,100–1,200 crore', '₹1,050.30–1,052.50 crore', '₹917.82 crore']
```

```
Language=[]
for i in range(3,len(new_list),6):
   Language.append(new_list[i])
print(Language)
```

```
['Hindi', 'Telugu', 'Telugu', 'Kannada', 'Hindi', 'Telugu', 'Hindi', 'Hindi']
```

```
Year=[]
for i in range(4,len(new_list),6):
   Year.append(new_list[i])
print(Year)
```

```
['2016', '2017', '2022', '2022', '2023', '2024', '2023', '2023']
```

STEP 5 :

This code creates a Pandas DataFrame named `df` with columns: "Rank," "Movie_Name," "Worldwide_gross," "Language," and "Year." Each column is populated with values from corresponding variables (`Rank`, `Movie_Name`, etc.). The `df` object will display the structured data in tabular form.

```
import pandas as pd

df=pd.DataFrame({"Rank":Rank,"Movie_Name":Movie_Name,"Worldwide_gross":Worldwide_gross, "Language":Language,"Ye
df
```

	Rank	Movie_Name	Worldwide_gross	Language	Year
0	1	Dangal	₹1,968.03–₹2,200 crore	Hindi	2016
1	2	Baahubali 2: The Conclusion[α]	₹1,810.60 crore	Telugu	2017
2	3	RRR	₹1,300 crore	Telugu	2022
3	4	KGF: Chapter 2	₹1,200–1,250 crore	Kannada	2022
4	5	Jawan	₹1,148.32–1,159 crore	Hindi	2023
5	6	Kalki 2898 AD	₹1,100–1,200 crore	Telugu	2024
6	7	Pathaan	₹1,050.30–1,052.50 crore	Hindi	2023
7	8	Animal	₹917.82 crore	Hindi	2023

Conclusion

The code effectively constructs a Pandas DataFrame to organize and represent data related to movies. By creating a structured table with specified columns, it allows for easier data manipulation, analysis, and visualization. This approach is essential in data science and analysis workflows, enabling users to work efficiently with datasets.

THANK YOU