Wikipedia Web Scraping Documentation: Largest Companies in India

By Adnan Rasool

Overview

This Python script performs web scraping on Wikipedia to extract comprehensive information about the largest companies in India. The script is designed to systematically extract data from two distinct tables on the Wikipedia page, process the information, and generate organized CSV files for further analysis. The implementation focuses on capturing both current and historical data about India's largest companies as ranked by Forbes.

Dependencies

The script relies on three primary Python libraries, each serving a crucial role in the web scraping pipeline:

- **Beautiful Soup 4 (bs4)**
  - A powerful library for pulling data from HTML and XML files
  - Provides intuitive methods to navigate, search, and modify parsing trees
  - Handles complex HTML structures with ease
  - Supports multiple parsers for different use cases
- **requests**
  - Handles HTTP/HTTPS requests with a simple, human-friendly API
  - Manages all HTTP operations needed to fetch web content
  - Automatically decodes returned content from the server
  - Supports various authentication methods and request customization
- **pandas**
  - Provides the DataFrame structure for efficient data manipulation
  - Offers comprehensive tools for data analysis and transformation
  - Enables seamless export of structured data to various formats
  - Handles complex data structures with built-in optimization

Implementation Details

URL Configuration

```
url = 'https://en.wikipedia.org/wiki/List_of_largest_companies_in_India'
```

The URL configuration points to the Wikipedia article containing comprehensive information about India's largest companies. This page is regularly updated and maintains historical data, making it an ideal source for both current and previous year rankings.

Data Extraction Process

1. Page Retrieval

```
page = requests.get(url)
soup = BeautifulSoup(page.text, 'html')
```

The page retrieval process involves two crucial steps:

1. The `requests.get()` method fetches the raw HTML content from the specified URL
2. BeautifulSoup transforms this raw HTML into a parsed object that can be navigated programmatically
    - The 'html' parser is specifically chosen for its robust handling of real-world HTML
    - The parsed object creates a nested data structure that mirrors the HTML document's hierarchy

2. First Table Processing (2024 Forbes List)

2.1 Table Location

```
table = soup.find('table', class_='wikitable sortable')
```

The script specifically targets the first table by:

- Using the `find()` method to locate the first matching table element
- Identifying the table through its CSS class combination of 'wikitable' and 'sortable'
- This approach ensures we capture the most recent Forbes rankings table

2.2 Header Extraction

```
world_titles = table.find_all('th')
world_table_titles = [title.text.strip() for title in world_titles]
```

The header extraction process:

1. Locates all `<th>` (table header) elements within the target table
2. Creates a list comprehension to process each header:
    - Extracts the text content from each header cell
    - Removes leading and trailing whitespace using `strip()`
    - Preserves the exact column names as they appear on Wikipedia

2.3 Data Frame Creation

```
df = pd.DataFrame(columns=world_table_titles)
```

The DataFrame initialization:

- Creates an empty pandas DataFrame structure
- Uses the extracted headers as column names
- Establishes the foundation for systematic data population

2.4 Row Data Population

```
column_data = table.find_all('tr')
for row in column_data[1:]:
    row_data = row.find_all('td')
    individual_row_data = [data.text.strip() for data in row_data]
    length = len(df)
    df.loc[length] = individual_row_data
```

The row population process follows these steps:

1. Extracts all table rows (`<tr>` elements)
2. Skips the header row using slice notation `[1:]`

3. For each row:
    - Finds all table data cells (`<td>` elements)
    - Extracts and cleans the text from each cell
    - Determines the current DataFrame length
    - Appends the processed row at the next available index

2.5 CSV Export

```
df.to_csv(r'Largest Companies in India 2024 Forbes.csv', index=False)
```

The export process:

- Converts the DataFrame to CSV format
- Uses a descriptive filename including the year
- Excludes the DataFrame index for cleaner output
- Creates a standalone file that can be used in other applications

3. Second Table Processing (2023 Forbes List)

3.1 Table Location

```
table2 = soup.find_all('table')[1]
```

The second table processing:

- Uses `find_all('table')[1]` to get the second table in the document
- This table contains the previous year's Forbes rankings
- Follows an identical processing pattern to the first table

3.2 Data Processing and Export

The script repeats the same systematic process for the second table:

1. Extracts headers using the same methodology
2. Creates a new DataFrame (df2)
3. Populates the DataFrame with row data
4. Exports to a separate CSV file for the 2023 rankings

Output Files

The script generates two comprehensive CSV files:

1. `Largest Companies in India 2024 Forbes.csv`
    - Contains the current year's Forbes rankings
    - Includes detailed company information
    - Maintains the original column structure from Wikipedia
2. `Largest Companies in India 2023 Forbes.csv`
    - Contains the previous year's Forbes rankings
    - Enables year-over-year comparison
    - Preserves historical data in a structured format

Each CSV file serves as a standalone dataset that can be used for:

- Data analysis and visualization
- Company performance tracking
- Year-over-year comparisons
- Integration with other data systems
- Business intelligence applications