TUTORIAL Basic Web Scraping Tutorial with Beautiful Soup

Scraping Books Data from books.toscrape.com

Step 1: Install Required Packages

```
!pip install requests beautifulsoup4 pandas
```

```
Requirement already satisfied: requests in
/usr/local/lib/python3.11/dist-packages (2.32.3)
Requirement already satisfied: beautifulsoup4 in
/usr/local/lib/python3.11/dist-packages (4.12.3)
Requirement already satisfied: pandas in
/usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.11/dist-packages (from requests) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests) (2024.12.14)
Requirement already satisfied: soupsieve>1.2 in
/usr/local/lib/python3.11/dist-packages (from beautifulsoup4) (2.6)
Requirement already satisfied: numpy>=1.23.2 in
/usr/local/lib/python3.11/dist-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.11/dist-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.11/dist-packages (from pandas) (2025.1)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2-
>pandas) (1.17.0)
```

Step 2: Import Libraries

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

print("Libraries imported successfully!")
```

```
Libraries imported successfully!
```

Step 3: Get the Webpage

```
# URL of the books website
url = "http://books.toscrape.com/"

# Get the webpage
response = requests.get(url)

# Check if request was successful
if response.status_code == 200:
    print("Successfully retrieved the webpage!")
    # Create BeautifulSoup object
    soup = BeautifulSoup(response.content, 'html.parser')
else:
    print(f"Failed to retrieve the webpage. Status code:
{response.status_code}")
```

```
Successfully retrieved the webpage!
```

Step 4: Extract Book Information

```
# Find all book articles
books = soup.find_all('article', class_='product_pod')
print(f"Found {len(books)} books on the page")

# Create a list to store book data
books_data = []

# Extract information from each book
for book in books:
    # Get title (in the image's alt text)
    title = book.h3.a['title']

    # Get price (in a <p> tag with class 'price_color')
    price = book.find('p', class_='price_color').text.strip()

    # Get availability (in a <p> tag with class 'availability')
    availability = book.find('p', class_='availability').text.strip()

    # Get rating (in the class attribute of <p> tag with class 'star-
rating')
    rating = book.find('p', class_='star-rating')['class'][1]

    # Store the data
    book_info = {
        'Title': title,
        'Price': price,
        'Availability': availability,
        'Rating': rating
    }
```

```
    books_data.append(book_info)

# Create DataFrame
df = pd.DataFrame(books_data)

# Display first few rows
print("\nFirst 5 books:")
display(df.head())
```

```
Found 20 books on the page

First 5 books:
```


```
{"summary":"{\n  \"name\": \"display(df\",\n  \"rows\": 5,\n
\"fields\": [\n    {\n      \"column\": \"Title\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n          \"Tipping
the Velvet\",\n          \"Sapiens: A Brief History of Humankind\",\n
\"Soumission\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Price\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n          \"\\
u00a353.74\",\n          \"\\u00a354.23\",\n          \"\\
u00a350.10\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Availability\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          \"In stock\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Rating\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          \"One\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


Step 5: Clean the Data

```
# Clean price (remove '£' symbol and convert to float)
df['Price'] = df['Price'].str.replace('£', '').astype(float)

# Clean availability (extract number of books)
df['Availability'] = df['Availability'].str.extract('(\d+)')

# Display cleaned data
print("Cleaned data:")
display(df.head())
```

```
Cleaned data:
```


```
{"repr_error":"Out of range float values are not JSON compliant:
nan","type":"dataframe"}
```

Step 6: Save to CSV File

```
# Save to CSV
df.to_csv('books_data.csv', index=False)
print("\nData saved to 'books_data.csv'")

# Verify the saved data
print("\nVerifying saved data:")
saved_df = pd.read_csv('books_data.csv')
display(saved_df.head())
```

```
Data saved to 'books_data.csv'

Verifying saved data:
```


```
{"summary":"{\n  \"name\": \"display(saved_df\",\n  \"rows\": 5,\n
\"fields\": [\n    {\n      \"column\": \"Title\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n          \"Tipping
the Velvet\",\n          \"Sapiens: A Brief History of Humankind\",\n
\"Soumission\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Price\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 2.647672562837028,\n        \"min\": 47.82,\n        \"max\":
54.23,\n        \"num_unique_values\": 5,\n        \"samples\": [\n
53.74,\n          54.23,\n          50.1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Availability\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
null,\n        \"min\": null,\n        \"max\": null,\n
\"num_unique_values\": 0,\n        \"samples\": [],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Rating\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\": 4,\n
\"samples\": [],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


Beginner-Friendly Web Scraping Problems [10 points each]

Problem 1: Scrape Book Titles and Prices

Objective: Extract a list of book titles and their corresponding prices from Books to Scrape.

Steps:

1. Navigate to the homepage of the website.

2. Identify all book titles and prices listed on the page.
3. Save the data into a CSV file with two columns: `Title` and `Price`.

Problem 2: Scrape Top 10 Quotes from Quotes to Scrape

Objective: Extract the top 10 quotes, their authors, and the associated tags from Quotes to Scrape.

Steps:

1. Go to the homepage of the website.
2. Extract the text of the first 10 quotes, their authors, and the tags associated with each quote.
3. Save the data in a CSV file with three columns: `Quote`, `Author`, and `Tags`.

Problem 3: Scrape Weather Data from World Weather Online

Objective: Extract the current weather conditions (temperature, weather condition, and humidity) for a given city.

Steps:

1. Visit https://www.timeanddate.com/weather/.
2. Search for the weather data for a city (e.g., New York).
3. Extract the current temperature, weather description, and humidity levels.
4. Save the data in a structured format (e.g., a JSON or CSV file).

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

url = 'https://books.toscrape.com/'

response = requests.get(url)

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

books = soup.find_all('article', class_='product_pod')

titles = []
prices = []

for book in books:
    title = book.find('h3').find('a')['title']
    titles.append(title)
```

```
    price = book.find('p', class_='price_color').text
    cleaned_price = price.replace('Â', '').strip()
    prices.append(cleaned_price[1:])

data = {'Title': titles, 'Price': prices}
df = pd.DataFrame(data)

df.to_csv('books_prices.csv', index=False, encoding='utf-8')

print("Data saved to books_prices.csv")
```

```
Data saved to books_prices.csv
```

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

url = 'https://quotes.toscrape.com/'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')
quotes = soup.find_all('div', class_='quote')

quote_texts = []
authors = []
tags = []

for i, quote in enumerate(quotes[:10]):
    quote_text = quote.find('span', class_='text').text
    quote_texts.append(quote_text)

    author = quote.find('small', class_='author').text
    authors.append(author)

    tag_list = quote.find_all('a', class_='tag')
    tag_names = [tag.text for tag in tag_list]
    tags.append(', '.join(tag_names))

data = {'Quote': quote_texts, 'Author': authors, 'Tags': tags}
df = pd.DataFrame(data)

df.to_csv('top_10_quotes.csv', index=False, encoding='utf-8')

print("Data saved to top_10_quotes.csv")
```

```
Data saved to top_10_quotes.csv
```

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

url = 'https://www.timeanddate.com/weather/india/delhi'
```

```
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')
temperature = soup.find('div', class_='h2').text.strip()
weather_condition = soup.find('div',
class_='h2').find_next('p').text.strip()

humidity_section = soup.find('div', class_='h2').find_next('table',
class_='zebra tb-wt fw va-m')
if humidity_section:
    rows = humidity_section.find_all('tr')
    for row in rows:
        if 'Humidity' in row.text:
            humidity = row.find_all('td')[1].text.strip()
            break
else:
    humidity = 'Not available'

print(f"Temperature: {temperature}")
print(f"Weather Condition: {weather_condition}")
print(f"Humidity: {humidity}")

weather_data = {
    'Temperature': [temperature],
    'Weather Condition': [weather_condition],
    'Humidity': [humidity]
}

df = pd.DataFrame(weather_data)

df.to_csv('weather_data.csv', index=False, encoding='utf-8')

df.to_json('weather_data.json', orient='records', lines=True)

print("Data saved to weather_data.csv and weather_data.json")

Temperature: 20 °C
Weather Condition: Sunny.
Humidity: Not available
Data saved to weather_data.csv and weather_data.json
```

Pandas Assignment [10 points each]

1. Create a DataFrame df from this dictionary data which has the index labels and Display a summary of the basic information about this DataFrame and its data.

```
import pandas as pd

df = pd.DataFrame(weather_data)

df_info = df.info()
df_summary = df.describe()
```

```
print("DataFrame Info:")
print(df_info)
print("\nSummary of the DataFrame:")
print(df_summary)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1 entries, 0 to 0
Data columns (total 3 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Temperature        1 non-null      object
 1   Weather Condition  1 non-null      object
 2   Humidity           1 non-null      object
dtypes: object(3)
memory usage: 156.0+ bytes
DataFrame Info:
None

Summary of the DataFrame:
       Temperature Weather Condition       Humidity
count            1                 1              1
unique           1                 1              1
top          20 °C            Sunny.  Not available
freq             1                 1              1
```

1. Return the first 5 rows of the DataFrame df.

```
df.head(n=5)
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 1,\n  \"fields\": [\n
{\n      \"column\": \"Temperature\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 1,\n
\"samples\": [\n          \"20\\u00a0\\u00b0C\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Weather Condition\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 1,\n        \"samples\": [\n
\"Sunny.\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Humidity\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          \"Not available\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```


1. Explain Pandas DataFrame Using Python List

```
import pandas as pd
```

```
data = [
    [1, 'Alice', 24],
    [2, 'Bob', 27],
    [3, 'Charlie', 22]
]

# Create DataFrame from list
df = pd.DataFrame(data, columns=['ID', 'Name', 'Age'])
print(df)
```

```
   ID     Name  Age
0   1    Alice   24
1   2      Bob   27
2   3  Charlie   22
```

1. How we can rename an index using the rename() method.

```
# Original DataFrame
df = pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [24, 27, 22]
})

# Renaming index
df = df.rename(index={0: 'A', 1: 'B', 2: 'C'})
print(df)
```

```
      Name  Age
A    Alice   24
B      Bob   27
C  Charlie   22
```

1. You have a 2D NumPy array that you have converted into a pandas DataFrame. You want to assign specific index values to the rows of this DataFrame. If you pass a list of index values to the DataFrame, how does it affect the DataFrame, and how would you apply these index values?

```
import numpy as np

# Create a 2D NumPy array
data = np.array([[1, 'Alice', 24],
                 [2, 'Bob', 27],
                 [3, 'Charlie', 22]])

# Create DataFrame and assign index
df = pd.DataFrame(data, columns=['ID', 'Name', 'Age'], index=['a',
'b', 'c'])
print(df)
```

```
  ID     Name Age
a  1    Alice  24
```

```
b  2      Bob  27
c  3  Charlie  22
```

1. You have a dictionary of data that you want to store as a pandas Series. After creating the Series and storing it in the df variable, you print it and observe that the data is represented in a one-dimensional linear format. Explain how to create this Series from the dictionary and describe the output you would expect when printing the Series.

```
# Creating a dictionary
data = {'a': 10, 'b': 20, 'c': 30}

# Create Series from dictionary
s = pd.Series(data)
print(s)
```

```
a    10
b    20
c    30
dtype: int64
```

1. You create a dictionary and store it as a DataFrame in the df variable. After printing, the data appears as 2-dimensional rows and columns. How would you create this DataFrame from the dictionary, and what does the output look like?

```
# Creating a dictionary
data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [24, 27, 22]}

# Create DataFrame from dictionary
df = pd.DataFrame(data)
print(df)
```

```
      Name  Age
0    Alice   24
1      Bob   27
2  Charlie   22
```