DSL 252 LAB 5(PLOTLY AND SEABORN)

12341820
ROHIT RAGHUWANSHI

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).

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

# Problem 1: Scrape Book Titles and Prices from "Books to Scrape" website
url = 'https://books.toscrape.com/'  # URL of the website to scrape
response = requests.get(url)  # Sending a GET request to fetch the page content
soup = BeautifulSoup(response.text, 'html.parser')  # Parsing the HTML content using BeautifulSoup

# Find all the book entries on the page
books = soup.find_all('article', class_='product_pod')

titles = []  # List to store book titles
prices = []  # List to store book prices

# Loop through each book and extract title and price
for book in books:
    # Extract the book title from the 'h3' tag
    title = book.find('h3').find('a')['title']
    titles.append(title)

    # Extract the book price and clean up the special character 'Â'
    price = book.find('p', class_='price_color').text
    cleaned_price = price.replace('Â', '').strip()  # Cleaning special character
    prices.append(cleaned_price[1:])  # Removing the currency symbol and adding to the list

# Create a DataFrame to store book data
data = {'Title': titles, 'Price': prices}
df = pd.DataFrame(data)

# Save the scraped data to a CSV file
df.to_csv('books_prices.csv', index=False, encoding='utf-8')
print("Data saved to books_prices.csv")  # Inform the user that the data is saved

# Problem 2: Scrape Top 10 Quotes from "Quotes to Scrape" website
url = 'https://quotes.toscrape.com/'  # URL of the Quotes website
response = requests.get(url)  # Sending a GET request to fetch the page content
soup = BeautifulSoup(response.text, 'html.parser')  # Parsing the HTML content

# Find all quotes on the page
quotes = soup.find_all('div', class_='quote')

quote_texts = []  # List to store quote text
authors = []  # List to store author names
tags = []  # List to store tags associated with quotes

# Loop through the first 10 quotes and extract the necessary information
for i, quote in enumerate(quotes[:10]):
    # Extract the text of the quote
    quote_text = quote.find('span', class_='text').text
    quote_texts.append(quote_text)

    # Extract the author's name
    author = quote.find('small', class_='author').text
    authors.append(author)
```

```
        # Extract tags associated with the quote (if any)
        tag_list = quote.find_all('a', class_='tag')
        tag_names = [tag.text for tag in tag_list]  # Extracting tag names
        tags.append(', '.join(tag_names))  # Join multiple tags with a comma

    # Create a DataFrame to store quote data
    data = {'Quote': quote_texts, 'Author': authors, 'Tags': tags}
    df = pd.DataFrame(data)

    # Save the scraped quote data to a CSV file
    df.to_csv('top_10_quotes.csv', index=False, encoding='utf-8')
    print("Data saved to top_10_quotes.csv")  # Inform the user that the data is saved

    # Problem 3: Scrape Weather Data from "Time and Date" website for Delhi
    url = 'https://www.timeanddate.com/weather/india/delhi'  # URL of the weather page
    response = requests.get(url)  # Sending a GET request to fetch the page content
    soup = BeautifulSoup(response.text, 'html.parser')  # Parsing the HTML content

    # Extract the temperature from the page
    temperature = soup.find('div', class_='h2').text.strip()

    # Extract the weather condition (e.g., clear, cloudy)
    weather_condition = soup.find('div', class_='h2').find_next('p').text.strip()

    # Initialize humidity as 'Not available' by default in case it isn't found
    humidity = 'Not available'

    # Look for the table containing humidity information
    humidity_section = soup.find('div', class_='h2').find_next('table', class_='zebra tb-wt fw va-m')
    if humidity_section:
        # Find all rows in the humidity table
        rows = humidity_section.find_all('tr')
        for row in rows:
            # Search for the row containing the word 'Humidity'
            if 'Humidity' in row.text:
                humidity = row.find_all('td')[1].text.strip()  # Extract humidity value from the second column
                break  # Exit the loop once the humidity is found

    # Print the scraped weather data
    print(f"Temperature: {temperature}")
    print(f"Weather Condition: {weather_condition}")
    print(f"Humidity: {humidity}")

    # Create a DataFrame to store weather data
    weather_data = {
        'Temperature': [temperature],
        'Weather Condition': [weather_condition],
        'Humidity': [humidity]
    }
    df = pd.DataFrame(weather_data)

    # Save the weather data to both CSV and JSON files
    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")  # Inform the user that the data is saved
```

```
[32] Data saved to books_prices.csv
     Data saved to top_10_quotes.csv
     Temperature: 73 °F
     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.

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.

```
[33] # Example dictionary data (replace with your actual data)
     data = {'col1': [1, 2, 3], 'col2': [4, 5, 6]}
     index_labels = ['A', 'B', 'C']
     # Create DataFrame with index labels
     df = pd.DataFrame(data, index=index_labels)
     # Display a summary of the DataFrame's information
     df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3 entries, A to C
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   col1    3 non-null      int64
 1   col2    3 non-null      int64
dtypes: int64(2)
memory usage: 72.0+ bytes
```

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

```
[38] pd.set_option("display.max_rows", None)
     df.head()
```

	Name	Age	City
0	Alice	25	New York
1	Bob	30	Los Angeles
2	Charlie	28	Chicago

3. Explain Pandas DataFrame Using Python List

```
import pandas as pd
# Example Python list of dictionaries
data = [
    {'Name': 'Alice', 'Age': 25, 'City': 'New York'},
    {'Name': 'Bob', 'Age': 30, 'City': 'Los Angeles'},
    {'Name': 'Charlie', 'Age': 28, 'City': 'Chicago'}
]
# Create a Pandas DataFrame from the list
df = pd.DataFrame(data)

# Explain Pandas DataFrame using the Python list
print("Explanation of Pandas DataFrame using a Python list:")
print("1. Python List of Dictionaries:")
print("   - The input data is a list where each element is a dictionary.")
print("   - Each dictionary represents a row in the DataFrame, with keys as column names and values as row data.")
print("   - For example:")
print(data)
print()
print("2. Creating DataFrame from List:")
print("   - Pandas `DataFrame()` constructor converts the list of dictionaries into a structured tabular format.")
print("   - Keys of the dictionaries become column names, and values become the corresponding row entries in those columns.")
print()
print("3. DataFrame Structure:")
print("   - The resulting DataFrame is a two-dimensional labeled data structure with columns of potentially different types.")
print("   - It has an index for row access and column names for column access.")
print()
print("4. DataFrame vs. List:")
print("   - Lists are versatile but lack structure and labels for columns and rows.  Accessing elements relies solely on index positions.")
print("   - DataFrames provide labeled structure, making it efficient to select and manipulate data based on column names and row indexes.")
print("   - They offer built-in functions for data analysis, manipulation, and visualization. Lists don't have such functionalities.")
print()
print("Example DataFrame:")
display(df)
```

[35]

```
Explanation of Pandas DataFrame using a Python list:
1. Python List of Dictionaries:
   - The input data is a list where each element is a dictionary.
   - Each dictionary represents a row in the DataFrame, with keys as column names and values as row data.
   - For example:
[{'Name': 'Alice', 'Age': 25, 'City': 'New York'}, {'Name': 'Bob', 'Age': 30, 'City': 'Los Angeles'}, {'Name': 'Charlie', 'Age': 28, 'City': 'Chicago'}]

2. Creating DataFrame from List:
   - Pandas `DataFrame()` constructor converts the list of dictionaries into a structured tabular format.
   - Keys of the dictionaries become column names, and values become the corresponding row entries in those columns.

3. DataFrame Structure:
   - The resulting DataFrame is a two-dimensional labeled data structure with columns of potentially different types.
   - It has an index for row access and column names for column access.

4. DataFrame vs. List:
   - Lists are versatile but lack structure and labels for columns and rows.  Accessing elements relies solely on index positions.
   - DataFrames provide labeled structure, making it efficient to select and manipulate data based on column names and row indexes.
   - They offer built-in functions for data analysis, manipulation, and visualization. Lists don't have such functionalities.

Example DataFrame:
```

	Name	Age	City
0	Alice	25	New York
1	Bob	30	Los Angeles
2	Charlie	28	Chicago

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

```python
# Example: Rename the index 'Title' to 'Book Title'
df = df.rename(index={'Title': 'Book Title'}) # Rename a specific index label
print(df.head())

# Rename multiple index labels
df = df.rename(index={0: 'First Book', 1: 'Second Book'}) # Rename multiple specific index labels
print(df.head())

# Rename all index labels using a function
df = df.rename(index=lambda x: x.upper()) # Rename all index labels using a lambda function
print(df.head())
```

```
        0  1  2
row_a   1  2  3
row_b   4  5  6
row_c   7  8  9
        0  1  2
row_a   1  2  3
row_b   4  5  6
row_c   7  8  9
        0  1  2
ROW_A   1  2  3
ROW_B   4  5  6
ROW_C   7  8  9
```

5. 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?

```python
import pandas as pd
import numpy as np
#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
# Sample 2D NumPy array
data = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# Create a DataFrame
df = pd.DataFrame(data)

# New index values
new_index = ['row_a', 'row_b', 'row_c']

# Assign new index to the DataFrame. This modifies the DataFrame in place.
df.index = new_index
# Display the DataFrame to see the changed index
df
```

	0	1	2
row_a	1	2	3
row_b	4	5	6
row_c	7	8	9

6. 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.

```
[2] import pandas as pd
    # Sample dictionary data
    data = {'a': 10, 'b': 20, 'c': 30, 'd': 40, 'e': 50}

    # Create a pandas Series from the dictionary
    df = pd.Series(data)

    # Print the Series
    df
```

```
    0
a  10
b  20
c  30
d  40
e  50

dtype: int64
```

7. 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?

```
[1]
    import pandas as pd
    # Sample dictionary (replace with your actual dictionary)
    data = {
        'col1': [1, 2, 3],
        'col2': [4, 5, 6],
        'col3': [7, 8, 9]
    }
    # Create the DataFrame
    df = pd.DataFrame(data)

    # Print the DataFrame
    df
```

	col1	col2	col3
0	1	4	7
1	2	5	8
2	3	6	9

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:

```
Navigate to the homepage of the website.
Identify all book titles and prices listed on the page.
Save the data into a CSV file with two columns: Title and Price.
```

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

def scrape_book_data():
    url = "http://books.toscrape.com/"
    try:
        response = requests.get(url)
        response.raise_for_status()  # Raise an exception for bad status codes

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

        book_titles = []
        book_prices = []

        # Find all book containers on the page (adjust the CSS selector if necessary)
        books = soup.find_all("article", class_="product_pod")

        for book in books:
            title = book.h3.a["title"]
            price = book.find("p", class_="price_color").text.strip()
            book_titles.append(title)
            book_prices.append(price)

        # Create a Pandas DataFrame
        data = {"Title": book_titles, "Price": book_prices}
        df = pd.DataFrame(data)
        return df

    except requests.exceptions.RequestException as e:
        print(f"Error fetching data: {e}")
        return None
    except AttributeError as e:
        print(f"Error parsing data: {e}")
        return None


df = scrape_book_data()

if df is not None:
    # 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')
    print(saved_df.head())
```

```
Data saved to 'books_data.csv'

Verifying saved data:
                                Title   Price
0                A Light in the Attic  £51.77
1                  Tipping the Velvet  £53.74
2                        Soumission  £50.10
3                     Sharp Objects  £47.82
4  Sapiens: A Brief History of Humankind  £54.23
```

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:

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

```python
def scrape_quotes():
    url = "http://quotes.toscrape.com/"
    try:
        response = requests.get(url)
        response.raise_for_status()
        soup = BeautifulSoup(response.content, "html.parser")

        quotes = []
        authors = []
        tags_list = []

        quote_divs = soup.find_all("div", class_="quote")
        for quote_div in quote_divs[:10]:  # Limit to the first 10 quotes
            quote = quote_div.find("span", class_="text").text
            author = quote_div.find("small", class_="author").text
            tags = [tag.text for tag in quote_div.find_all("a", class_="tag")]

            quotes.append(quote)
            authors.append(author)
            tags_list.append(", ".join(tags)) # Join tags with comma and space

        data = {"Quote": quotes, "Author": authors, "Tags": tags_list}
        df = pd.DataFrame(data)
        return df

    except requests.exceptions.RequestException as e:
        print(f"Error fetching data: {e}")
        return None
    except AttributeError as e:
        print(f"Error parsing data: {e}")
        return None

df_quotes = scrape_quotes()

if df_quotes is not None:
    df_quotes.to_csv('quotes_data.csv', index=False)
    print("\nData saved to 'quotes_data.csv'")
    print(df_quotes.head(10))
```

```
Data saved to 'quotes_data.csv'
                                               Quote             Author  \
0  “The world as we have created it is a process ...    Albert Einstein
1  “It is our choices, Harry, that show what we t...       J.K. Rowling
2  “There are only two ways to live your life. On...    Albert Einstein
3  “The person, be it gentleman or lady, who has ...        Jane Austen
4  “Imperfection is beauty, madness is genius and...     Marilyn Monroe
5  “Try not to become a man of success. Rather be...    Albert Einstein
6  “It is better to be hated for what you are tha...         André Gide
7  “I have not failed. I’ve just found 10,000 way...   Thomas A. Edison
8  “A woman is like a tea bag; you never know how...  Eleanor Roosevelt
9  “A day without sunshine is like, you know, nig...       Steve Martin

                                               Tags
0       change, deep-thoughts, thinking, world
1                    abilities, choices
2  inspirational, life, live, miracle, miracles
3            aliteracy, books, classic, humor
4                  be-yourself, inspirational
5                    adulthood, success, value
6                                   life, love
7  edison, failure, inspirational, paraphrased
8                misattributed-eleanor-roosevelt
9                    humor, obvious, simile
```