ajain74_PRG550A.231.Ass2

December 7, 2023

[45]:
```
# program: ajain74_PRG550A.231.Ass2.ipynb
# name: Aayush Jain
# student number: 125609214
# date: December 7, 2023
# course: PRG550A
# purpose: solution to PRG550C Assignment #2 (Fall 2023) using Jupyter on the␣
 ↪Raspberry Pi
```

[10]:
```
import re
from datetime import datetime
import calendar
import matplotlib.pyplot as plt
import pandas as pd
import os
```

[11]:
```
def createAllCsvFiles(startYear, endYear, pathOfTheFolder):
    """
    Creates CSV files for weather data from HTML files.

    Args:
        startYear (int): The starting year.
        endYear (int): The ending year.
        pathOfTheFolder (str): The path of the folder containing HTML files.

    Returns:
        pd.DataFrame: DataFrame containing all weather data.
    """
    all_data = []

    for year in range(startYear, endYear + 1):
        pathOfTheHtmlFile = os.path.join(pathOfTheFolder, f'torontoWeather.
 ↪{year}.html')
        if not os.path.exists(pathOfTheHtmlFile):
            print(f"File not found: {pathOfTheHtmlFile}")
            continue

        with open(pathOfTheHtmlFile, 'r') as content:
            contentOfHtml = content.read()
```

```
        design = re.compile(r"<a href='/cities/toronto/day/(\w+-\d+)'.*?>(-?
↪\d+\.\d+)</td>\s*"
                            r"<td class='text-right .*?'>(-?\d+\.\d+)</td>\s*"
                            r"<td class='text-right .*?'>(-?\d+\.\d+)</td>",␣
↪re.S | re.M)

        filenameCsv = os.path.join(pathOfTheFolder, f"torontoWeather.{year}.
↪csv")
        dataOfTheYear = []

        for match in re.finditer(design, contentOfHtml):
            dateStr, tempHigh, tempLow, precipitation = match.groups()

            try:
                dateObj = datetime.strptime(f"{year}-{dateStr.title()}",␣
↪'%Y-%B-%d')
            except ValueError as e:
                if dateStr == 'February-29':
                    print(f"Skipping invalid date 'February-29' for non-leap␣
↪year {year}")
                    continue
                else:
                    print(f"Error parsing date '{dateStr}' in year {year}: {e}")
                    continue

            dataOfTheYear.append(["Toronto", dateObj.timetuple().tm_yday,␣
↪dateObj.strftime('%B'),
                                  dateObj.day, year, float(tempHigh),␣
↪float(tempLow), float(precipitation)])

        all_data.extend(dataOfTheYear)
        pd.DataFrame(dataOfTheYear, columns=["City", "dayOfyear", "month",␣
↪"dayOfMonth",
                                             "Year", "highTemp", "lowTemp",␣
↪"precipitation"]).to_csv(filenameCsv, index=False)

    return pd.DataFrame(all_data, columns=["City", "dayOfyear", "month",␣
↪"dayOfMonth",
                                           "Year", "highTemp", "lowTemp",␣
↪"precipitation"]).sort_values(by=["Year", "dayOfyear"])
```

[12]:
```
def graphWeatherByDayForEachYear(pdFrame, dayNum):
    """
    Plots weather statistics for a specific day across multiple years.
```

```
    Args:
        pdFrame (pd.DataFrame): DataFrame containing weather data.
        dayNum (int): Day number to plot.

    Returns:
        None
    """
    filteredData = pdFrame[pdFrame['dayOfyear'] == dayNum]

    if filteredData.empty:
        print(f"No weather data found for day number {dayNum}.")
        return

    dataGrouped = filteredData.groupby('Year').agg({'highTemp': 'mean',␣
↪'lowTemp': 'mean', 'precipitation': 'sum'}).reset_index()
    plt.figure(figsize=(12, 8))
    plt.plot(dataGrouped['Year'], dataGrouped['highTemp'], label='Mean High␣
↪Temp', color='red')
    plt.plot(dataGrouped['Year'], dataGrouped['lowTemp'], label='Mean Low␣
↪Temp', color='blue')
    plt.bar(dataGrouped['Year'], dataGrouped['precipitation'], label='Total␣
↪Precipitation', color='grey', alpha=0.5)
    plt.title(f'Weather Stats for Day Number {dayNum} Across Each Year')
    plt.xlabel('Year')
    plt.ylabel('Temperature (°C) / Precipitation (mm)')
    plt.legend()
    plt.grid(True)
    plt.show()
```

[13]:
```
def showWeatherByMonthForAllYears(pdFrame, month):
    """
    Displays average weather statistics for a specific month across all years.

    Args:
        pdFrame (pd.DataFrame): DataFrame containing weather data.
        month (int): Month number.

    Returns:
        None
    """
    nameOfTheMonth = calendar.month_name[month]
    filteredData = pdFrame[pdFrame['month'] == nameOfTheMonth]

    if filteredData.empty:
        print(f"No weather data found for {nameOfTheMonth}.")
        return
```

```
    meanTempHigh, meanTempLow, meanPrecipitation = filteredData['highTemp'].
↪mean(), filteredData['lowTemp'].mean(), filteredData['precipitation'].mean()
    print(f"\nAverage Weather stats for {nameOfTheMonth} across all years: Mean␣
↪High Temperature: {meanTempHigh:.2f}°C | Mean Low Temperature: {meanTempLow:.
↪2f}°C | Mean Precipitation: {meanPrecipitation:.2f} mm")
```

[14]:
```
def showWeatherByDayForAllYears(pdFrame, dayNum):
    """
    Displays weather statistics for a specific day across all years.

    Args:
        pdFrame (pd.DataFrame): DataFrame containing weather data.
        dayNum (int): Day number.

    Returns:
        None
    """
    filteredData = pdFrame.query(f'dayOfyear == {dayNum}')

    if filteredData.empty:
        print(f"No weather data found for day number {dayNum}.")
        return

    print(f"\nWeather stats for day number {dayNum} across all years: Mean High␣
↪Temperature: {filteredData['highTemp'].mean():.2f}°C | Mean Low Temperature:␣
↪{filteredData['lowTemp'].mean():.2f}°C | Mean Total Precipitation:␣
↪{filteredData['precipitation'].mean():.2f} mm")
```

[15]:
```
def graphWeatherByMonthForEachYear(pdFrame, month):
    """
    Plots mean high and low temperatures for a specific month across multiple␣
↪years.

    Args:
        pdFrame (pd.DataFrame): DataFrame containing weather data.
        month (int): Month number.

    Returns:
        None
    """
    nameOfTheMonth = calendar.month_name[month]
    filteredData = pdFrame[pdFrame['month'] == nameOfTheMonth]
    dataGrouped = filteredData.groupby('Year').agg({'highTemp': 'mean',␣
↪'lowTemp': 'mean'}).reset_index()

    if dataGrouped.empty:
        print(f"No weather data found for {nameOfTheMonth}.")
```

```
        return

    plt.figure(figsize=(10, 6))
    plt.plot(dataGrouped['Year'], dataGrouped['highTemp'], label='Mean High␣
↪Temp')
    plt.plot(dataGrouped['Year'], dataGrouped['lowTemp'], label='Mean Low Temp')
    plt.title(f'Mean High and Low Temperatures for {nameOfTheMonth} (Each␣
↪Year)')
    plt.xlabel('Year')
    plt.ylabel('Temperature (°C)')
    plt.legend()
    plt.grid(True)
    plt.show()
```

[16]:
```
def showWeatherByMonthAndYear(pdFrame, year, month):
    """
    Displays weather statistics for a specific month and year.

    Args:
        pdFrame (pd.DataFrame): DataFrame containing weather data.
        year (int): Year.
        month (int): Month number.

    Returns:
        None
    """
    nameOfTheMonth = calendar.month_name[month]
    filteredData = pdFrame[(pdFrame['month'] == nameOfTheMonth) &␣
↪(pdFrame['Year'] == year)]

    if filteredData.empty:
        print(f"No weather data found for {nameOfTheMonth} {year}.")
        return

    tempHigh, tempLow, totalPrecipitation, tempMean = filteredData['highTemp'].
↪max(), filteredData['lowTemp'].min(), filteredData['precipitation'].sum(),␣
↪(filteredData['highTemp'] + filteredData['lowTemp']).mean() / 2
    print(f"\nWeather stats for {nameOfTheMonth} {year}: Highest Temperature:␣
↪{tempHigh}°C | Lowest Temperature: {tempLow}°C | Total Precipitation:␣
↪{totalPrecipitation:.2f} mm | Mean Temperature: {tempMean:.2f}°C")
```

[17]:
```
def showWeatherByDayMonthYear(pdFrame, day, month, year):
    """
    Displays weather statistics for a specific day, month, and year.

    Args:
        pdFrame (pd.DataFrame): DataFrame containing weather data.
```

```
        day (int): Day number.
        month (int): Month number.
        year (int): Year.

    Returns:
        None
    """
    filteredData = pdFrame.query(f'dayOfMonth == {day} and month == "{calendar.
↪month_name[month]}" and Year == {year}')

    if filteredData.empty:
        print(f"No weather data found for {day}-{month}-{year}.")
        return
    tempHigh, tempLow, precipitation = filteredData.iloc[0, 5:8]

    print(f"Weather on {day}-{month}-{year}: High Temperature: {tempHigh}°C |␣
↪Low Temperature: {tempLow}°C | Total Precipitation: {precipitation} mm")
```

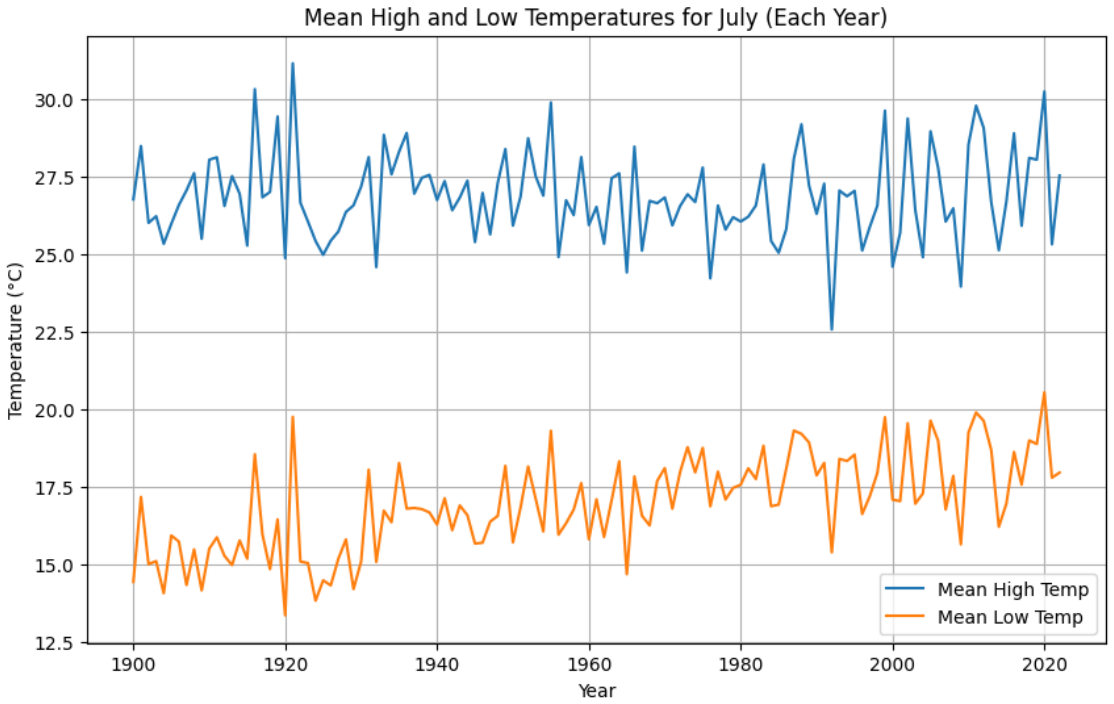
[18]:
```
df = createAllCsvFiles(1900, 2023, "")
showWeatherByDayMonthYear(df, 10, 8, 2022)
showWeatherByDayForAllYears(df, 20)
showWeatherByMonthAndYear(df, 2022, 8)
showWeatherByMonthForAllYears(df, 5)
graphWeatherByMonthForEachYear(df, 7)
graphWeatherByDayForEachYear(df, 20)
```

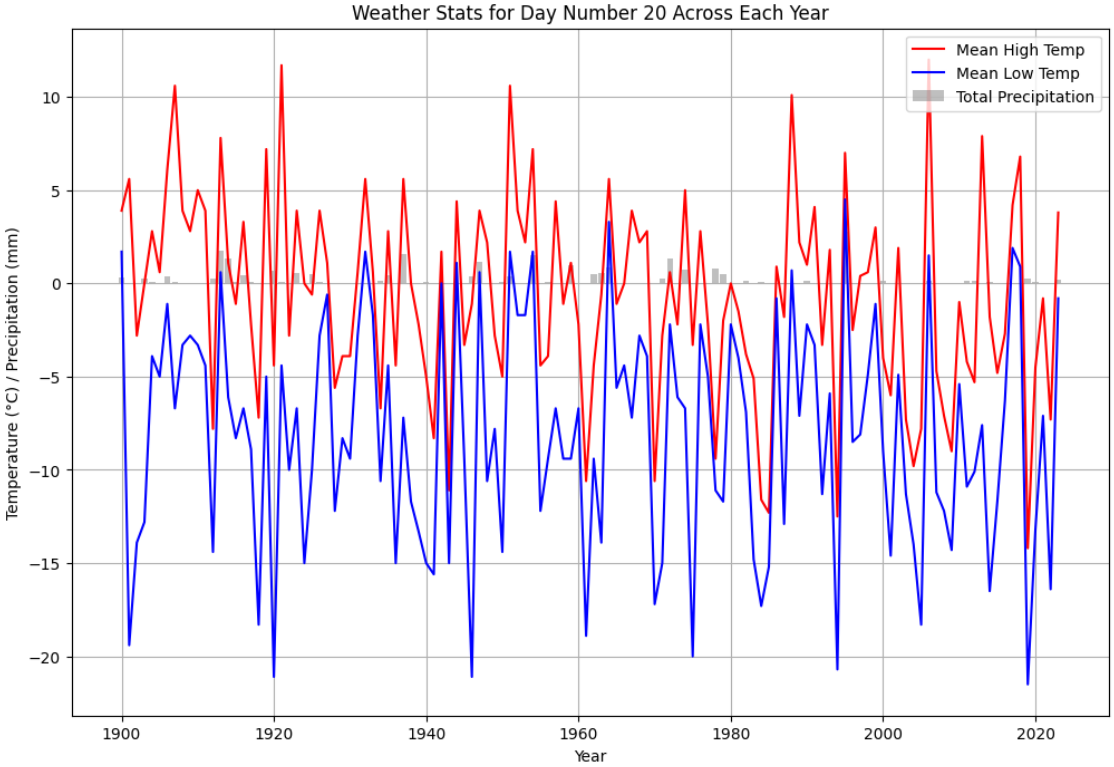
```
Weather on 10-8-2022: High Temperature: 26.6°C | Low Temperature: 17.3°C | Total
Precipitation: 0.0 mm

Weather stats for day number 20 across all years: Mean High Temperature: -0.62°C
| Mean Low Temperature: -8.09°C | Mean Total Precipitation: 0.19 mm

Weather stats for August 2022: Highest Temperature: 31.4°C | Lowest Temperature:
13.6°C | Total Precipitation: 6.45 mm | Mean Temperature: 22.87°C

Average Weather stats for May across all years: Mean High Temperature: 18.30°C |
Mean Low Temperature: 8.65°C | Mean Precipitation: 0.27 mm
```


Mean High and Low Temperatures for July (Each Year)
Temperature (°C)
Year
Mean High Temp
Mean Low Temp
30.0
27.5
25.0
22.5
20.0
17.5
15.0
12.5
1900
1920
1940
1960
1980
2000
2020


Weather Stats for Day Number 20 Across Each Year
Temperature (°C) / Precipitation (mm)
Year
Mean High Temp
Mean Low Temp
Total Precipitation
10
5
0
−5
−10
−15
−20
1900
1920
1940
1960
1980
2000
2020