# **Breakout Sessions Handout**

## **Breakout Session 1:**

|  |
| --- |
| *# Time Series Basics - Breakout Session #1*  # W5D5 - Weather Data Analysis  """  INSTRUCTIONS:  Working in groups, complete the following exercises to practice creating and  manipulating time series data. Fill in the code where indicated and answer  the discussion questions at the end.  GOALS:  1. Create a DataFrame with daily weather data for a month  2. Convert string dates to datetime and set as index  3. Perform various datetime selections and manipulations  4. Create basic time series visualizations  """  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from datetime import datetime, timedelta  # -----------------------------------------------------------------------------  # Exercise 1: Create a time series dataset from scratch  # -----------------------------------------------------------------------------  # Create a date range for March 2023 (all days in March)  # YOUR CODE HERE  march\_dates =  # Generate random temperature data with a realistic pattern  # Hint: March temperatures might follow a slight upward trend as spring begins  # YOUR CODE HERE  # Create temperatures with a base of 10°C, gradually increasing to about 15°C  # Add some daily fluctuation with random noise  # Generate random precipitation data (values between 0 and 15mm)  # YOUR CODE HERE  # Make some days have 0 precipitation, others have varying amounts  # Create your weather DataFrame with temperature and precipitation columns  # YOUR CODE HERE  weather\_df =  # Print the first 5 rows to verify  print("Exercise 1: Created weather dataset")  print(weather\_df.head())  print()  # -----------------------------------------------------------------------------  # Exercise 2: Working with string dates  # -----------------------------------------------------------------------------  # Below is weather data with dates as strings  weather\_data = {  'date': ['2023-03-01', '2023-03-02', '2023-03-03', '2023-03-04', '2023-03-05',  '2023-03-06', '2023-03-07', '2023-03-08', '2023-03-09', '2023-03-10'],  'temperature': [12.5, 13.2, 11.8, 14.5, 15.2, 12.8, 10.5, 11.7, 14.3, 16.5],  'humidity': [78, 82, 76, 70, 65, 73, 88, 83, 74, 68],  'wind\_speed': [5.2, 4.8, 7.3, 6.5, 3.2, 2.5, 8.1, 6.2, 4.5, 3.8]  }  # Create a DataFrame from this dictionary  string\_dates\_df = pd.DataFrame(weather\_data)  # 1. Convert the 'date' column to datetime format  # YOUR CODE HERE  # 2. Set the date column as the index  # YOUR CODE HERE  # 3. Print the result to verify  print("Exercise 2: Converted string dates to datetime index")  print(string\_dates\_df.head())  print()  # -----------------------------------------------------------------------------  # Exercise 3: Time-based selection and indexing  # -----------------------------------------------------------------------------  # Using the DataFrame you created in Exercise 2, perform the following selections:  # 1. Select data for March 5th  # YOUR CODE HERE  march\_5 =  print("Data for March 5th:")  print(march\_5)  print()  # 2. Select data from March 3rd to March 7th  # YOUR CODE HERE  march\_3\_to\_7 =  print("Data from March 3rd to March 7th:")  print(march\_3\_to\_7)  print()  # 3. Get the average temperature for the first week of March (1st to 7th)  # YOUR CODE HERE  first\_week\_avg =  print(f"Average temperature for the first week: {first\_week\_avg:.2f}°C")  print()  # 4. Find the day with the highest temperature  # YOUR CODE HERE  hottest\_day =  print(f"Hottest day: {hottest\_day}, Temperature: {string\_dates\_df.loc[hottest\_day, 'temperature']}°C")  print()  # -----------------------------------------------------------------------------  # Exercise 4: Basic time series visualization  # -----------------------------------------------------------------------------  # 1. Create a line plot showing temperature over time  plt.figure(figsize=(10, 6))  # YOUR CODE HERE  # Plot temperature with markers for each data point  # Add a title, labels, and grid  plt.tight\_layout()  # Uncomment to save the plot  # plt.savefig('temperature\_plot.png')  # 2. Create a plot that shows both temperature and humidity  plt.figure(figsize=(12, 6))  # YOUR CODE HERE  # Create a plot with two y-axes to show both metrics  # Be sure to include a legend  plt.tight\_layout()  # Uncomment to save the plot  # plt.savefig('temperature\_humidity\_plot.png')  # -----------------------------------------------------------------------------  # Exercise 5: Resampling and frequency conversion  # -----------------------------------------------------------------------------  # Load a larger dataset with hourly weather data for March 2023  # Note: In a real breakout session, this data might be provided in a CSV file  # For this exercise, we'll generate some simulated data  # Create an hourly datetime range for March 2023  hourly\_dates = pd.date\_range(start='2023-03-01', end='2023-03-31 23:00:00', freq='H')  # Create simulated hourly temperature data with daily cycles  hours = np.arange(len(hourly\_dates))  daily\_cycle = 3 \* np.sin(2 \* np.pi \* hours / 24) # Daily temperature cycle: +/- 3°C  trend = 10 + 0.2 \* (hourly\_dates.day - 1) # Base temp of 10°C, increasing through the month  hourly\_temps = trend + daily\_cycle + np.random.normal(0, 1, len(hourly\_dates)) # Add noise  # Create the hourly DataFrame  hourly\_df = pd.DataFrame({  'temperature': hourly\_temps  }, index=hourly\_dates)  # 1. Resample the hourly data to daily frequency (calculate the mean for each day)  # YOUR CODE HERE  daily\_df =  # 2. Resample to get the daily minimum and maximum temperatures  # YOUR CODE HERE  daily\_min\_max =  # 3. Calculate the daily temperature range (max - min)  # YOUR CODE HERE  daily\_range =  print("Daily temperature summary (first 5 days):")  print(daily\_min\_max.head())  print("\nDaily temperature range (first 5 days):")  print(daily\_range.head())  # 4. Create a plot showing daily min, mean, and max temperatures  plt.figure(figsize=(12, 6))  # YOUR CODE HERE  # Plot the daily minimum, mean, and maximum temperatures  # Use different colors and include a legend  plt.tight\_layout()  # Uncomment to save the plot  # plt.savefig('daily\_temp\_range\_plot.png')  # -----------------------------------------------------------------------------  # Discussion Questions  # -----------------------------------------------------------------------------  """  After completing the exercises, discuss the following questions with your group:  1. What challenges did you encounter when working with datetime objects?  2. How does having a datetime index change how you interact with the data?  3. What types of time-based selections seem most useful for weather analysis?  4. How might the frequency of observations (hourly, daily, monthly) impact analysis?  5. What patterns did you observe in the temperature data? How would you enhance  the visualizations to better show these patterns?  """  # When you finish, be prepared to share one insight or challenge with the class. |

## **Breakout Session 2:**

|  |
| --- |
| *# Time Series Basics - Breakout Session #1*  # W5D5 - Weather Data Analysis  """  INSTRUCTIONS:  Working in groups, complete the following exercises to practice creating and  manipulating time series data. Fill in the code where indicated and answer  the discussion questions at the end.  GOALS:  1. Create a DataFrame with daily weather data for a month  2. Convert string dates to datetime and set as index  3. Perform various datetime selections and manipulations  4. Create basic time series visualizations  """  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from datetime import datetime, timedelta  # -----------------------------------------------------------------------------  # Exercise 1: Create a time series dataset from scratch  # -----------------------------------------------------------------------------  # Create a date range for March 2023 (all days in March)  # YOUR CODE HERE  march\_dates =  # Generate random temperature data with a realistic pattern  # Hint: March temperatures might follow a slight upward trend as spring begins  # YOUR CODE HERE  # Create temperatures with a base of 10°C, gradually increasing to about 15°C  # Add some daily fluctuation with random noise  # Generate random precipitation data (values between 0 and 15mm)  # YOUR CODE HERE  # Make some days have 0 precipitation, others have varying amounts  # Create your weather DataFrame with temperature and precipitation columns  # YOUR CODE HERE  weather\_df =  # Print the first 5 rows to verify  print("Exercise 1: Created weather dataset")  print(weather\_df.head())  print()  # -----------------------------------------------------------------------------  # Exercise 2: Working with string dates  # -----------------------------------------------------------------------------  # Below is weather data with dates as strings  weather\_data = {  'date': ['2023-03-01', '2023-03-02', '2023-03-03', '2023-03-04', '2023-03-05',  '2023-03-06', '2023-03-07', '2023-03-08', '2023-03-09', '2023-03-10'],  'temperature': [12.5, 13.2, 11.8, 14.5, 15.2, 12.8, 10.5, 11.7, 14.3, 16.5],  'humidity': [78, 82, 76, 70, 65, 73, 88, 83, 74, 68],  'wind\_speed': [5.2, 4.8, 7.3, 6.5, 3.2, 2.5, 8.1, 6.2, 4.5, 3.8]  }  # Create a DataFrame from this dictionary  string\_dates\_df = pd.DataFrame(weather\_data)  # 1. Convert the 'date' column to datetime format  # YOUR CODE HERE  # 2. Set the date column as the index  # YOUR CODE HERE  # 3. Print the result to verify  print("Exercise 2: Converted string dates to datetime index")  print(string\_dates\_df.head())  print()  # -----------------------------------------------------------------------------  # Exercise 3: Time-based selection and indexing  # -----------------------------------------------------------------------------  # Using the DataFrame you created in Exercise 2, perform the following selections:  # 1. Select data for March 5th  # YOUR CODE HERE  march\_5 =  print("Data for March 5th:")  print(march\_5)  print()  # 2. Select data from March 3rd to March 7th  # YOUR CODE HERE  march\_3\_to\_7 =  print("Data from March 3rd to March 7th:")  print(march\_3\_to\_7)  print()  # 3. Get the average temperature for the first week of March (1st to 7th)  # YOUR CODE HERE  first\_week\_avg =  print(f"Average temperature for the first week: {first\_week\_avg:.2f}°C")  print()  # 4. Find the day with the highest temperature  # YOUR CODE HERE  hottest\_day =  print(f"Hottest day: {hottest\_day}, Temperature: {string\_dates\_df.loc[hottest\_day, 'temperature']}°C")  print()  # -----------------------------------------------------------------------------  # Exercise 4: Basic time series visualization  # -----------------------------------------------------------------------------  # 1. Create a line plot showing temperature over time  plt.figure(figsize=(10, 6))  # YOUR CODE HERE  # Plot temperature with markers for each data point  # Add a title, labels, and grid  plt.tight\_layout()  # Uncomment to save the plot  # plt.savefig('temperature\_plot.png')  # 2. Create a plot that shows both temperature and humidity  plt.figure(figsize=(12, 6))  # YOUR CODE HERE  # Create a plot with two y-axes to show both metrics  # Be sure to include a legend  plt.tight\_layout()  # Uncomment to save the plot  # plt.savefig('temperature\_humidity\_plot.png')  # -----------------------------------------------------------------------------  # Exercise 5: Resampling and frequency conversion  # -----------------------------------------------------------------------------  # Load a larger dataset with hourly weather data for March 2023  # Note: In a real breakout session, this data might be provided in a CSV file  # For this exercise, we'll generate some simulated data  # Create an hourly datetime range for March 2023  hourly\_dates = pd.date\_range(start='2023-03-01', end='2023-03-31 23:00:00', freq='H')  # Create simulated hourly temperature data with daily cycles  hours = np.arange(len(hourly\_dates))  daily\_cycle = 3 \* np.sin(2 \* np.pi \* hours / 24) # Daily temperature cycle: +/- 3°C  trend = 10 + 0.2 \* (hourly\_dates.day - 1) # Base temp of 10°C, increasing through the month  hourly\_temps = trend + daily\_cycle + np.random.normal(0, 1, len(hourly\_dates)) # Add noise  # Create the hourly DataFrame  hourly\_df = pd.DataFrame({  'temperature': hourly\_temps  }, index=hourly\_dates)  # 1. Resample the hourly data to daily frequency (calculate the mean for each day)  # YOUR CODE HERE  daily\_df =  # 2. Resample to get the daily minimum and maximum temperatures  # YOUR CODE HERE  daily\_min\_max =  # 3. Calculate the daily temperature range (max - min)  # YOUR CODE HERE  daily\_range =  print("Daily temperature summary (first 5 days):")  print(daily\_min\_max.head())  print("\nDaily temperature range (first 5 days):")  print(daily\_range.head())  # 4. Create a plot showing daily min, mean, and max temperatures  plt.figure(figsize=(12, 6))  # YOUR CODE HERE  # Plot the daily minimum, mean, and maximum temperatures  # Use different colors and include a legend  plt.tight\_layout()  # Uncomment to save the plot  # plt.savefig('daily\_temp\_range\_plot.png')  # -----------------------------------------------------------------------------  # Discussion Questions  # -----------------------------------------------------------------------------  """  After completing the exercises, discuss the following questions with your group:  1. What challenges did you encounter when working with datetime objects?  2. How does having a datetime index change how you interact with the data?  3. What types of time-based selections seem most useful for weather analysis?  4. How might the frequency of observations (hourly, daily, monthly) impact analysis?  5. What patterns did you observe in the temperature data? How would you enhance  the visualizations to better show these patterns?  """  # When you finish, be prepared to share one insight or challenge with the class. |