Day - 3

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Question - 3

In Python, what is the primary purpose of the iloc method when working with Pandas DataFrames?

A. Selecting columns by label

B. Selecting rows by label

C. Selecting rows and columns by integer position

D. Creating a new DataFrame

Day - 3

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 1

homefoodi

Celebrate Home Food Everyday

Scenario: You work for a popular food delivery app. The company wants to improve the accuracy of delivery time estimates provided to customers. Currently, delivery times are often inaccurate due to various factors such as traffic, restaurant preparation time, and delivery distance.

Day - 3

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 1

1) What data cleaning and preprocessing steps would you perform on the collected data to ensure its quality and reliability for analysis?

2) What exploratory data analysis techniques would you apply to understand the relationships between delivery times and factors like traffic, restaurant preparation time, and delivery distance?

3) If user feedback data is available, how would you analyze it to gain insights into the accuracy of delivery time estimates and areas for improvement?

homefoodi

Day - 3

10 Days Python Data Analytics Interview Class

Industries Helping Hands

homefoodi

Interview Question - 1

1) What data cleaning and preprocessing steps would you perform on the collected data to ensure its quality and reliability for analysis?

Handling Missing Values: Address any missing data points in the dataset by either imputing missing values or removing records with missing critical information.

Outlier Detection: Identify and handle outliers in delivery times or distances that may skew the analysis. Consider using statistical methods like the Z-score or IQR method.

Timestamp Standardization: Ensure that all timestamps are in a consistent format and timezone for accurate time-based analysis.

Data Validation: Check for data consistency and integrity, including cross-verifying restaurant preparation times with actual delivery times.

Day - 3

10 Days Python Data Analytics Interview Class

Industries Helping Hands

homefoodi

Interview Question - 1

2) What exploratory data analysis techniques would you apply to understand the relationships between delivery times and factors like traffic, restaurant preparation time, and delivery distance?

Descriptive Statistics: Calculate summary statistics for delivery times, distances, and other relevant variables to get an overview of the data distribution.

Correlation Analysis: Use correlation coefficients to measure the strength and direction of relationships between delivery times and factors like traffic, preparation time, and distance.

Data Visualization: Create visualizations such as scatter plots, histograms, or heatmaps to visualize patterns and trends in the data, especially how delivery times vary with different factors.

Time-Series Analysis: Analyze delivery time trends over time to identify any seasonality or temporal patterns.

Day - 3

10 Days Python Data Analytics Interview Class

Industries Helping Hands

homefoodi

Interview Question - 1

3) If user feedback data is available, how would you analyze it to gain insights into the accuracy of delivery time estimates and areas for improvement?

Sentiment Analysis: Perform sentiment analysis on user feedback comments to categorize feedback as positive, negative, or neutral. This helps gauge overall satisfaction.

Word Clouds: Create word clouds to visualize frequently mentioned keywords or phrases in user feedback, highlighting areas of concern or praise.

Feature Extraction: Extract valuable insights from user feedback by identifying common themes, such as complaints about late deliveries or positive comments about accurate time estimates.

Quantitative Metrics: Use quantitative metrics like Net Promoter Score (NPS) or Customer Satisfaction Score (CSAT) to quantify user satisfaction and track improvements over time.

Day - 3

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 2

meesho

How does Pandas handle time series data, and what are the advantages of using Pandas for time series analysis?

Day - 3

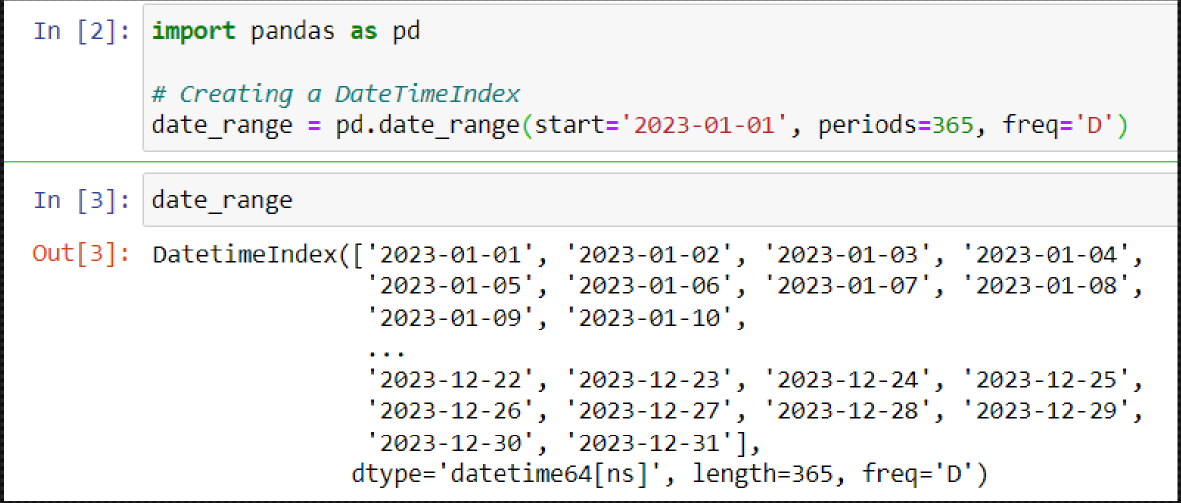
10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 2

Handling Time Series Data in Pandas:

DateTime Index: Pandas allows you to use the ‘DateTimeIndex’ to represent time-related data. This index type is designed for efficient time-based indexing and slicing.



```
In [2]: import pandas as pd

        # Creating a DateTimeIndex
        date_range = pd.date_range(start='2023-01-01', periods=365, freq='D')
```

```
In [3]: date_range

Out[3]: DatetimeIndex(['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-04',
                       '2023-01-05', '2023-01-06', '2023-01-07', '2023-01-08',
                       '2023-01-09', '2023-01-10',
                       ...
                       '2023-12-22', '2023-12-23', '2023-12-24', '2023-12-25',
                       '2023-12-26', '2023-12-27', '2023-12-28', '2023-12-29',
                       '2023-12-30', '2023-12-31'],
                      dtype='datetime64[ns]', length=365, freq='D')
```

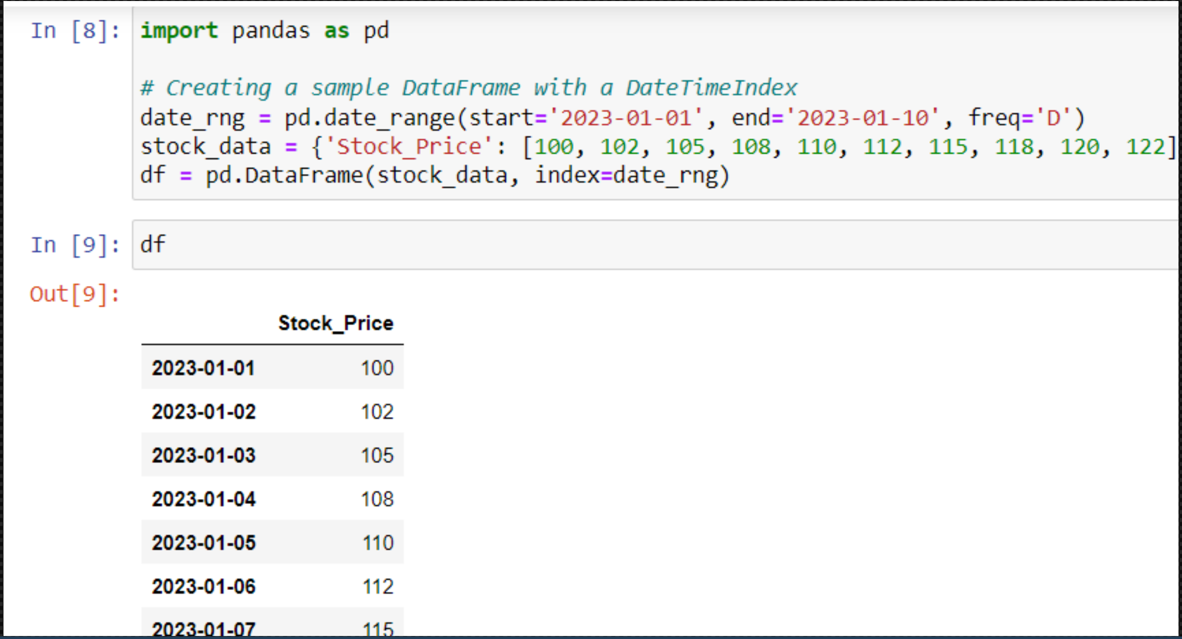
Day - 3

10 Days Python Data Analytics Interview Class

Interview Question - 2

Handling Time Series Data in Pandas:

Time-Based Slicing: You can easily slice and filter time series data using date and time criteria



```
In [8]: import pandas as pd

        # Creating a sample DataFrame with a DateTimeIndex
        date_rng = pd.date_range(start='2023-01-01', end='2023-01-10', freq='D')
        stock_data = {'Stock_Price': [100, 102, 105, 108, 110, 112, 115, 118, 120, 122]
        df = pd.DataFrame(stock_data, index=date_rng)

In [9]: df

Out[9]:
```

	Stock_Price
2023-01-01	100
2023-01-02	102
2023-01-03	105
2023-01-04	108
2023-01-05	110
2023-01-06	112
2023-01-07	115

Day - 3

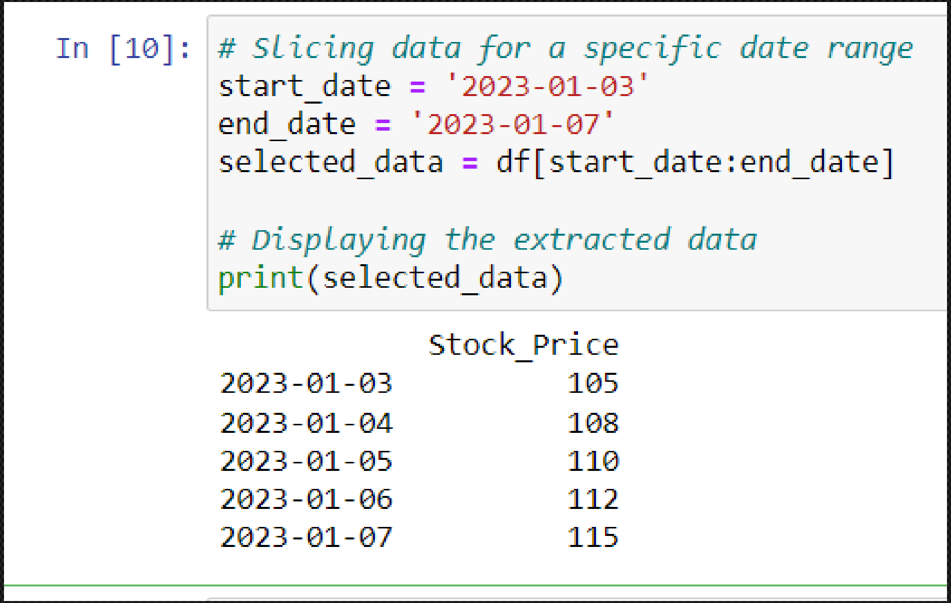
10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 2

Handling Time Series Data in Pandas:

Time-Based Slicing: You can easily slice and filter time series data using date and time criteria



```
In [10]: # Slicing data for a specific date range
         start_date = '2023-01-03'
         end_date = '2023-01-07'
         selected_data = df[start_date:end_date]

         # Displaying the extracted data
         print(selected_data)
```

```
            Stock_Price
2023-01-03          105
2023-01-04          108
2023-01-05          110
2023-01-06          112
2023-01-07          115
```

Day - 3

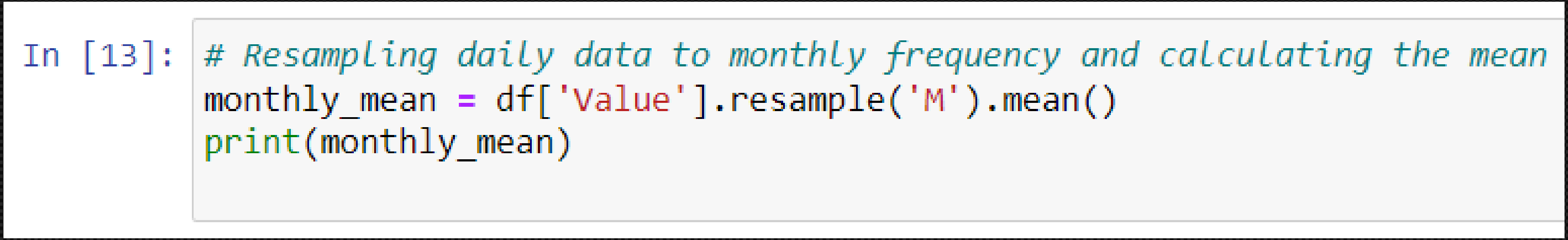
10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 2

Resampling and Aggregation:

Pandas allows you to resample time series data to different frequencies and apply aggregation functions. Here's how to resample daily data to monthly frequency and calculate the mean:



```
In [13]: # Resampling daily data to monthly frequency and calculating the mean
         monthly_mean = df['Value'].resample('M').mean()
         print(monthly_mean)
```

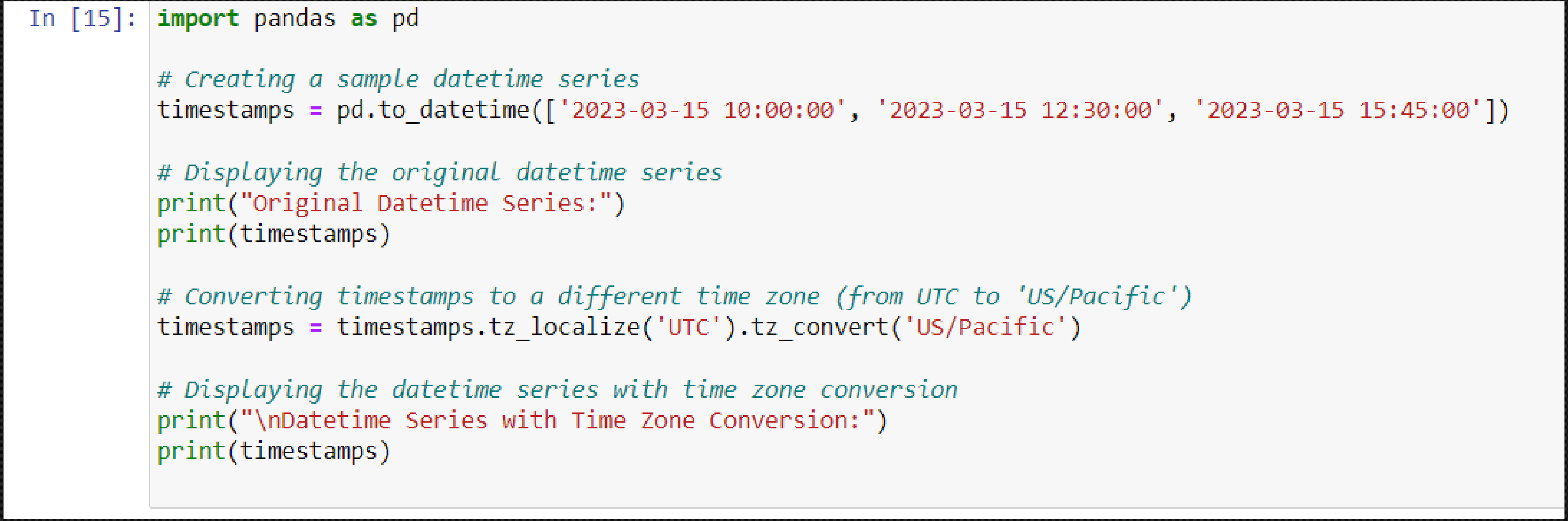
10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 2

Time Zone Handling:

Pandas supports time zone handling. Here's an example of converting timestamps to a different time zone:



```
In [15]: import pandas as pd

         # Creating a sample datetime series
         timestamps = pd.to_datetime(['2023-03-15 10:00:00', '2023-03-15 12:30:00', '2023-03-15 15:45:00'])

         # Displaying the original datetime series
         print("Original Datetime Series:")
         print(timestamps)

         # Converting timestamps to a different time zone (from UTC to 'US/Pacific')
         timestamps = timestamps.tz_localize('UTC').tz_convert('US/Pacific')

         # Displaying the datetime series with time zone conversion
         print("\nDatetime Series with Time Zone Conversion:")
         print(timestamps)
```

Day - 3

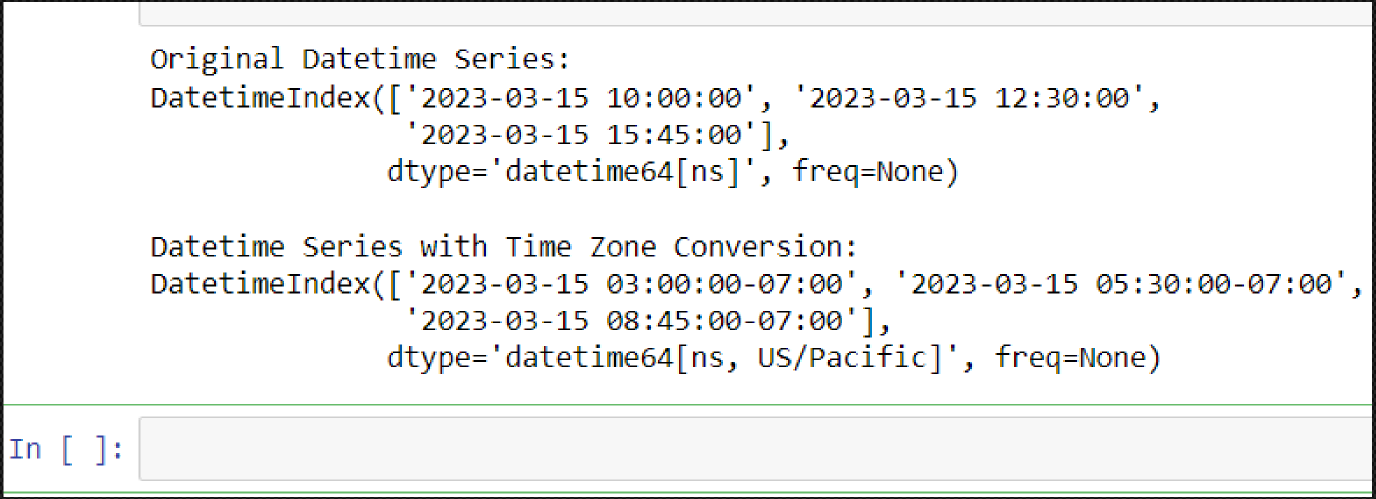
10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 2

Time Zone Handling:

Pandas supports time zone handling. Here's an example of converting timestamps to a different time zone:



```
Original Datetime Series:
DatetimeIndex(['2023-03-15 10:00:00', '2023-03-15 12:30:00',
               '2023-03-15 15:45:00'],
              dtype='datetime64[ns]', freq=None)

Datetime Series with Time Zone Conversion:
DatetimeIndex(['2023-03-15 03:00:00-07:00', '2023-03-15 05:30:00-07:00',
               '2023-03-15 08:45:00-07:00'],
              dtype='datetime64[ns, US/Pacific]', freq=None)
```

```
In [ ]:
```

Day - 3

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 2

Advantages of Using Pandas for Time Series Analysis:

Data Alignment : Pandas automatically aligns time series data based on timestamps, ensuring that data points are correctly matched, even when dealing with missing or irregular intervals.

Data Transformation: You can easily perform common time series operations like shifting, differencing, and rolling calculations using Pandas, simplifying data preparation for analysis.

Data Visualization: Pandas integrates seamlessly with data visualization libraries like Matplotlib and Seaborn, enabling the creation of informative time series plots and charts.

Integration with Other Libraries: You can seamlessly integrate Pandas with other data analysis and machine learning libraries like NumPy, Scikit-Learn, and Statsmodels, allowing for more advanced time series modeling and forecasting.

Day - 3

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 3

Explain the difference between Python lists and NumPy arrays, and when would you use one over the other in data analysis?

10 Days Python Data Analytics Interview Class

Interview Question - 3

Python lists and NumPy arrays are both used for storing and manipulating data, but they have several key differences:

1. Data Type Homogeneity:

Python Lists: Python lists can contain elements of different data types. For example, a single list can hold integers, floats, strings, and even other lists.

NumPy Arrays: NumPy arrays are homogeneous, meaning they store elements of the same data type. This homogeneity allows for efficient memory storage and optimized numerical operations.

Day - 3

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 3

Python lists and NumPy , key differences:

2. Performance:

Python Lists: Lists are not optimized for numerical operations and can be slower when performing operations on large datasets. They are implemented in Python's standard library and are relatively slower for mathematical calculations.

NumPy Arrays: NumPy arrays are highly efficient for numerical computations. They are implemented in C and provide low-level memory optimizations. This makes NumPy arrays significantly faster than Python lists for numerical operations, especially on large datasets.

Day - 3

10 Days Python Data Analytics Interview Class

Interview Question - 3

Python lists and NumPy , key differences:

3. Size:

Python Lists: Lists are dynamic, which means you can change their size by appending, inserting, or removing elements. They do not have a fixed size.

NumPy Arrays: NumPy arrays have a fixed size upon creation, and you cannot change their size without creating a new array. This fixed size is useful for memory optimization and efficient data storage.

Day - 3

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 3

Python lists and NumPy , key differences:

4. Functionality:

Python Lists: Python lists have limited built-in functions for numerical operations. While you can perform basic operations, such as addition and multiplication, they are not as optimized as NumPy functions.

NumPy Arrays: NumPy provides a wide range of mathematical and statistical functions that are optimized for arrays. It enables vectorized operations, broadcasting, and element-wise computations, making it a powerful tool for scientific and numerical computing.

10 Days Python Data Analytics Interview Class

Interview Question - 3

Python lists and NumPy , key differences:

5. Syntax and Convenience:

Python Lists: Python lists are part of the core Python language and are easy to create and manipulate. They are suitable for general-purpose programming tasks.

NumPy Arrays: NumPy arrays require importing the NumPy library, which adds an extra step. However, they provide extensive functionality and performance benefits for numerical tasks.

Day - 3

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 4

How would you count the occurrences of a specific substring within a larger string in Python?

Day - 3

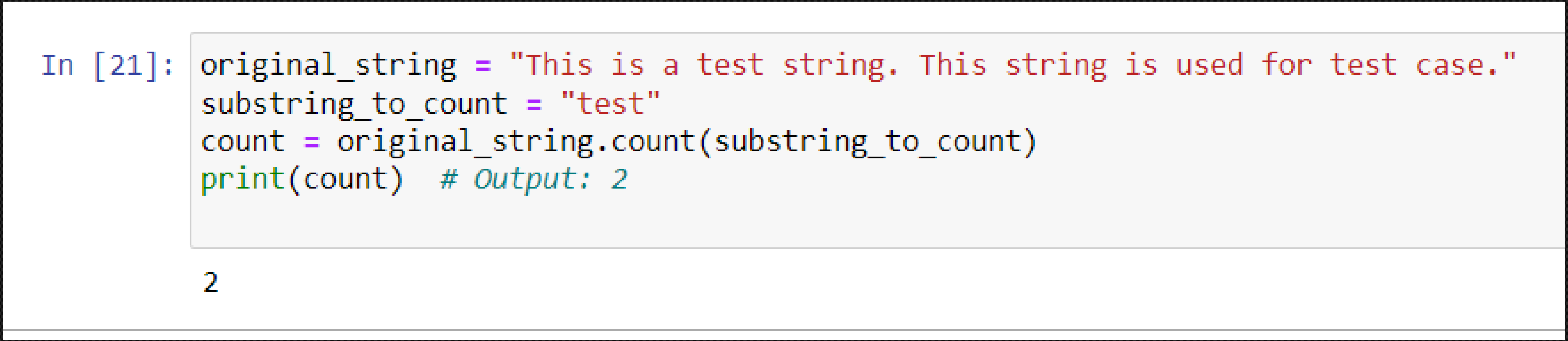
10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 4

1) Using str.count() method:

The str.count() method allows you to count the non-overlapping occurrences of a substring within a larger string. Here's an example:



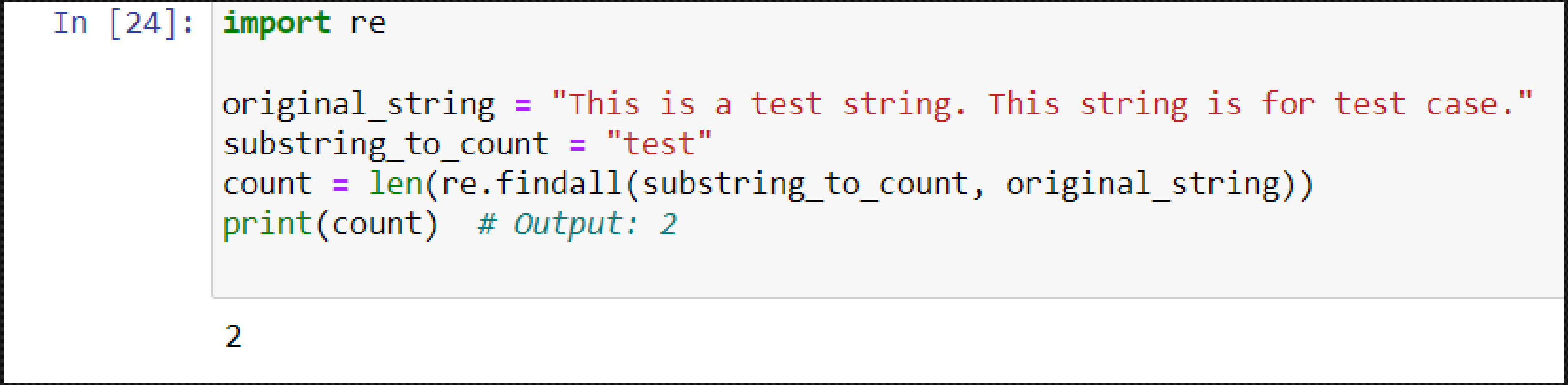
```
In [21]: original_string = "This is a test string. This string is used for test case."
         substring_to_count = "test"
         count = original_string.count(substring_to_count)
         print(count)  # Output: 2

         2
```

10 Days Python Data Analytics Interview Class

Interview Question - 4

2) Using regular expressions (regex):

You can use the re module in Python to count occurrences of a substring using regular expressions. Here's an example:



```
In [24]: import re

         original_string = "This is a test string. This string is for test case."
         substring_to_count = "test"
         count = len(re.findall(substring_to_count, original_string))
         print(count)  # Output: 2

         2
```

Day - 3

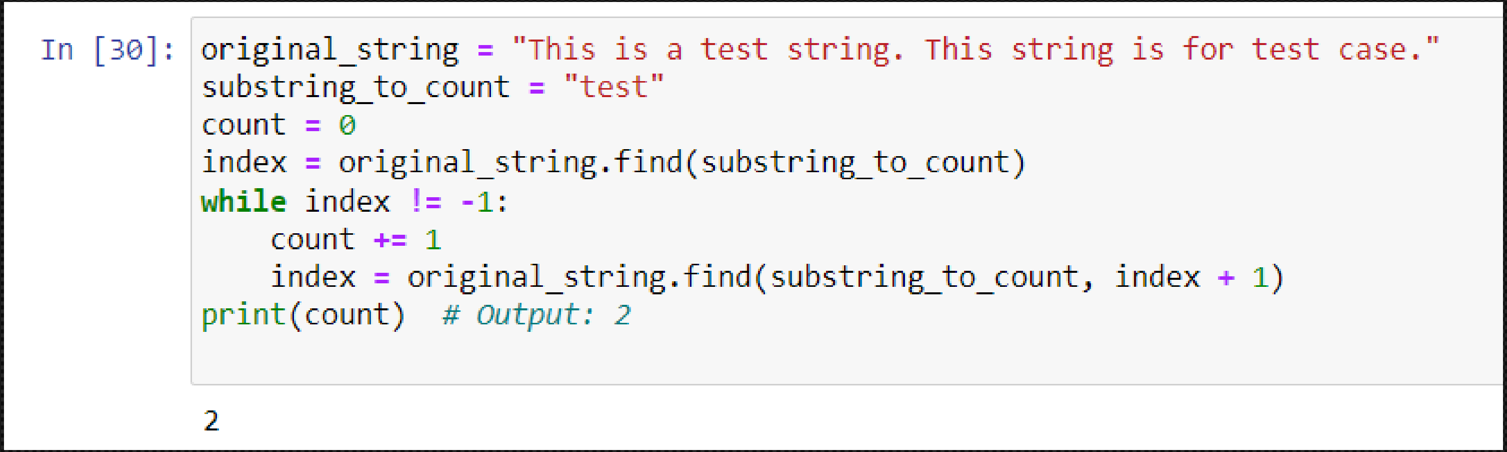
10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 4

3) Using a loop:

You can manually iterate through the larger string and count occurrences of the substring. Here's an example:



```
In [30]: original_string = "This is a test string. This string is for test case."
         substring_to_count = "test"
         count = 0
         index = original_string.find(substring_to_count)
         while index != -1:
             count += 1
             index = original_string.find(substring_to_count, index + 1)
         print(count)  # Output: 2

         2
```

India's Most Affordable Pay After Placement Data Analytics Course

+91-7880-113-112 Contact or Fill the Form in the Description

Day - 3

10 Days Python Data Analytics Interview Class

Interview Question - 5

How can you create a dictionary from two lists, one containing keys and the other containing values, efficiently in Python?

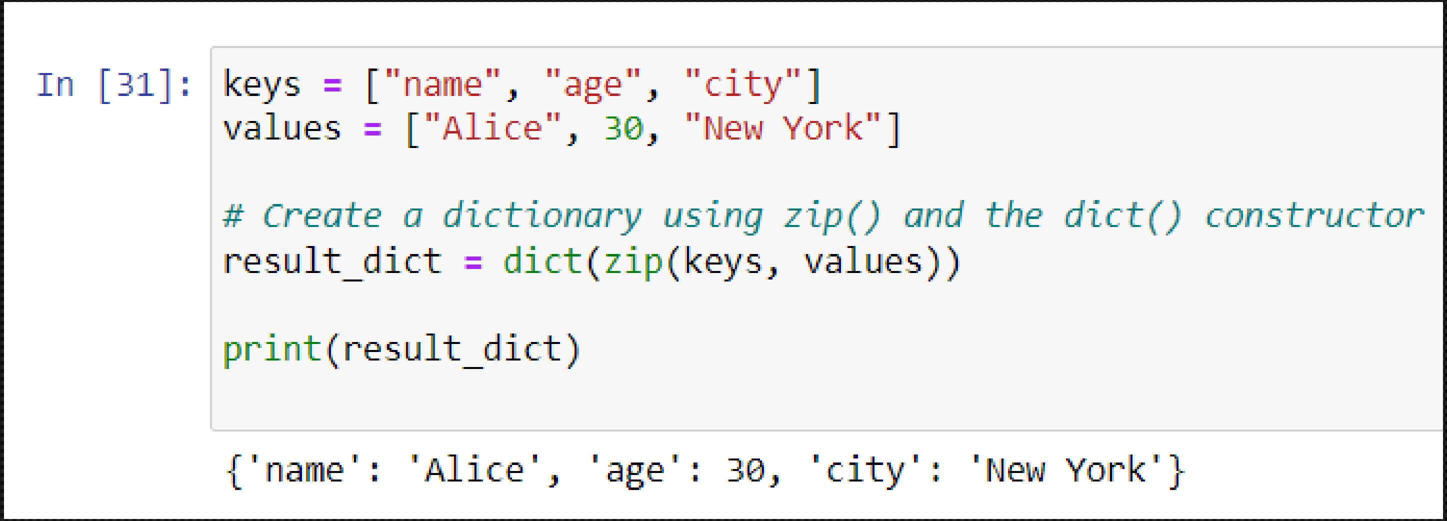
Day - 3

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 5

You can efficiently create a dictionary from two lists—one containing keys and the other containing values—using the dict() constructor and the zip() function in Python.



```
In [31]: keys = ["name", "age", "city"]
         values = ["Alice", 30, "New York"]

         # Create a dictionary using zip() and the dict() constructor
         result_dict = dict(zip(keys, values))

         print(result_dict)

         {'name': 'Alice', 'age': 30, 'city': 'New York'}
```

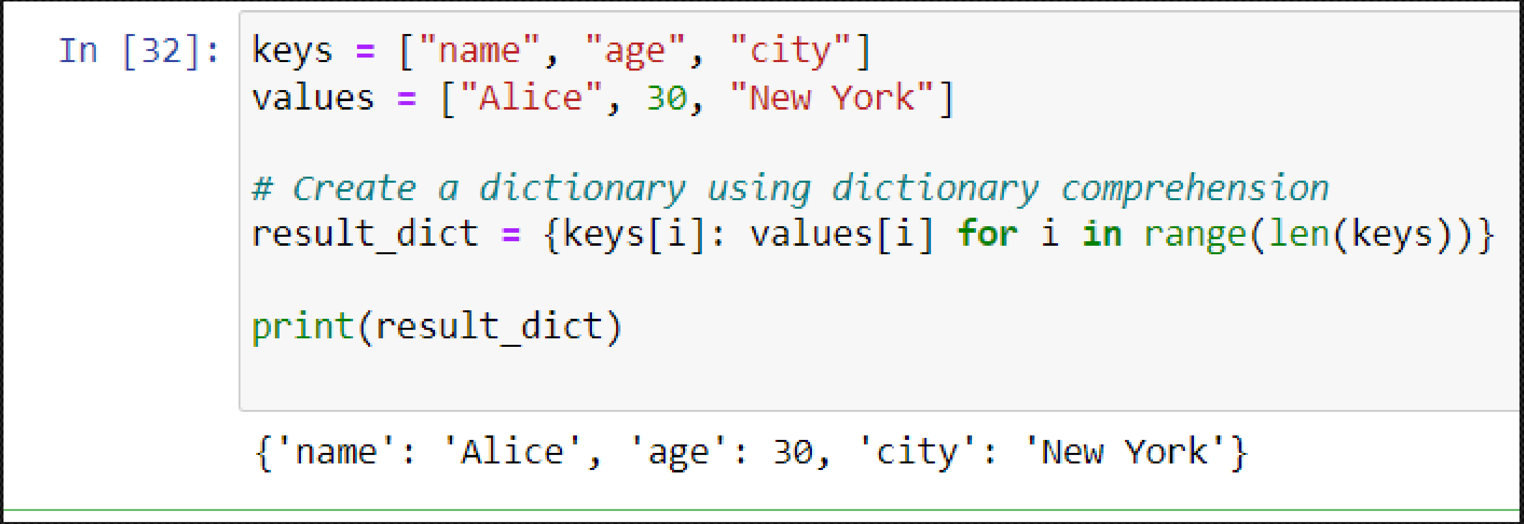
Day - 3

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 5

Using Dictionary Comprehension:



```
In [32]: keys = ["name", "age", "city"]
         values = ["Alice", 30, "New York"]

         # Create a dictionary using dictionary comprehension
         result_dict = {keys[i]: values[i] for i in range(len(keys))}

         print(result_dict)

         {'name': 'Alice', 'age': 30, 'city': 'New York'}
```

Day - 3

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 6

weRize

Explain the process of adding annotations, titles, and labels to Seaborn plots to enhance their readability and interpretation.

Day - 3

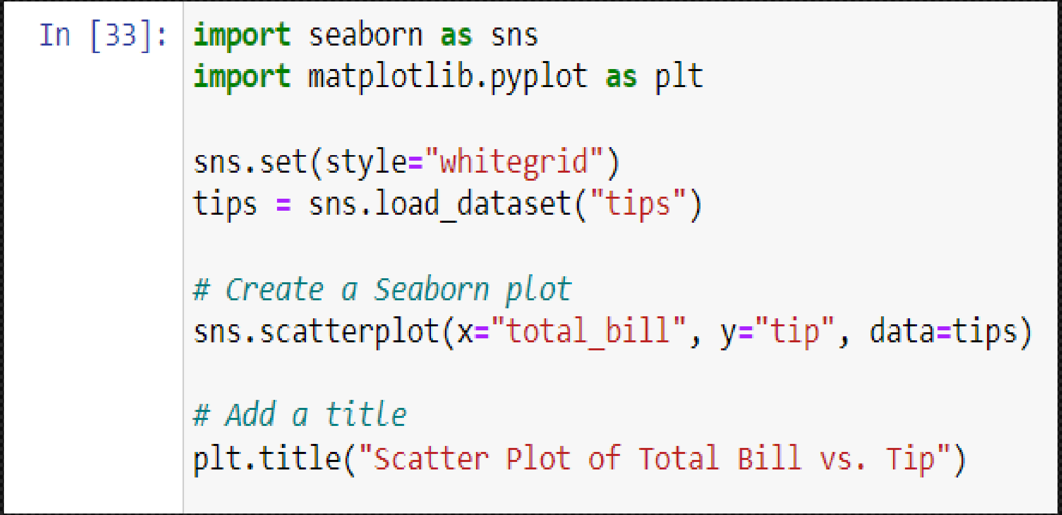
10 Days Python Data Analytics Interview Class

Industries Helping Hands

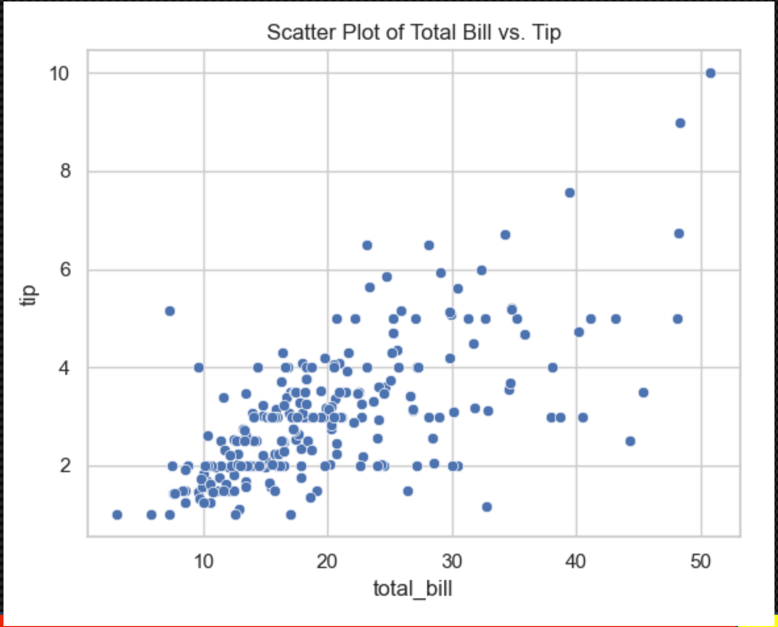
Interview Question - 6

Titles and Labels:

Title: You can add a title to your Seaborn plot using the plt.title() function from Matplotlib. For example:



```
In [33]: import seaborn as sns
         import matplotlib.pyplot as plt

         sns.set(style="whitegrid")
         tips = sns.load_dataset("tips")

         # Create a Seaborn plot
         sns.scatterplot(x="total_bill", y="tip", data=tips)

         # Add a title
         plt.title("Scatter Plot of Total Bill vs. Tip")
```



India's Most Affordable Pay After Placement Data Analytics Course

+91-7880-113-112 Contact or Fill the Form in the Description

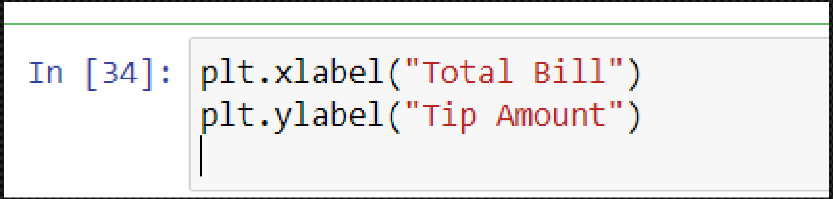
Day - 3

10 Days Python Data Analytics Interview Class

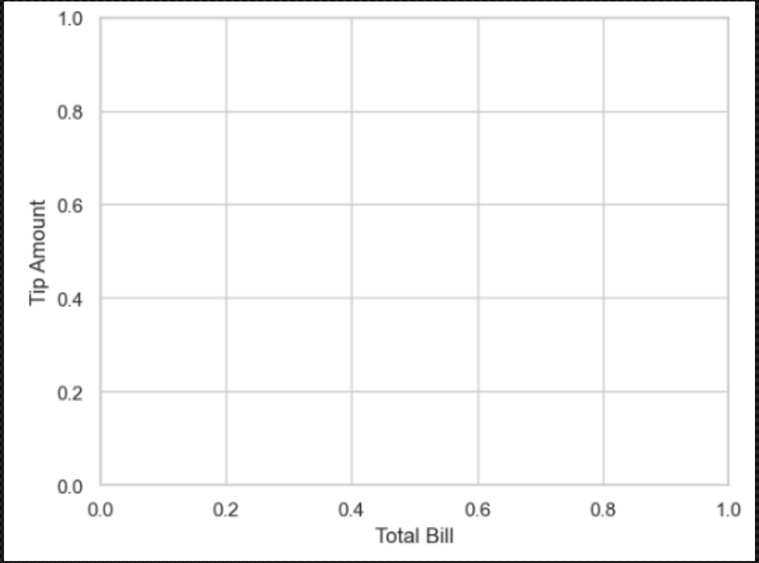
Industries Helping Hands

Interview Question - 6

Axis Labels: You can label the x and y axes using plt.xlabel() and plt.ylabel() functions. For example:



```
In [34]: plt.xlabel("Total Bill")
         plt.ylabel("Tip Amount")
```



Day - 3

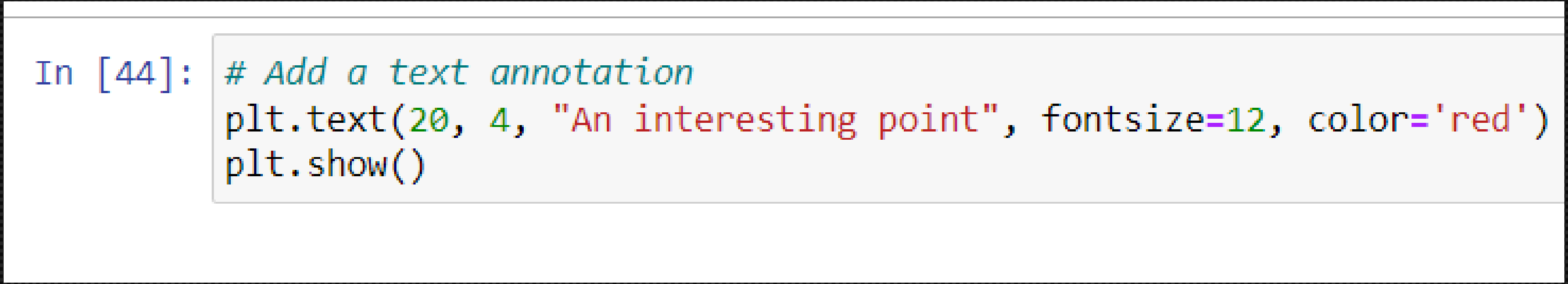
10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 6

Annotations:

Text Annotations: You can add text annotations to specific points on the plot using the plt.text() function. Provide the x and y coordinates where you want to place the text and the text itself. For example:



```
In [44]: # Add a text annotation
         plt.text(20, 4, "An interesting point", fontsize=12, color='red')
         plt.show()
```

10 Days Python Data Analytics Interview Class

Interview Question - 6

Legends:

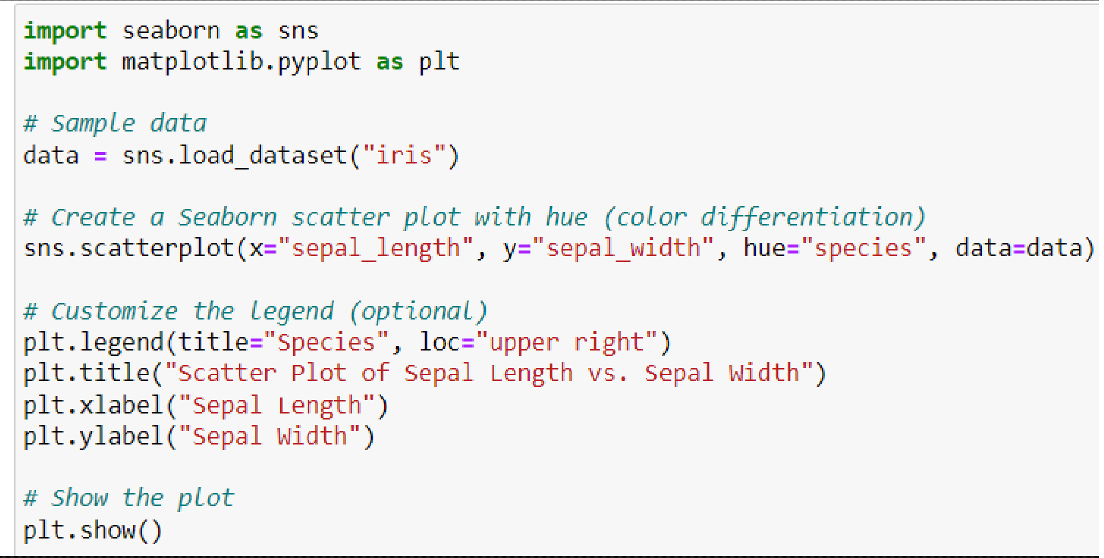
When you have multiple data series or categories on the same plot, you can add a legend to differentiate them. Seaborn often handles legends automatically, but you can customize them using plt.legend(). For example:

Day - 3

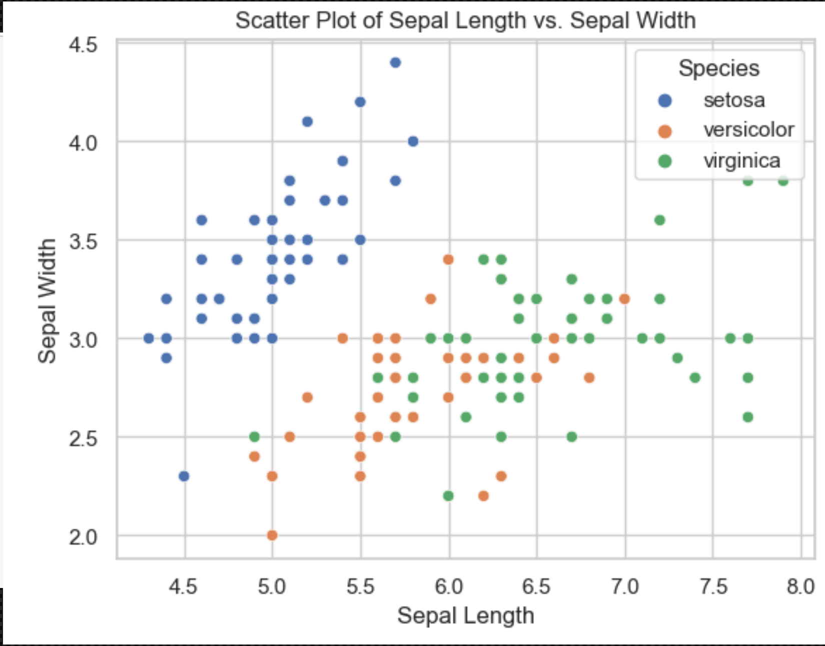
10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 6



```
import seaborn as sns
import matplotlib.pyplot as plt

# Sample data
data = sns.load_dataset("iris")

# Create a Seaborn scatter plot with hue (color differentiation)
sns.scatterplot(x="sepal_length", y="sepal_width", hue="species", data=data)

# Customize the legend (optional)
plt.legend(title="Species", loc="upper right")
plt.title("Scatter Plot of Sepal Length vs. Sepal Width")
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")

# Show the plot
plt.show()
```



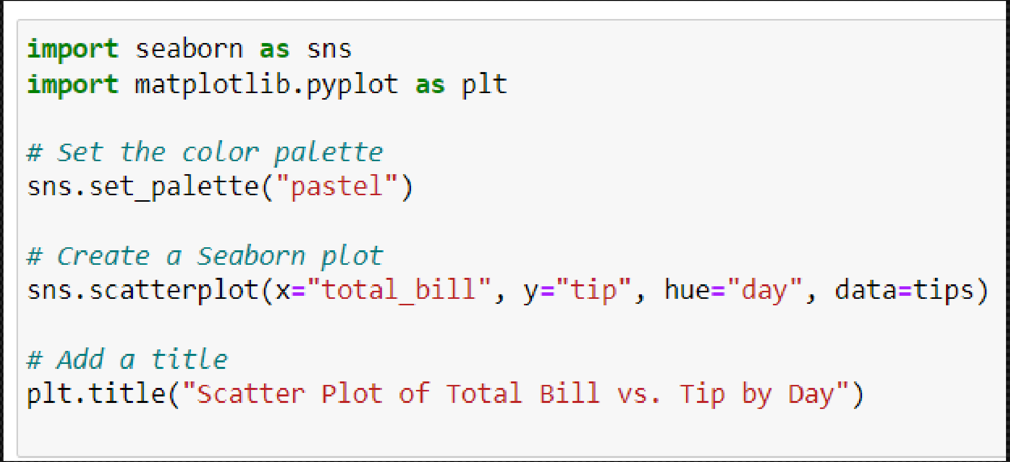
Day - 3

10 Days Python Data Analytics Interview Class

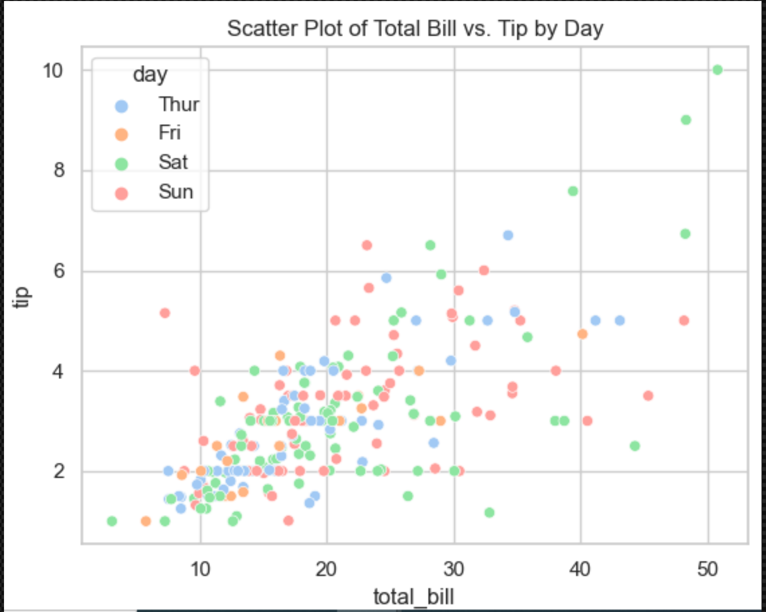
Interview Question - 6

1. Color Palettes:

Seaborn offers several built-in color palettes that you can use to set the color scheme of your plots.



```python
import seaborn as sns
import matplotlib.pyplot as plt

# Set the color palette
sns.set_palette("pastel")

# Create a Seaborn plot
sns.scatterplot(x="total_bill", y="tip", hue="day", data=tips)

# Add a title
plt.title("Scatter Plot of Total Bill vs. Tip by Day")
```



Day - 3

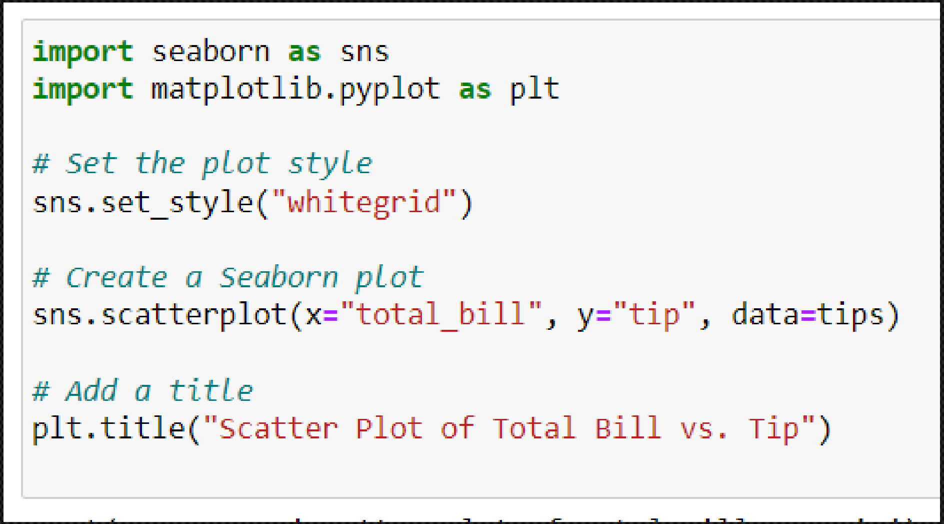
10 Days Python Data Analytics Interview Class

Industries Helping Hands

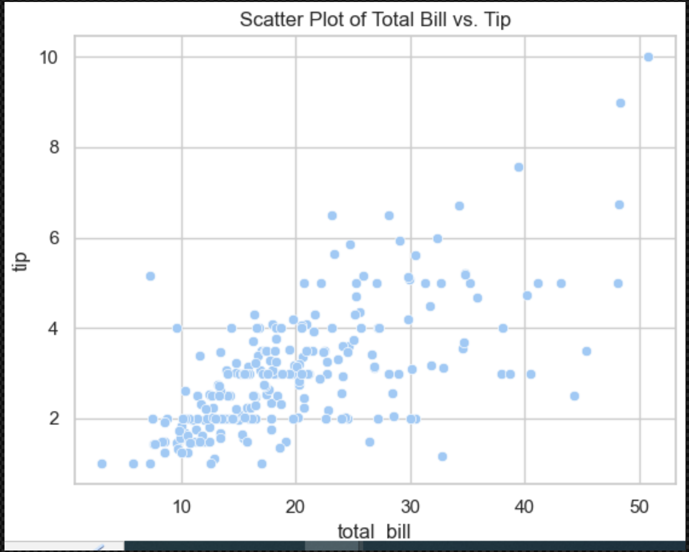
Interview Question - 6

Styles:

Seaborn also provides different plotting styles that you can apply to your plots. Common styles include "whitegrid," "darkgrid," "white," "dark," and "ticks."



```
import seaborn as sns
import matplotlib.pyplot as plt

# Set the plot style
sns.set_style("whitegrid")

# Create a Seaborn plot
sns.scatterplot(x="total_bill", y="tip", data=tips)

# Add a title
plt.title("Scatter Plot of Total Bill vs. Tip")
```



India’s Most Affordable Pay After Placement Data Analytics Course

+91-7880-113-112 Contact or Fill the Form in the Description

Day - 3

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Question - 1

What is the purpose of broadcasting in NumPy?

A. Broadcasting allows you to broadcast live data streams.

B. Broadcasting is used to synchronize multiple CPUs.

C. Broadcasting enables element-wise operations on arrays with different shapes.

D. Broadcasting refers to sending data over a network.