**Q1. What is the distinction between a numpy array and a pandas data frame? Is there a way to convert between the two if there is?**

Ans- The main distinction between a numpy array and a pandas DataFrame is their underlying structure and functionality.

Numpy array: It is a multi-dimensional, homogeneous array that stores elements of the same data type. Numpy arrays are efficient for numerical computations and provide a wide range of mathematical operations.

Pandas DataFrame: It is a two-dimensional, heterogeneous data structure that can store data of different types. It provides more flexibility in terms of handling and manipulating tabular data. DataFrames have built-in functionality for handling missing data, indexing, grouping, merging, and performing various operations on the data.

You can convert a numpy array to a pandas DataFrame using the pandas.DataFrame() constructor, passing the numpy array as the data argument. Similarly, you can convert a pandas DataFrame to a numpy array using the ‘DataFrame.to\_numpy()’ method.

**Q2. What can go wrong when an user enters in a stock-ticker symbol, and how do you handle it?**

Ans- Several things can go wrong when a user enters a stock-ticker symbol. The symbol might not exist, or there could be a typo in the symbol entered. It could also be the case that the symbol is not recognised by the data source or that there is no data available for that symbol.

To handle these situations, we can implement error handling and input validation. We can check if the symbol entered exists and if it is recognised by the data source before proceeding with any analysis. We can also provide informative error messages to the user to help them correct their input.

**Q3. Identify some of the plotting techniques that are used to produce a stock-market chart.**

Ans- Some of the commonly used plotting techniques for producing stock-market charts include:

1. Line charts: These are simple charts that show the stock price over a period of time as a continuous line.
2. Candlestick charts: These charts show the stock price as a series of candlesticks, with each candlestick representing a specific time period. They include information such as opening and closing prices, as well as highs and lows.
3. Bar charts: Similar to candlestick charts, bar charts display the opening and closing prices of a stock, as well as the high and low prices, using vertical bars.
4. Scatter plots: These plots show the correlation between two variables, such as stock price and trading volume, using individual data points.

**Q4. Why is it essential to print a legend on a stock market chart?**

Ans- It is essential to print a legend on a stock market chart to provide a key for interpreting the information displayed. The legend typically indicates the meaning of different lines, colours, or patterns used in the chart. It helps the viewers understand the representation of different data elements, such as stock prices, moving averages, or technical indicators, improving the overall clarity and comprehension of the chart.

**Q5. What is the best way to limit the length of a pandas data frame to less than a year?**

Ans- To limit the length of a pandas DataFrame to less than a year, you can filter the DataFrame based on the date range. Assuming you have a column named 'date' containing the date values, you can use the following code:

import pandas as pd

start\_date = pd.Timestamp('YYYY-MM-DD') # Specify the start date

end\_date = pd.Timestamp('YYYY-MM-DD') # Specify the end date

filtered\_df = df[(df['date'] >= start\_date) & (df['date'] <= end\_date)]

This code filters the DataFrame to include only rows where the 'date' column falls within the specified range.

**Q6. What is the definition of a 180-day moving average?**

Ans- A 180-day moving average is a technical analysis indicator that calculates the average price of a security over the last 180 days. It is also known as a simple moving average (SMA) and is used to smooth out price fluctuations and identify trends in the market. The 180-day moving average is calculated by taking the sum of the closing prices over the last 180 days and dividing by 180.

**Q7. Did the chapter's final example use "indirect" importing? If so, how exactly do you do it?**

Ans- es, the chapter's final example used "indirect" importing. Indirect importing allows us to import modules dynamically based on certain conditions or user input. In the final example, the data\_source variable is used to determine which module to import for the stock data. We can achieve this by using the ‘importlib’ module's ‘import\_module’ function, which allows us to import a module