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?

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

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

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

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

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

Q7. Did the chapter’s final example use &quot;indirect&quot; importing? If so, how exactly do you do it?

### **Q1. What is the distinction between a NumPy array and a pandas DataFrame? Is there a way to convert between the two if there is?**

**Distinction:**

* **NumPy Array:**
  + A NumPy array is a homogeneous, multidimensional array of fixed-size items. All elements in a NumPy array must be of the same data type.
  + It is optimized for numerical operations and is the foundation for numerical computing in Python.
  + NumPy arrays do not have labels for rows and columns.
* **pandas DataFrame:**
  + A pandas DataFrame is a two-dimensional, size-mutable, and potentially heterogeneous tabular data structure with labeled axes (rows and columns).
  + It is built on top of NumPy arrays but provides more flexibility and functionality, especially for data manipulation and analysis.
  + DataFrames can hold different data types in different columns and allow for missing data.

**Conversion:**

**From NumPy Array to pandas DataFrame:**python  
  
import numpy as np

import pandas as pd

np\_array = np.array([[1, 2, 3], [4, 5, 6]])

df = pd.DataFrame(np\_array, columns=['A', 'B', 'C'])

**From pandas DataFrame to NumPy Array:**python  
  
np\_array = df.to\_numpy() # or df.values (older method)

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

**Potential Issues:**

1. **Invalid Ticker Symbol:** The user may enter a non-existent or misspelled ticker symbol.
2. **Delisted or Inactive Symbol:** The ticker symbol may refer to a company that has been delisted or is no longer active.
3. **Case Sensitivity:** The ticker symbol may be case-sensitive, leading to errors if entered incorrectly.
4. **Network or API Errors:** Issues like network failure or API request limits may prevent retrieving the data.

**Handling Strategies:**

* **Validation:** Use regular expressions or predefined lists of valid ticker symbols to validate user input.
* **Error Handling:** Implement try-except blocks to handle exceptions and provide meaningful error messages.
* **API Response Check:** Check the API response to ensure the data is retrieved successfully before proceeding.
* **User Feedback:** If the input is invalid, prompt the user to re-enter the ticker symbol or provide suggestions.

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

* **Line Plot:** To show the trend of stock prices over time, typically with dates on the x-axis and prices on the y-axis.
* **Candlestick Chart:** Used to visualize the open, high, low, and close prices of a stock over a period, providing insights into price movements and trends.
* **Volume Bar Chart:** Plotted alongside price charts to show trading volume, often as bars at the bottom of the chart.
* **Moving Averages Plot:** Overlaying moving averages on the price chart to smooth out price data and identify trends.
* **Bollinger Bands:** A technique that plots two standard deviations away from a simple moving average, used to visualize volatility.

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

A legend is essential on a stock market chart because it:

* **Identifies Lines and Indicators:** Helps differentiate between multiple lines or indicators, such as different moving averages, stock prices, volume, etc.
* **Enhances Clarity:** Provides context to the viewer, making the chart easier to understand, especially when multiple data series are plotted.
* **Reduces Ambiguity:** Prevents confusion by clearly labeling each component of the chart, ensuring that the data is interpreted correctly.

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

You can filter the DataFrame based on the date range to limit the length to less than a year. Assuming you have a DataFrame with a datetime index or a date column:

python

import pandas as pd

# Assuming df is your DataFrame and 'Date' is the datetime column

start\_date = '2023-01-01'

end\_date = '2023-12-31'

# Filter the DataFrame to keep only data within the desired date range

df\_filtered = df[(df['Date'] >= start\_date) & (df['Date'] <= end\_date)]

Alternatively, if the DataFrame index is a datetime index:

python

df\_filtered = df.loc[start\_date:end\_date]

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

A 180-day moving average is a technical indicator that smooths out daily stock price data by averaging the closing prices over the past 180 days. It provides a longer-term view of the stock's price trend, filtering out short-term fluctuations.

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

Without access to the specific chapter mentioned, I can explain the concept of "indirect" importing. Indirect importing refers to a situation where a module is imported not directly in the current module, but through another module that is imported.

**Example:**

python

# module\_a.py

import module\_b # Direct import

python

# module\_b.py

import module\_c # Indirect import from the perspective of module\_a

In this case, when module\_a.py is executed, it indirectly imports module\_c through module\_b. Indirect importing is typically handled by ensuring that dependencies are properly structured and that circular imports are avoided.

If the chapter used indirect importing, it likely involved a scenario similar to the one above, where one module relied on another to import a third module.