**PYTHON ADVANCE ASSIGNMENT\_25**

**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?**

Numpy arrays and Pandas DataFrames are both popular data structures used in data analysis and manipulation, but they have some important differences:

Dimensionality: Numpy arrays are homogeneous, multidimensional arrays of fixed size, while Pandas DataFrames are heterogeneous, two-dimensional tables with labeled axes (rows and columns).

Data types: Numpy arrays are designed to handle numerical data, so they are typically used to store and manipulate arrays of numbers. In contrast, Pandas DataFrames can handle a variety of data types, including numerical, categorical, and text data.

Indexing and Selection: Numpy arrays are indexed using integer indices or boolean masks, while Pandas DataFrames can be indexed using labels or integer indices, and can also be sliced by rows and columns.

Missing Data: Numpy arrays do not have a built-in way to handle missing data, while Pandas DataFrames have several built-in methods for handling missing data.

To convert between a Numpy array and a Pandas DataFrame, you can use the pd.DataFrame() function to create a DataFrame from a Numpy array, and the to\_numpy() method to convert a DataFrame to a Numpy array. Here's an example:

import numpy as np

import pandas as pd

# create a Numpy array

arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# convert Numpy array to Pandas DataFrame

df = pd.DataFrame(arr, columns=['A', 'B', 'C'])

print(df)

# convert Pandas DataFrame to Numpy array

arr2 = df.to\_numpy()

print(arr2)

This will output:

A B C

0 1 2 3

1 4 5 6

2 7 8 9

[[1 2 3]

[4 5 6]

[7 8 9]]

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

There are several things that can go wrong when a user enters a stock ticker symbol. Here are a few examples:

Incorrect Ticker Symbol: The user may enter an incorrect or misspelled ticker symbol, which could result in an error or an incorrect result.

Outdated Ticker Symbol: Ticker symbols can change over time, and an old or outdated symbol may not be recognized by the system.

Unsupported Ticker Symbol: Some ticker symbols may not be supported by the system, particularly if the stock is not traded on a major exchange.

Lack of Data: Some ticker symbols may not have sufficient data available for analysis, particularly for newer or smaller companies.

To handle these situations, you can implement some of the following strategies:

Provide Auto-Complete Suggestions: As the user types in the stock ticker symbol, you can provide auto-complete suggestions to help them avoid misspellings or incorrect symbols.

Validate the Ticker Symbol: You can use an API to validate the ticker symbol entered by the user to ensure that it is correct and up-to-date.

Provide Error Messages: If the ticker symbol entered by the user is incorrect or unsupported, you can provide an error message explaining the issue and suggesting a solution.

Provide Additional Information: If there is not enough data available for the ticker symbol, you can provide additional information to the user, such as the last available trading price or the date of the last trade.

By implementing these strategies, you can help ensure that the user has a smooth and error-free experience when entering a stock ticker symbol.

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

There are several plotting techniques that are commonly used to produce a stock-market chart. Here are some of the most common ones:

Line Chart: A line chart is the most common type of chart used in the stock market. It shows the price of a stock over a period of time, with a continuous line connecting the data points.

Bar Chart: A bar chart is also commonly used in the stock market. It displays the opening and closing prices, as well as the high and low prices for each day or time period. Each bar represents a single day or time period.

Candlestick Chart: A candlestick chart is a type of chart used in technical analysis. It displays the same information as a bar chart, but in a more visually appealing way. Each candlestick represents a single day or time period, and shows the opening and closing prices as well as the high and low prices. The body of the candlestick is filled if the closing price is higher than the opening price, and empty if the closing price is lower than the opening price.

Area Chart: An area chart is another type of chart used in the stock market. It shows the price of a stock over a period of time, with the area under the curve filled in. This type of chart can be useful for showing overall trends in the price of a stock.

Scatter Plot: A scatter plot can be used to show the relationship between two variables, such as the price of a stock and the volume of shares traded. Each point on the plot represents a single day or time period.

These are just a few of the most common plotting techniques used to produce a stock-market chart. Other techniques can also be used depending on the specific data being analyzed and the information being conveyed.

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

A legend on a stock market chart is essential because it provides critical information about the chart's data and allows the viewer to understand the chart better. A legend typically contains labels that explain the meaning of each line or symbol on the chart.

In the context of a stock market chart, a legend may include information such as:

Ticker Symbols: Ticker symbols are abbreviations used to identify individual stocks, and they are often displayed on stock charts. The legend provides a key for identifying each ticker symbol represented on the chart.

Timeframe: The legend may also provide information about the timeframe displayed on the chart. This can be especially important for investors who need to compare different time periods or track trends over time.

Prices: The legend may include information about the price scale used on the chart, including the minimum and maximum values displayed.

Indicators: Many stock charts include technical indicators, such as moving averages or relative strength index (RSI). The legend can explain what these indicators represent and how they are calculated.

By providing this information, the legend helps viewers to understand the chart and interpret the data more effectively. It enables them to make more informed investment decisions based on the information presented. Without a legend, the viewer may be left confused about what the chart is showing, leading to incorrect or incomplete conclusions.

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

To limit the length of a pandas DataFrame to less than a year, you can filter the DataFrame to only include rows that fall within a specific date range. Here are the steps you can follow:

Convert the index of the DataFrame to a DatetimeIndex if it's not already a DatetimeIndex.

df.index = pd.to\_datetime(df.index)

Determine the start and end dates for the range you want to keep. For example, if you want to keep data from January 1st to December 31st of a specific year, you can use the following code:

start\_date = pd.to\_datetime("YYYY-01-01")

end\_date = pd.to\_datetime("YYYY-12-31")

Filter the DataFrame using the start and end dates:

filtered\_df = df[(df.index >= start\_date) & (df.index <= end\_date)]

This will create a new DataFrame, filtered\_df, that contains only the rows that fall within the specified date range.

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

A 180-day moving average in Python is a statistical calculation used to analyze time-series data. It is a rolling average that calculates the average value of a variable over the previous 180 days, and then shifts the window by one day and recalculates the average for the new 180-day period. This process continues for the entire length of the time series.

To calculate the 180-day moving average in Python, you can use the Pandas library, which provides a rolling() function that can be used to apply rolling window calculations to data.

Here's an example code snippet that demonstrates how to calculate the 180-day moving average for a time series using Pandas:

import pandas as pd

# create a sample time series

data = pd.Series([10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40])

# calculate the 180-day moving average using the rolling() function

moving\_average = data.rolling(window=180).mean()

# print the results

print(moving\_average)

In this example, we first create a Pandas Series object called data that contains some sample time series data. We then use the rolling() function to calculate the moving average with a window size of 180 days. Finally, we print the resulting moving average values.

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