Week 3 Attempt Summary

This week we were tasked with moving from our v1 code base to the v2 code base

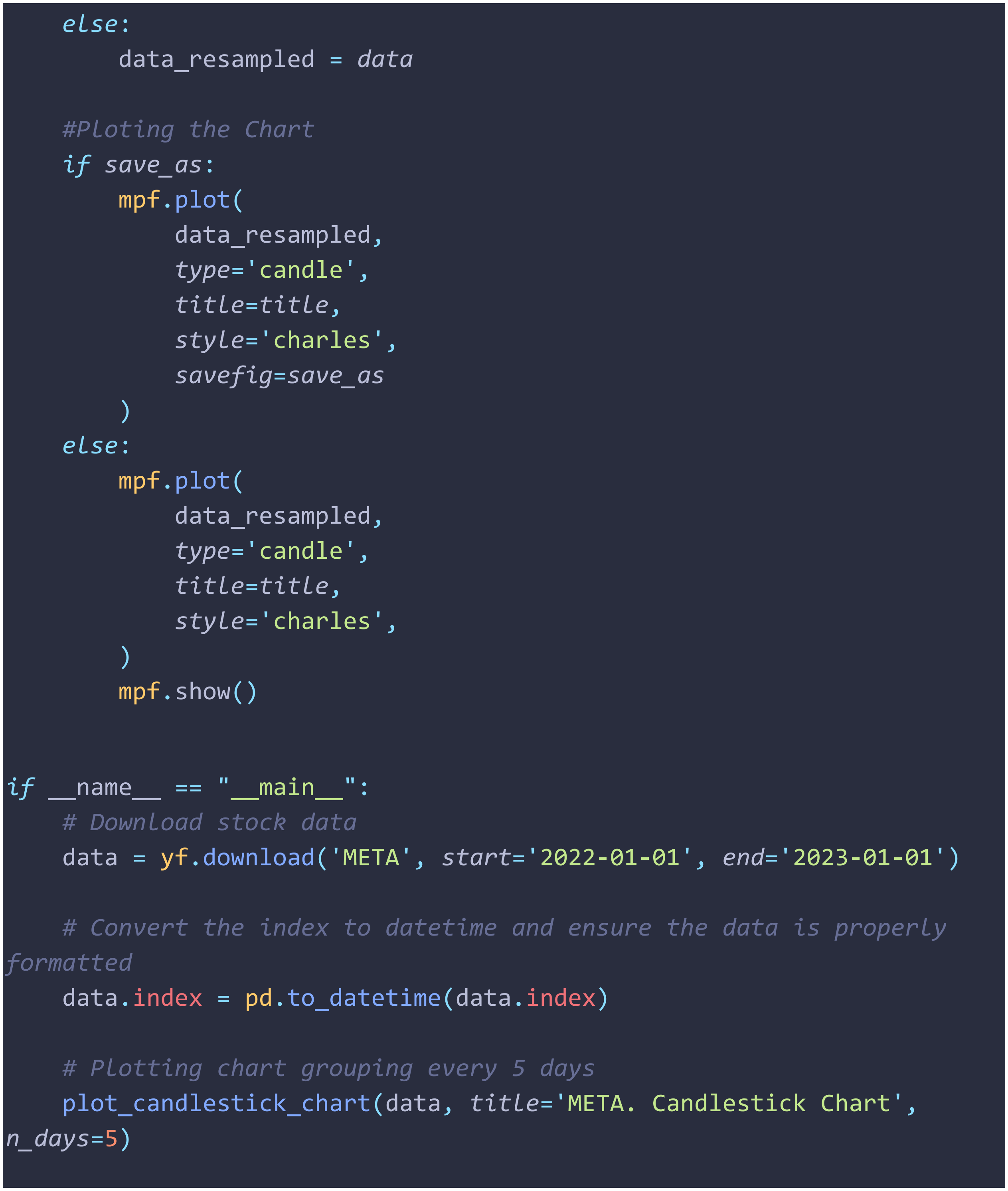
- Create a candle Stick chart to display the stock market financial data
- Create a boxplot chart to display the stock market financial data

Creating the Candel Stick Chart

The online resource that was give to us uses yet another new library called the **“mplfinance”.** This library is a financial charting library developed by the developers of **“matplotlib”.**

Code for the Candel Stick Chart

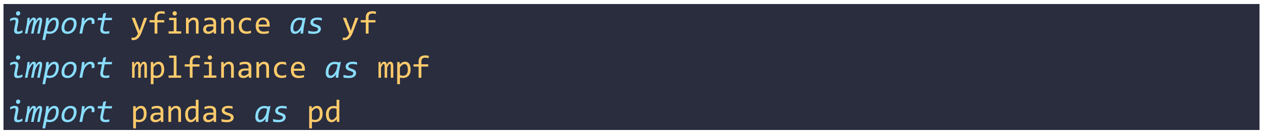
```
import yfinance as yf
import mplfinance as mpf
import pandas as pd

def plot_candlestick_chart(data, title='Candlestick Chart', n_days=1,
save_as=None):

    if n_days < 1:
        raise ValueError("n_days must be greater than or equal to 1.")

    # Resample the data if n_days > 1
    if n_days > 1:
        data_resampled = data.resample(f'{n_days}D').agg({
            'Open': 'first',
            'High': 'max',
            'Low': 'min',
            'Close': 'last',
            'Volume': 'sum'
        }).dropna()
```



```
    else:
        data_resampled = data

    #Ploting the Chart
    if save_as:
        mpf.plot(
            data_resampled,
            type='candle',
            title=title,
            style='charles',
            savefig=save_as
        )
    else:
        mpf.plot(
            data_resampled,
            type='candle',
            title=title,
            style='charles',
        )
        mpf.show()


if __name__ == "__main__":
    # Download stock data
    data = yf.download('META', start='2022-01-01', end='2023-01-01')

    # Convert the index to datetime and ensure the data is properly
formatted
    data.index = pd.to_datetime(data.index)

    # Plotting chart grouping every 5 days
    plot_candlestick_chart(data, title='META. Candlestick Chart',
n_days=5)
```

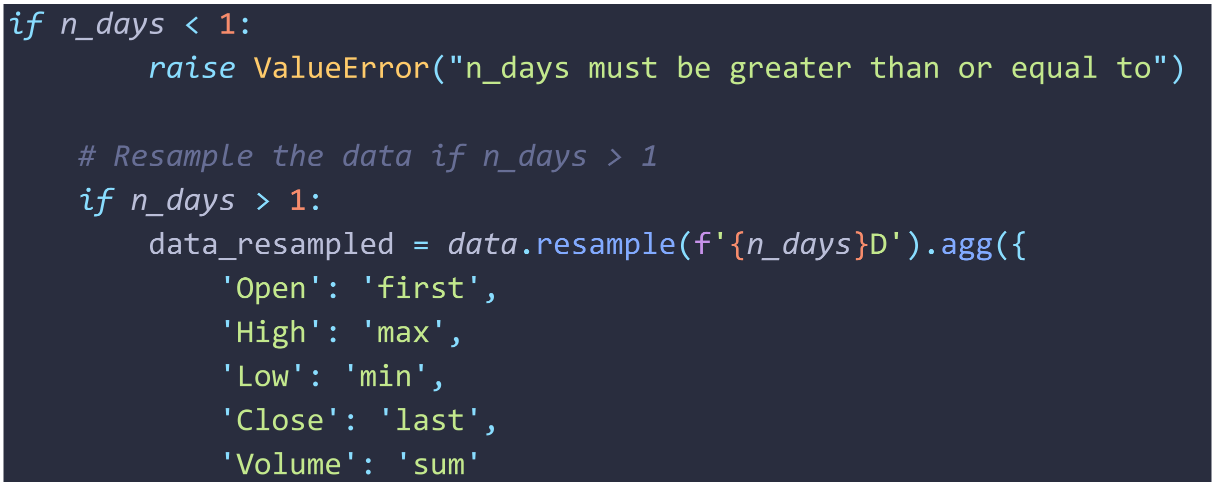
How the code works

1. The code will first import the necessary libraries



```
import yfinance as yf
import mplfinance as mpf
import pandas as pd
```

2. Then the function definition of the plot_candelstick_chart function
3. Then the function will check for the trading days (n_days)

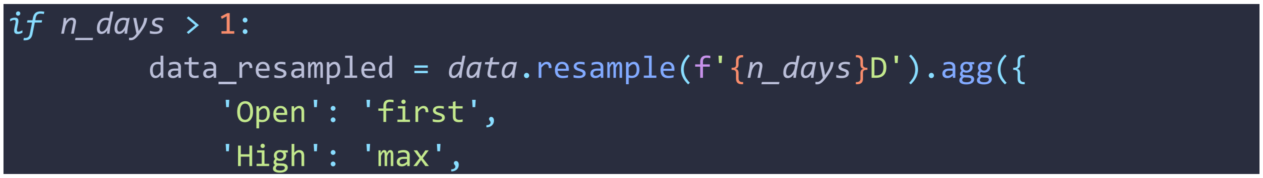


```
if n_days < 1:
        raise ValueError("n_days must be greater than or equal to")

    # Resample the data if n_days > 1
    if n_days > 1:
        data_resampled = data.resample(f'{n_days}D').agg({
            'Open': 'first',
            'High': 'max',
            'Low': 'min',
            'Close': 'last',
            'Volume': 'sum'
```

If the function finds that the n_days variable is less than 1, it will raise an error massage because it’s not possible to have a valid chart that represents less than 1 day

4. Data resampling – This will pile the data into larger intervals, if the ‘n_days = 5’ it will group every 5 treading days into a single candlestick

5. (agg)is a method use to “Aggregate” data after the resampling process



```
if n_days > 1:
        data_resampled = data.resample(f'{n_days}D').agg({
            'Open': 'first',
            'High': 'max',
            'Low': 'min',
            'Close': 'last',
            'Volume': 'sum'
        }).dropna()
```

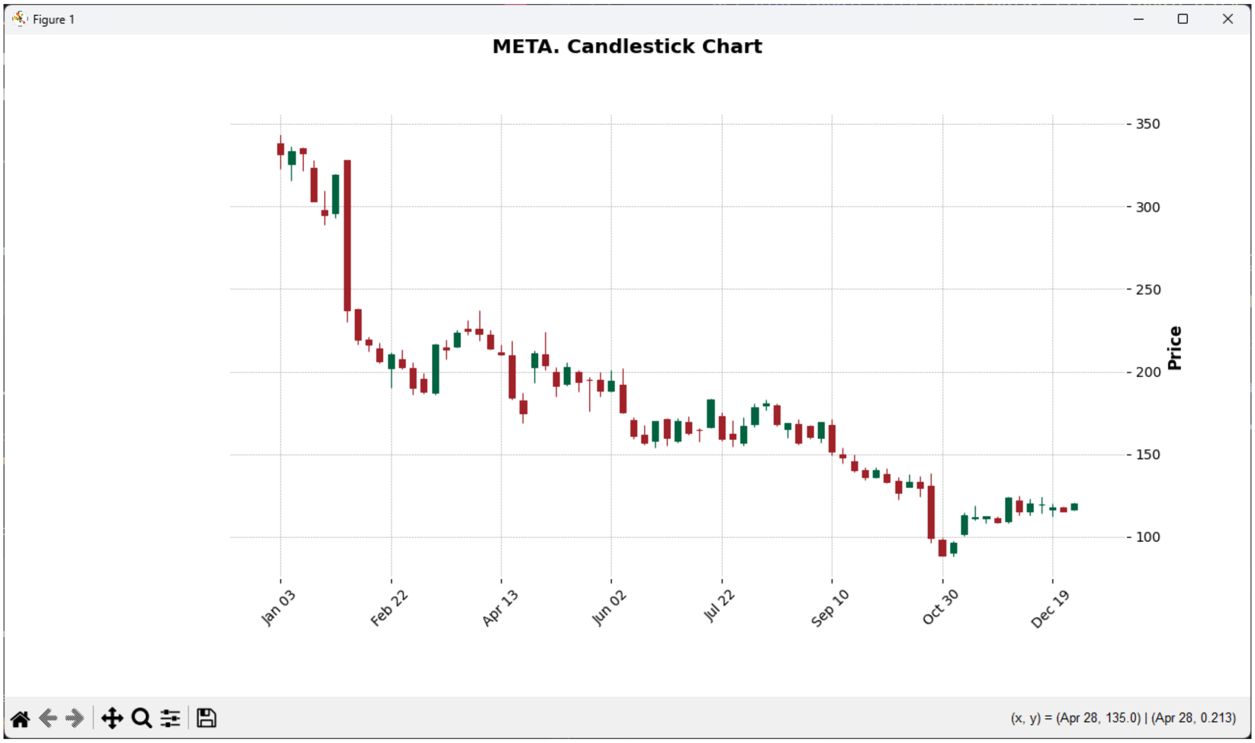
6. Plotting the chart

```
#Ploting the Chart
    if save_as:
        mpf.plot(
            data_resampled,
            type='candle',
            title=title,
            style='charles',
            savefig=save_as
        )
    else:
        mpf.plot(
            data_resampled,
            type='candle',
            title=title,
            style='charles',
        )
        mpf.show()
```

7. Then from this line of code, it will ensure that the Data Frame’s index is correctly set date-time format, this is important for the resampling and the plotting of the chart

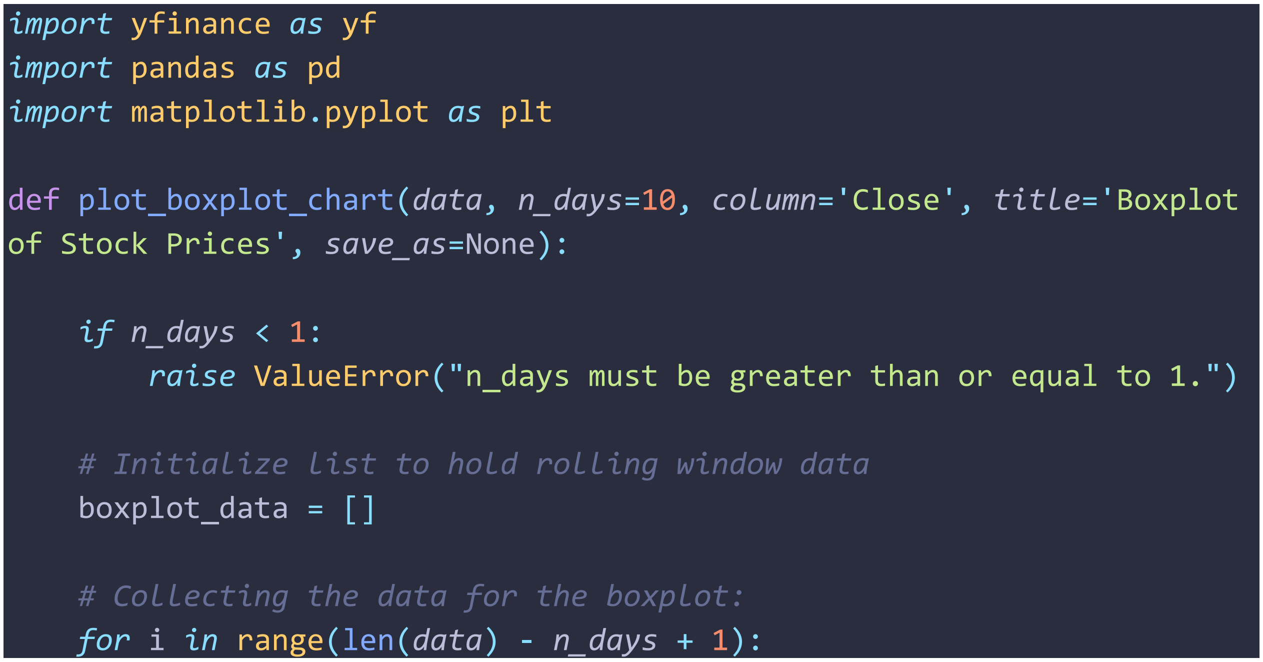
```
data.index = pd.to_datetime(data.index)
```

Output of the code

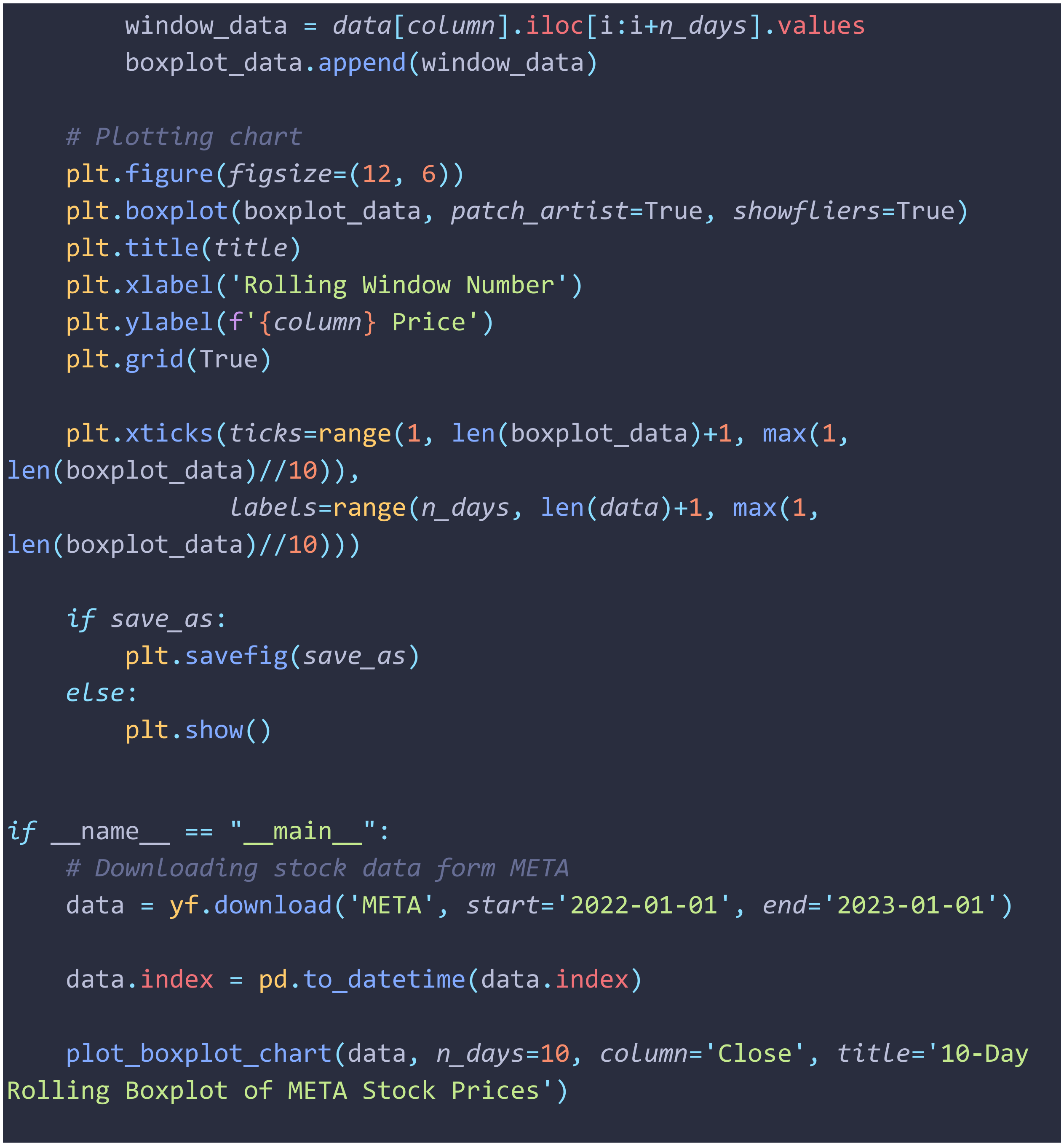


Creating the Boxplot chart for META stock market

Code Used



```
import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt

def plot_boxplot_chart(data, n_days=10, column='Close', title='Boxplot
of Stock Prices', save_as=None):

    if n_days < 1:
        raise ValueError("n_days must be greater than or equal to 1.")

    # Initialize list to hold rolling window data
    boxplot_data = []

    # Collecting the data for the boxplot:
    for i in range(len(data) - n_days + 1):
```



```
        window_data = data[column].iloc[i:i+n_days].values
        boxplot_data.append(window_data)

    # Plotting chart
    plt.figure(figsize=(12, 6))
    plt.boxplot(boxplot_data, patch_artist=True, showfliers=True)
    plt.title(title)
    plt.xlabel('Rolling Window Number')
    plt.ylabel(f'{column} Price')
    plt.grid(True)

    plt.xticks(ticks=range(1, len(boxplot_data)+1, max(1,
len(boxplot_data)//10)),
               labels=range(n_days, len(data)+1, max(1,
len(boxplot_data)//10)))

    if save_as:
        plt.savefig(save_as)
    else:
        plt.show()


if __name__ == "__main__":
    # Downloading stock data form META
    data = yf.download('META', start='2022-01-01', end='2023-01-01')

    data.index = pd.to_datetime(data.index)

    plot_boxplot_chart(data, n_days=10, column='Close', title='10-Day
Rolling Boxplot of META Stock Prices')
```

How the code works

1. First the code starts by Importing the libraries

```
import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt
```

2. Then we move to the function definition of the 'plot_boxplot_chart'

```
def plot_boxplot_chart(data, n_days=10, column='Close', title='Boxplot
of Stock Prices', save_as=None):

    if n_days < 1:
        raise ValueError("n_days must be greater than or equal to 1.")

    # Initialize list to hold rolling window data
    boxplot_data = []

    # Collecting the data for the boxplot:
    for i in range(len(data) - n_days + 1):
        window_data = data[column].iloc[i:i+n_days].values
        boxplot_data.append(window_data)

    # Plotting chart
    plt.figure(figsize=(12, 6))
    plt.boxplot(boxplot_data, patch_artist=True, showfliers=True)
    plt.title(title)
    plt.xlabel('Rolling Window Number')
    plt.ylabel(f'{column} Price')
    plt.grid(True)

    plt.xticks(ticks=range(1, len(boxplot_data)+1, max(1,
len(boxplot_data)//10)),
               labels=range(n_days, len(data)+1, max(1,
len(boxplot_data)//10)))

    if save_as:
        plt.savefig(save_as)
    else:
        plt.show()
```

3. This function is also using the same logic to determine whether it’s a valid graph or not.
4. Then a for loop in initiated to extract the rolling windows data of ‘n_days’ from the ‘data’ Data frame and store those windows data in a list called the ‘boxplot_data’

```
for i in range(len(data) - n_days + 1):
        window_data = data[column].iloc[i:i+n_days].values
        boxplot_data.append(window_data)
```

5. In the 2nd line of code in the for loop, we use the iloc method slice the dataset form the index ‘i’ to ***“i + n_days ”*** . This slice will allows us to have the stock price data for window of ‘n_days’ (Trading Days)

6. Then we move to plotting the chart

```
     plt.figure(figsize=(12, 6))

    plt.boxplot(boxplot_data, patch_artist=True, showfliers=True)
    plt.title(title)
    plt.xlabel('Rolling Window Number')
    plt.ylabel(f'{column} Price')
    plt.grid(True)

    plt.xticks(ticks=range(1, len(boxplot_data)+1, max(1, 
len(boxplot_data)//10)),
               labels=range(n_days, len(data)+1, max(1, 
len(boxplot_data)//10)))

    if save_as:
        plt.savefig(save_as)
    else:
        plt.show()
```

The first few statements are standard for plotting charts

```
 plt.xticks(ticks=range(1, len(boxplot_data)+1, max(1,
len(boxplot_data)//10)),
                   labels=range(n_days, len(data)+1, max(1,
len(boxplot_data)//10)))
```

But this is not standard, this function is used to customize the ticks on the x – axis. Ticks are similar to spacings between two units.

Output of the code

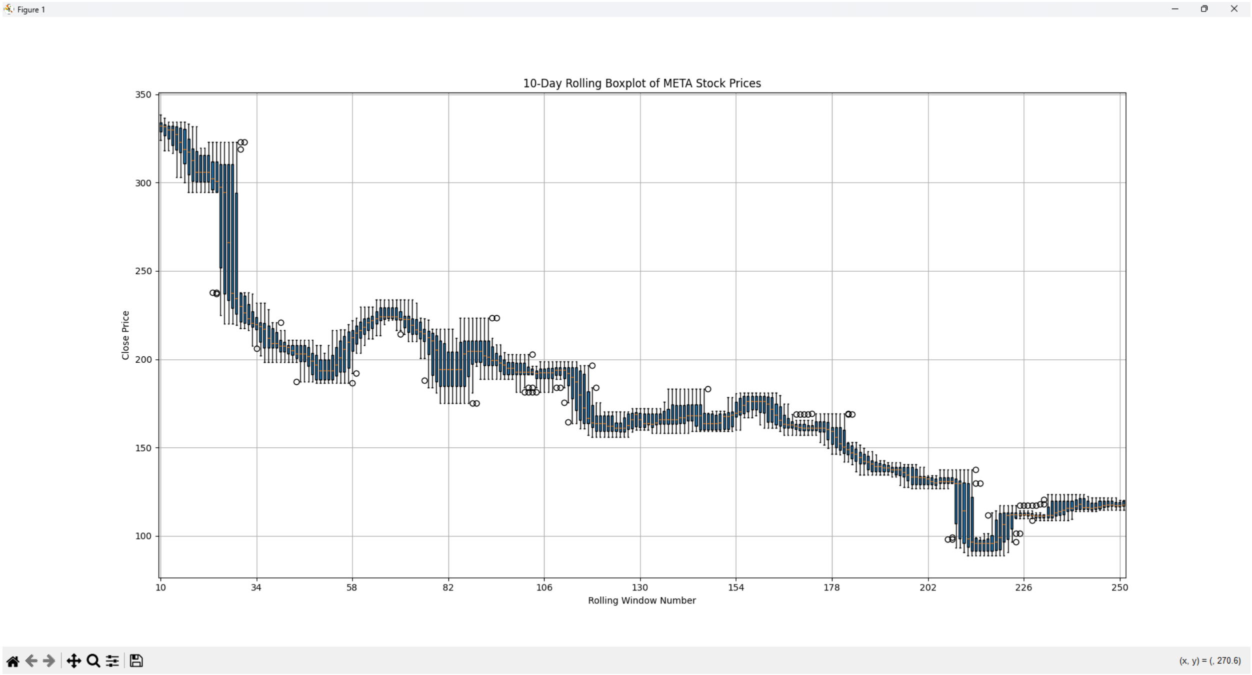


Figure 1
10-Day Rolling Boxplot of META Stock Prices
Close Price
350
300
250
200
150
100
10
34
58
82
106
130
154
178
202
226
250
Rolling Window Number
(x, y) = (, 270.6)

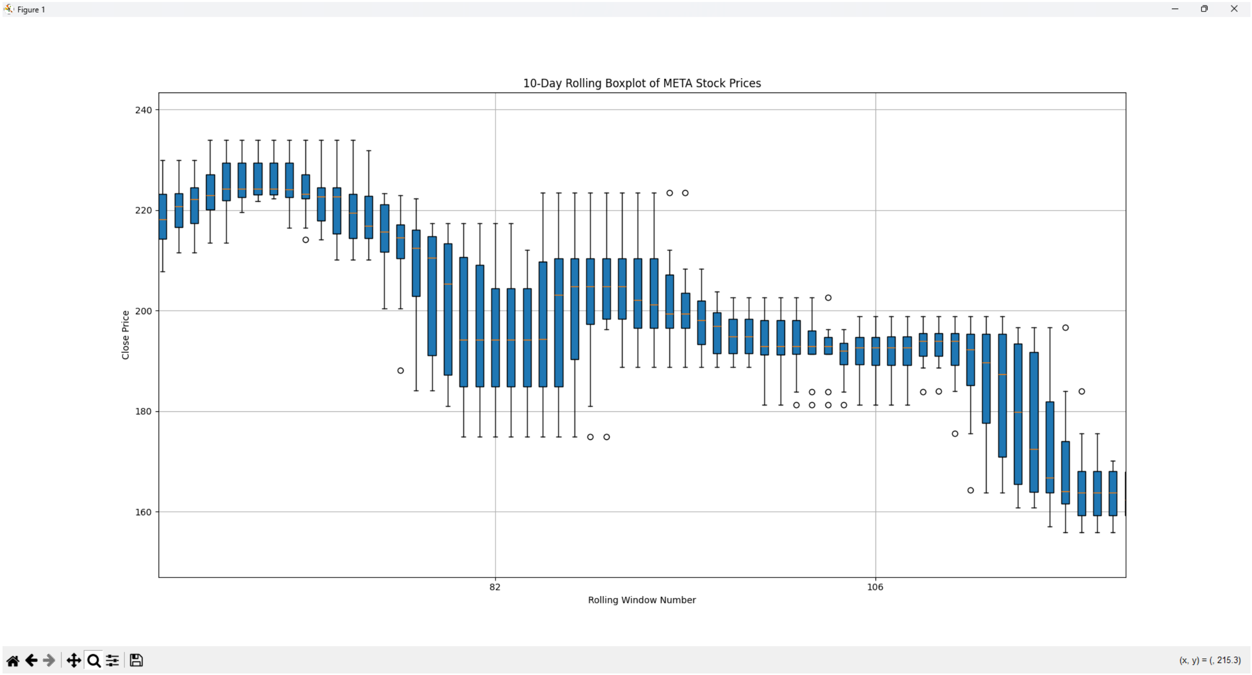


Figure 1
10-Day Rolling Boxplot of META Stock Prices
Close Price
240
220
200
180
160
82
106
Rolling Window Number
(x, y) = (, 215.3)