Assigment Final

July 21, 2025

[1]:
```
# Step 1: Import the required library
import yfinance as yf

# Step 2: Create a Ticker object for Tesla (TSLA)
tesla = yf.Ticker("TSLA")

# Step 3: Extract historical stock data with maximum period
tesla_data = tesla.history(period="max")

# Step 4: Reset the index of the DataFrame
tesla_data.reset_index(inplace=True)

# Step 5: Display the first five rows of the tesla_data DataFrame
print(tesla_data.head())
```

```
                       Date      Open      High       Low     Close  \
0 2010-06-29 00:00:00-04:00  1.266667  1.666667  1.169333  1.592667
1 2010-06-30 00:00:00-04:00  1.719333  2.028000  1.553333  1.588667
2 2010-07-01 00:00:00-04:00  1.666667  1.728000  1.351333  1.464000
3 2010-07-02 00:00:00-04:00  1.533333  1.540000  1.247333  1.280000
4 2010-07-06 00:00:00-04:00  1.333333  1.333333  1.055333  1.074000

      Volume  Dividends  Stock Splits
0  281494500        0.0           0.0
1  257806500        0.0           0.0
2  123282000        0.0           0.0
3   77097000        0.0           0.0
4  103003500        0.0           0.0
```

[2]:
```
# Question 2: Webscraping Tesla Revenue Data (Updated for pandas 1.4.0+)

import requests
from bs4 import BeautifulSoup
import pandas as pd

# Step 1: Download the webpage
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
 ↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm"
```

```
html_data = requests.get(url).text

# Step 2: Parse using html.parser
soup = BeautifulSoup(html_data, 'html.parser')

# Step 3: Extract revenue data
def extract_revenue(soup):
    # Find the revenue table - it's the second table on the page
    table = soup.find_all('table')[1]

    # Create empty list to store data
    data = []

    # Extract data from each row
    for row in table.find('tbody').find_all('tr'):
        cols = row.find_all('td')
        if len(cols) >= 2:  # Ensure we have both Date and Revenue columns
            date = cols[0].text.strip()
            revenue = cols[1].text.strip().replace('$', '').replace(',', '')
            if revenue:  # Only add if revenue exists
                data.append({'Date': date, 'Revenue': revenue})

    # Create DataFrame
    revenue_data = pd.DataFrame(data)

    # Convert Revenue to numeric
    revenue_data['Revenue'] = pd.to_numeric(revenue_data['Revenue'])
    return revenue_data

# Extract and display data
tesla_revenue = extract_revenue(soup)
print("Tesla Quarterly Revenue Data:")
print(tesla_revenue.tail())
```

```
Tesla Quarterly Revenue Data:
          Date  Revenue
48  2010-09-30       31
49  2010-06-30       28
50  2010-03-31       21
51  2009-09-30       46
52  2009-06-30       27
```

[3]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd

# Step 1: Download the webpage
```

```
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
 ↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm"
html_data = requests.get(url).text

# Step 2: Parse the HTML
soup = BeautifulSoup(html_data, 'html.parser')

# Step 3: Create empty DataFrame
tesla_revenue = pd.DataFrame(columns=['Date', 'Revenue'])

# Step 4: Find the relevant table (second table containing revenue data)
tables = soup.find_all('table')
revenue_table = tables[1]  # The revenue table is the second table on the page

# Step 5: Iterate through rows in the table body
for row in revenue_table.find('tbody').find_all('tr'):
    # Step 6: Extract data from columns
    cols = row.find_all('td')
    if len(cols) >= 2:  # Ensure we have both Date and Revenue columns
        date = cols[0].text.strip()
        revenue = cols[1].text.strip().replace('$', '').replace(',', '')

        # Append data to DataFrame using concat (modern alternative to append)
        new_row = pd.DataFrame({'Date': [date], 'Revenue': [revenue]})
        tesla_revenue = pd.concat([tesla_revenue, new_row], ignore_index=True)

# Clean and convert data
tesla_revenue['Revenue'] = pd.to_numeric(tesla_revenue['Revenue'],␣
 ↪errors='coerce')
tesla_revenue = tesla_revenue.dropna()  # Remove any empty rows

# Display the results
print("Tesla Quarterly Revenue Data:")
print(tesla_revenue.tail())
```

```
Tesla Quarterly Revenue Data:
          Date  Revenue
48  2010-09-30     31.0
49  2010-06-30     28.0
50  2010-03-31     21.0
52  2009-09-30     46.0
53  2009-06-30     27.0
```

[4]:
```
import pandas as pd

def clean_revenue(df):
    """Clean the revenue column in a dataframe"""
```

```
    # Convert to string if not already
    df['Revenue'] = df['Revenue'].astype(str)

    # Remove commas and dollar signs
    df['Revenue'] = df['Revenue'].str.replace(r'[,$]', '', regex=True)

    # Convert to numeric, coercing errors to NaN
    df['Revenue'] = pd.to_numeric(df['Revenue'], errors='coerce')

    # Drop NA values and empty strings
    df = df.dropna()
    df = df[df['Revenue'] != ""]

    return df

# Apply cleaning
tesla_revenue = clean_revenue(tesla_revenue)

# Display results
print("Last 5 rows of cleaned data:")
print(tesla_revenue.tail())
```

```
Last 5 rows of cleaned data:
          Date  Revenue
48  2010-09-30     31.0
49  2010-06-30     28.0
50  2010-03-31     21.0
52  2009-09-30     46.0
53  2009-06-30     27.0
```

[5]:
```
# Question 3: Extract GameStop Stock Data using yfinance

import yfinance as yf
import pandas as pd

# Step 1: Create a Ticker object for GameStop (GME)
gme = yf.Ticker("GME")

# Step 2: Extract historical stock data with maximum period
gme_data = gme.history(period="max")

# Step 3: Reset the index of the DataFrame
gme_data.reset_index(inplace=True)

# Step 4: Display the first five rows
print("GameStop Stock Data (First 5 rows):")
print(gme_data.head())
```

```
# Step 5: Display the last five rows (optional)
print("\nGameStop Stock Data (Last 5 rows):")
print(gme_data.tail())
```

```
GameStop Stock Data (First 5 rows):
                       Date      Open      High       Low     Close    Volume  \
0 2002-02-13 00:00:00-05:00  1.620128  1.693350  1.603296  1.691666  76216000
1 2002-02-14 00:00:00-05:00  1.712707  1.716074  1.670626  1.683250  11021600
2 2002-02-15 00:00:00-05:00  1.683250  1.687458  1.658001  1.674834   8389600
3 2002-02-19 00:00:00-05:00  1.666418  1.666418  1.578047  1.607504   7410400
4 2002-02-20 00:00:00-05:00  1.615920  1.662210  1.603296  1.662210   6892800

   Dividends  Stock Splits
0        0.0           0.0
1        0.0           0.0
2        0.0           0.0
3        0.0           0.0
4        0.0           0.0

GameStop Stock Data (Last 5 rows):
                          Date       Open       High        Low      Close  \
5891 2025-07-15 00:00:00-04:00  23.660000  23.680000  23.170000  23.219999
5892 2025-07-16 00:00:00-04:00  23.400000  23.850000  23.309999  23.680000
5893 2025-07-17 00:00:00-04:00  23.540001  23.719999  23.379999  23.400000
5894 2025-07-18 00:00:00-04:00  23.459999  23.510000  23.129999  23.280001
5895 2025-07-21 00:00:00-04:00  23.280001  23.391199  23.059999  23.260000

       Volume  Dividends  Stock Splits
5891  7321600        0.0           0.0
5892  7738300        0.0           0.0
5893  7307200        0.0           0.0
5894  6607900        0.0           0.0
5895  1952260        0.0           0.0
```

[6]:
```
# Import required library
import yfinance as yf

# Create ticker object for GameStop (GME)
gme = yf.Ticker("GME")

# Extract maximum historical data and save to DataFrame
gme_data = gme.history(period="max")

# Display the first 5 rows to verify
print("GameStop Historical Stock Data (First 5 rows):")
print(gme_data.head())
```

```
GameStop Historical Stock Data (First 5 rows):
                                Open      High       Low     Close    Volume  \
Date
2002-02-13 00:00:00-05:00  1.620128  1.693350  1.603296  1.691666  76216000
2002-02-14 00:00:00-05:00  1.712707  1.716073  1.670626  1.683250  11021600
2002-02-15 00:00:00-05:00  1.683251  1.687459  1.658002  1.674834   8389600
2002-02-19 00:00:00-05:00  1.666418  1.666418  1.578047  1.607504   7410400
2002-02-20 00:00:00-05:00  1.615920  1.662210  1.603296  1.662210   6892800

                           Dividends  Stock Splits
Date
2002-02-13 00:00:00-05:00        0.0           0.0
2002-02-14 00:00:00-05:00        0.0           0.0
2002-02-15 00:00:00-05:00        0.0           0.0
2002-02-19 00:00:00-05:00        0.0           0.0
2002-02-20 00:00:00-05:00        0.0           0.0
```

[7]:
```python
# Question 3: Extract GameStop Stock Data using yfinance

# Import the library
import yfinance as yf

# Create ticker object for GameStop (GME)
gme = yf.Ticker("GME")

# Extract maximum historical data
gme_data = gme.history(period="max")

# Reset the index to make Date a column
gme_data.reset_index(inplace=True)

# Display the first 5 rows
print("GameStop Stock Data (First 5 rows):")
print(gme_data.head())
```

```
GameStop Stock Data (First 5 rows):
                       Date      Open      High       Low     Close    Volume  \
0 2002-02-13 00:00:00-05:00  1.620128  1.693350  1.603296  1.691667  76216000
1 2002-02-14 00:00:00-05:00  1.712707  1.716074  1.670626  1.683250  11021600
2 2002-02-15 00:00:00-05:00  1.683250  1.687458  1.658001  1.674834   8389600
3 2002-02-19 00:00:00-05:00  1.666418  1.666418  1.578047  1.607504   7410400
4 2002-02-20 00:00:00-05:00  1.615920  1.662209  1.603296  1.662209   6892800

   Dividends  Stock Splits
0        0.0           0.0
1        0.0           0.0
2        0.0           0.0
3        0.0           0.0
```

```
4        0.0            0.0
```

[11]:
```python
# Question 4: Webscraping GME Revenue Data with BeautifulSoup Parsers (Corrected)

import requests
from bs4 import BeautifulSoup
import pandas as pd

def extract_gme_revenue(soup):
    """Extract revenue data from BeautifulSoup object"""
    # Find all tables - revenue table is the second one
    tables = soup.find_all('table')
    if len(tables) < 2:
        print("Error: Could not find revenue table")
        return pd.DataFrame()

    # Create DataFrame
    gme_revenue = pd.DataFrame(columns=['Date', 'Revenue'])

    # Extract data from each row
    for row in tables[1].find('tbody').find_all('tr'):
        cols = row.find_all('td')
        if len(cols) == 2:
            date = cols[0].text.strip()
            revenue = cols[1].text.strip().replace('$', '').replace(',', '')
            if revenue:  # Only add if revenue exists
                gme_revenue = pd.concat([
                    gme_revenue,
                    pd.DataFrame({'Date': [date], 'Revenue': [revenue]})
                ], ignore_index=True)

    # Convert and clean data
    gme_revenue['Revenue'] = pd.to_numeric(gme_revenue['Revenue'], errors='coerce')
    gme_revenue = gme_revenue.dropna()

    return gme_revenue

# Step 1: Download the webpage
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"
response = requests.get(url)
html_data_2 = response.text

# Step 2: Parse using html.parser (built-in)
print("Using html.parser:")
```

```
soup_parser = BeautifulSoup(html_data_2, 'html.parser')
gme_revenue_parser = extract_gme_revenue(soup_parser)
print("Last 5 rows:")
print(gme_revenue_parser.tail())

# Step 3: Parse using html5lib (more lenient)
# First install if needed: pip install html5lib
print("\nUsing html5lib:")
soup_lib = BeautifulSoup(html_data_2, 'html5lib')
gme_revenue_lib = extract_gme_revenue(soup_lib)
print("Last 5 rows:")
print(gme_revenue_lib.tail())

# Final cleaned data (using html.parser)
gme_revenue = gme_revenue_parser
print("\nFinal GameStop Quarterly Revenue Data:")
print(gme_revenue.tail())
```

```
Using html.parser:
Last 5 rows:
          Date  Revenue
57  2006-01-31     1667
58  2005-10-31      534
59  2005-07-31      416
60  2005-04-30      475
61  2005-01-31      709

Using html5lib:
Last 5 rows:
          Date  Revenue
57  2006-01-31     1667
58  2005-10-31      534
59  2005-07-31      416
60  2005-04-30      475
61  2005-01-31      709

Final GameStop Quarterly Revenue Data:
          Date  Revenue
57  2006-01-31     1667
58  2005-10-31      534
59  2005-07-31      416
60  2005-04-30      475
61  2005-01-31      709
```

[12]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd
```

```
# Download the webpage
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
 ↪IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"
html_data = requests.get(url).text

# Parse with BeautifulSoup
soup = BeautifulSoup(html_data, 'html.parser')

# Create empty DataFrame
gme_revenue = pd.DataFrame(columns=['Date', 'Revenue'])

# Find the revenue table (second table)
table = soup.find_all('table')[1]

# Extract data from each row
for row in table.find('tbody').find_all('tr'):
    cols = row.find_all('td')
    if len(cols) == 2:  # Ensure we have both Date and Revenue
        date = cols[0].text.strip()
        revenue = cols[1].text.strip().replace('$', '').replace(',', '')
        # Add to DataFrame using concat (modern alternative to append)
        gme_revenue = pd.concat([
            gme_revenue,
            pd.DataFrame({'Date': [date], 'Revenue': [revenue]})
        ], ignore_index=True)

# Convert Revenue to numeric
gme_revenue['Revenue'] = pd.to_numeric(gme_revenue['Revenue'])

print("GameStop Revenue (BeautifulSoup method):")
print(gme_revenue.tail())
```

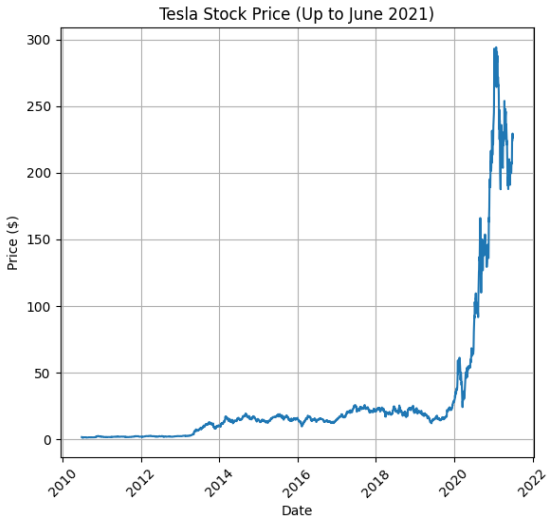
```
GameStop Revenue (BeautifulSoup method):
          Date  Revenue
57  2006-01-31     1667
58  2005-10-31      534
59  2005-07-31      416
60  2005-04-30      475
61  2005-01-31      709
```

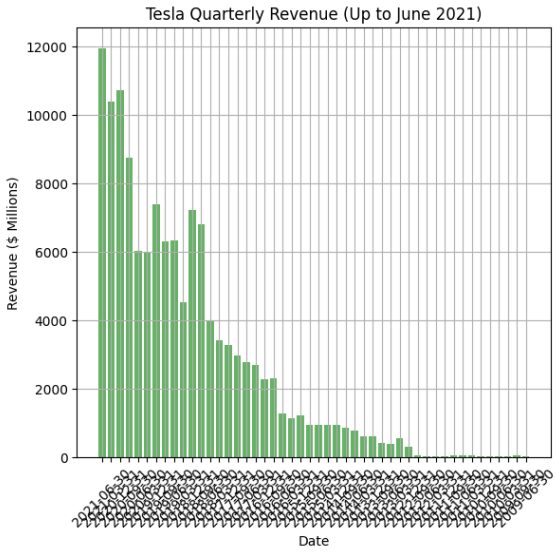
[13]:
```
# Display the last 5 rows of GameStop revenue data
print("Last 5 rows of GameStop Quarterly Revenue:")
print(gme_revenue.tail())
```

```
Last 5 rows of GameStop Quarterly Revenue:
          Date  Revenue
57  2006-01-31     1667
```

```
58  2005-10-31       534
59  2005-07-31       416
60  2005-04-30       475
61  2005-01-31       709
```

[14]:
```python
# Question 5: Plot Tesla Stock Graph

import matplotlib.pyplot as plt

def make_graph(stock_data, revenue_data, company_name):
    """
    Plots stock price and revenue data with two y-axes
    """
    plt.figure(figsize=(12, 6))

    # Filter data up to June 2021
    stock_data = stock_data[stock_data['Date'] <= '2021-06-30']
    revenue_data = revenue_data[revenue_data['Date'] <= '2021-06-30']

    # Plot stock price
    plt.subplot(1, 2, 1)
    plt.plot(stock_data['Date'], stock_data['Close'], label='Closing Price')
    plt.title(f'{company_name} Stock Price (Up to June 2021)')
    plt.xlabel('Date')
    plt.ylabel('Price ($)')
    plt.xticks(rotation=45)
    plt.grid(True)

    # Plot revenue
    plt.subplot(1, 2, 2)
    plt.bar(revenue_data['Date'], revenue_data['Revenue'], color='green',␣
 ↪alpha=0.6)
    plt.title(f'{company_name} Quarterly Revenue (Up to June 2021)')
    plt.xlabel('Date')
    plt.ylabel('Revenue ($ Millions)')
    plt.xticks(rotation=45)
    plt.grid(True)

    plt.tight_layout()
    plt.show()

# Call the function with Tesla data
make_graph(tesla_data, tesla_revenue, 'Tesla')
```





[15]:
```python
# Question 6: Plot GameStop Stock Graph

def make_graph(stock_data, revenue_data, company_name):
    """
    Plots stock price and revenue data with two y-axes
    """
    plt.figure(figsize=(14, 7))

    # Filter data up to June 2021
    stock_data = stock_data[stock_data['Date'] <= '2021-06-30']
    revenue_data = revenue_data[revenue_data['Date'] <= '2021-06-30']

    # Create figure and primary axis
    fig, ax1 = plt.subplots(figsize=(12, 6))

    # Plot stock price (red line)
    color = 'tab:red'
    ax1.set_xlabel('Date (Up to June 2021)')
    ax1.set_ylabel('Stock Price ($)', color=color)
    ax1.plot(stock_data['Date'], stock_data['Close'], color=color, linewidth=2)
    ax1.tick_params(axis='y', labelcolor=color)
    ax1.grid(True, linestyle='--', alpha=0.7)

    # Create secondary axis for revenue (blue bars)
    ax2 = ax1.twinx()
    color = 'tab:blue'
    ax2.set_ylabel('Revenue ($ Millions)', color=color)
```

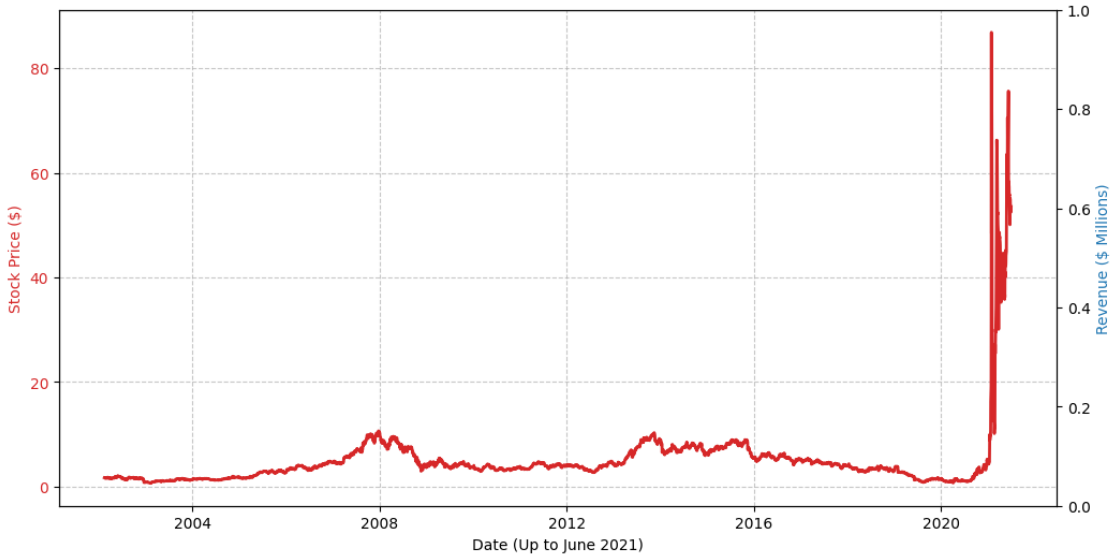
```
    ax2.bar(revenue_data['Date'], revenue_data['Revenue'], color=color, alpha=0.
 ↪6, width=20)
    ax2.tick_params(axis='y', labelcolor=color)

    # Formatting
    plt.title(f'{company_name} Stock Price and Quarterly Revenue', pad=20,␣
 ↪fontsize=14, fontweight='bold')
    fig.autofmt_xdate(rotation=45)
    plt.tight_layout()
    plt.show()

# Call the function with GameStop data
make_graph(gme_data, gme_revenue, 'GameStop')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
File /opt/conda/lib/python3.12/site-packages/matplotlib/axis.py:1811, in Axis.
 ↪convert_units(self, x)
   1810 try:
-> 1811     ret = self._converter.convert(x, self.units, self)
   1812 except Exception as e:

File /opt/conda/lib/python3.12/site-packages/matplotlib/category.py:53, in␣
 ↪StrCategoryConverter.convert(value, unit, axis)
     49     raise ValueError(
     50         'Missing category information for StrCategoryConverter; '
     51         'this might be caused by unintendedly mixing categorical and '
     52         'numeric data')
---> 53 StrCategoryConverter._validate_unit(unit)
     54 # dtype = object preserves numerical pass throughs

File /opt/conda/lib/python3.12/site-packages/matplotlib/category.py:114, in␣
 ↪StrCategoryConverter._validate_unit(unit)
    113 if not hasattr(unit, '_mapping'):
--> 114     raise ValueError(
    115         f'Provided unit "{unit}" is not valid for a categorical '
    116         'converter, as it does not have a _mapping attribute.')

ValueError: Provided unit "America/New_York" is not valid for a categorical␣
 ↪converter, as it does not have a _mapping attribute.

The above exception was the direct cause of the following exception:

ConversionError                           Traceback (most recent call last)
Cell In[15], line 38
     35     plt.show()
     37 # Call the function with GameStop data
```

```
---> 38 make_graph(gme_data, gme_revenue, 'GameStop')

Cell In[15], line 28, in make_graph(stock_data, revenue_data, company_name)
     26 color = 'tab:blue'
     27 ax2.set_ylabel('Revenue ($ Millions)', color=color)
---> 28␣
 ↪ax2.bar(revenue_data['Date'], revenue_data['Revenue'], color=color, alpha=0.6  width=20)
     29 ax2.tick_params(axis='y', labelcolor=color)
     31 # Formatting

File /opt/conda/lib/python3.12/site-packages/matplotlib/__init__.py:1521, in␣
 ↪_preprocess_data.<locals>.inner(ax, data, *args, **kwargs)
   1518 @functools.wraps(func)
   1519 def inner(ax, *args, data=None, **kwargs):
   1520     if data is None:
-> 1521         return func(
   1522             ax,
   1523             *map(cbook.sanitize_sequence, args),
   1524             **{k: cbook.sanitize_sequence(v) for k, v in kwargs.items()} )
   1526     bound = new_sig.bind(ax, *args, **kwargs)
   1527     auto_label = (bound.arguments.get(label_namer)
   1528                   or bound.kwargs.get(label_namer))

File /opt/conda/lib/python3.12/site-packages/matplotlib/axes/_axes.py:2572, in␣
 ↪Axes.bar(self, x, height, width, bottom, align, **kwargs)
   2570 if self.xaxis is not None:
   2571     x0 = x
-> 2572     x = np.asarray(self.convert_xunits(x))
   2573     width = self._convert_dx(width, x0, x, self.convert_xunits)
   2574     if xerr is not None:

File /opt/conda/lib/python3.12/site-packages/matplotlib/artist.py:278, in Artis
 ↪convert_xunits(self, x)
    276 if ax is None or ax.xaxis is None:
    277     return x
--> 278 return ax.xaxis.convert_units(x)

File /opt/conda/lib/python3.12/site-packages/matplotlib/axis.py:1813, in Axis.
 ↪convert_units(self, x)
   1811     ret = self._converter.convert(x, self.units, self)
   1812 except Exception as e:
-> 1813     raise munits.ConversionError('Failed to convert value(s) to axis '
   1814                                  f'units: {x!r}') from e
   1815 return ret

ConversionError: Failed to convert value(s) to axis units: 0     2020-04-30
1     2020-01-31
2     2019-10-31
```

```
3     2019-07-31
4     2019-04-30
         …
57    2006-01-31
58    2005-10-31
59    2005-07-31
60    2005-04-30
61    2005-01-31
Name: Date, Length: 62, dtype: object
```

```
<Figure size 1400x700 with 0 Axes>
```



[ ]: