10. Introduction to Data Visualization with Matplotlib

Chapter 1 - Introduction to Matplotlib

This chapter introduces the Matplotlib visualization library and demonstrates how to use it with data.

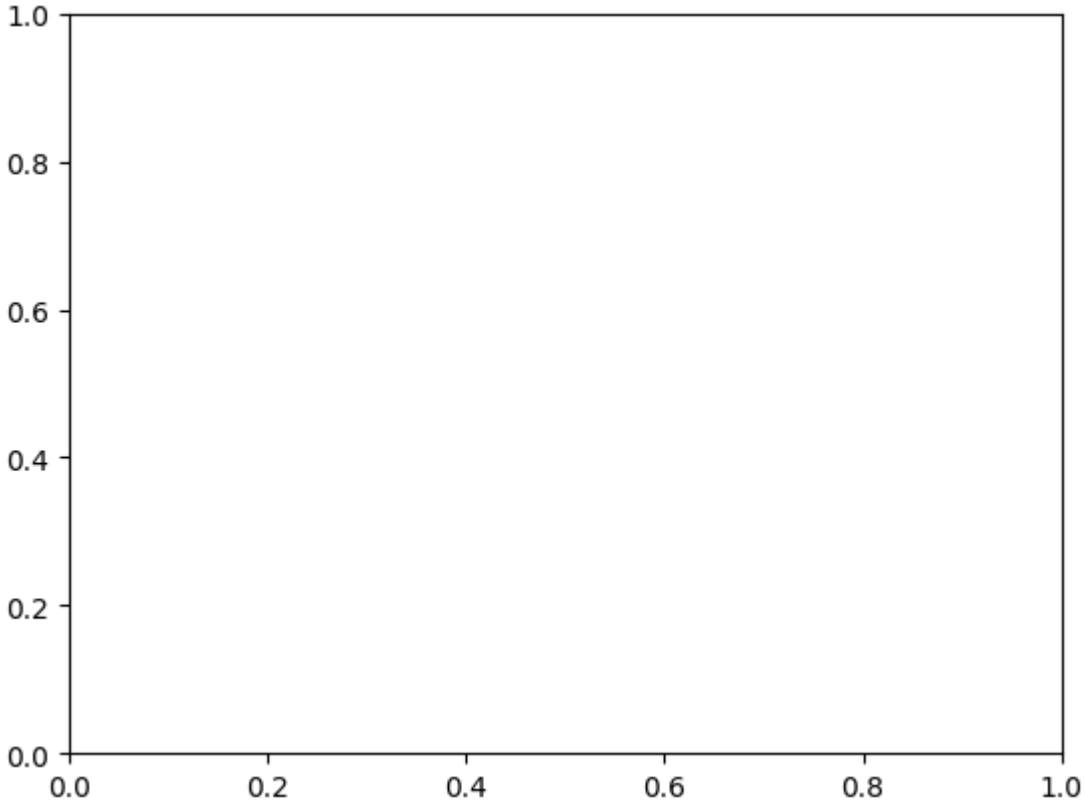
Using the matplotlib.pyplot interface

There are many ways to use Matplotlib. In this course, we will focus on the pyplot interface, which provides the most flexibility in creating and customizing data visualizations.

Initially, we will use the pyplot interface to create two kinds of objects: Figure objects and Axes objects.

This course introduces a lot of new concepts, so if you ever need a quick refresher, download the Matplotlib Cheat Sheet and keep it handy!

In [ ]:
```
# Import the matplotlib.pyplot submodule and name it plt
import matplotlib.pyplot as plt

# Create a Figure and an Axes with plt.subplots
fig, ax = plt.subplots()

# Call the show function to show the result
plt.show()
```

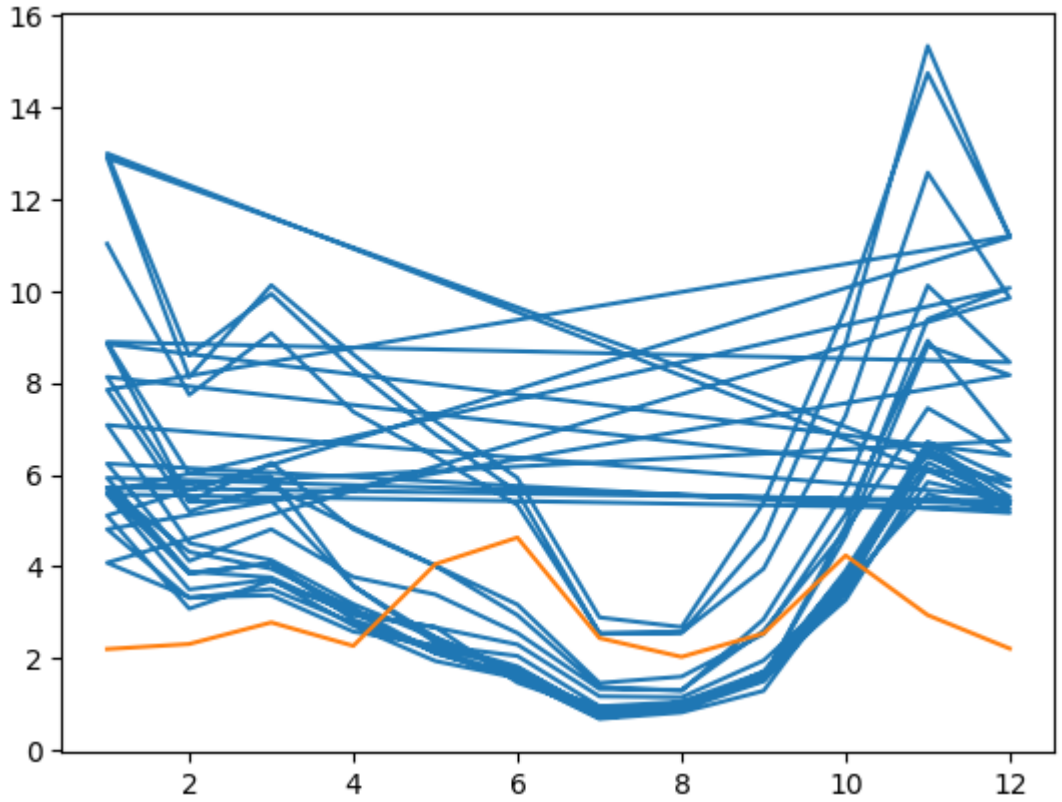


Adding data to an Axes object

In [ ]:

```
import pandas as pd

# Specify the file paths using double backslashes
austin_weather = pd.read_csv('C:\\Users\\yeiso\\OneDrive - Douglas College\\0. DOUGLAS COLLEGE\\3. Fund Machine Learning\\0. Python Course

seattle_weather = pd.read_csv('C:\\Users\\yeiso\\OneDrive - Douglas College\\0. DOUGLAS COLLEGE\\3. Fund Machine Learning\\0. Python Course

# Import the matplotlib.pyplot submodule and name it plt
import matplotlib.pyplot as plt

# Create a Figure and an Axes with plt.subplots
fig, ax = plt.subplots()

# Plot MLY-PRCP-NORMAL from seattle_weather against the MONTH
ax.plot(seattle_weather["DATE"], seattle_weather["MLY-PRCP-NORMAL"])

# Plot MLY-PRCP-NORMAL from austin_weather against MONTH
```

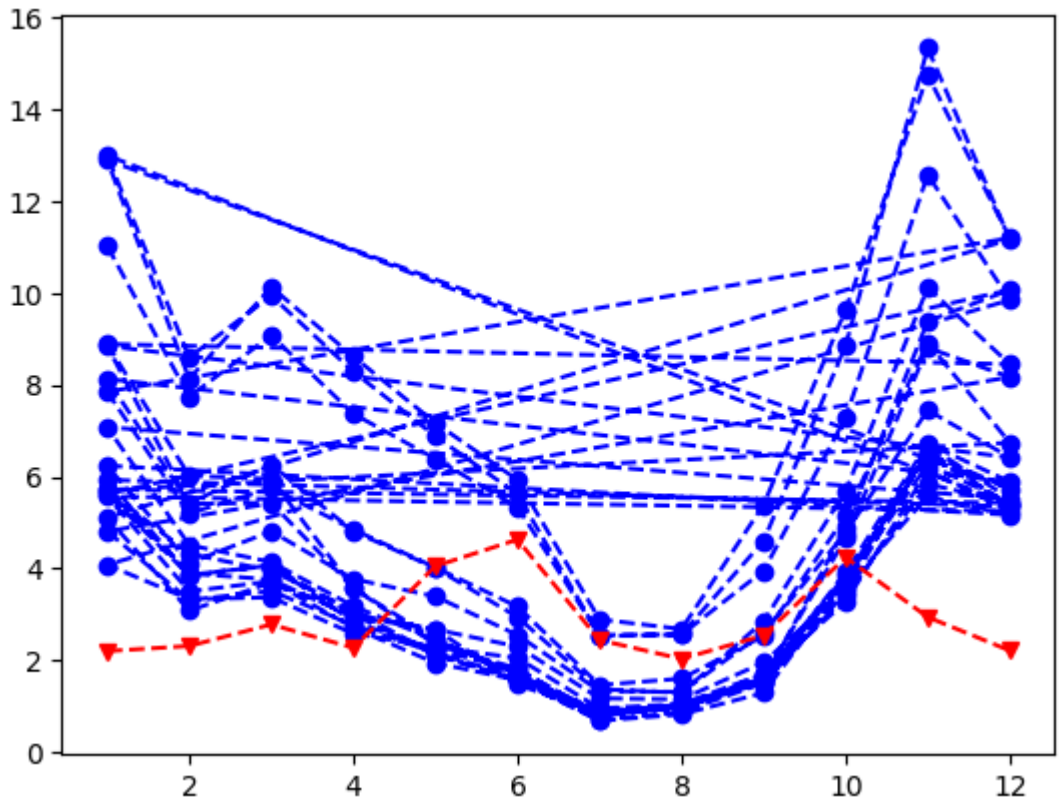
```
ax.plot(austin_weather["DATE"], austin_weather["MLY-PRCP-NORMAL"])

# Call the show function
plt.show()
```



Customizing data appearance

In [ ]:
```
import pandas as pd

# Specify the file paths using double backslashes
austin_weather = pd.read_csv('C:\\Users\\yeiso\\OneDrive - Douglas College\\0. DOUGLAS COLLEGE\\3. Fund Machine Learning\\0. Python Course

seattle_weather = pd.read_csv('C:\\Users\\yeiso\\OneDrive - Douglas College\\0. DOUGLAS COLLEGE\\3. Fund Machine Learning\\0. Python Course

# Import the matplotlib.pyplot submodule and name it plt
import matplotlib.pyplot as plt

# Create a Figure and an Axes with plt.subplots
fig, ax = plt.subplots()
```

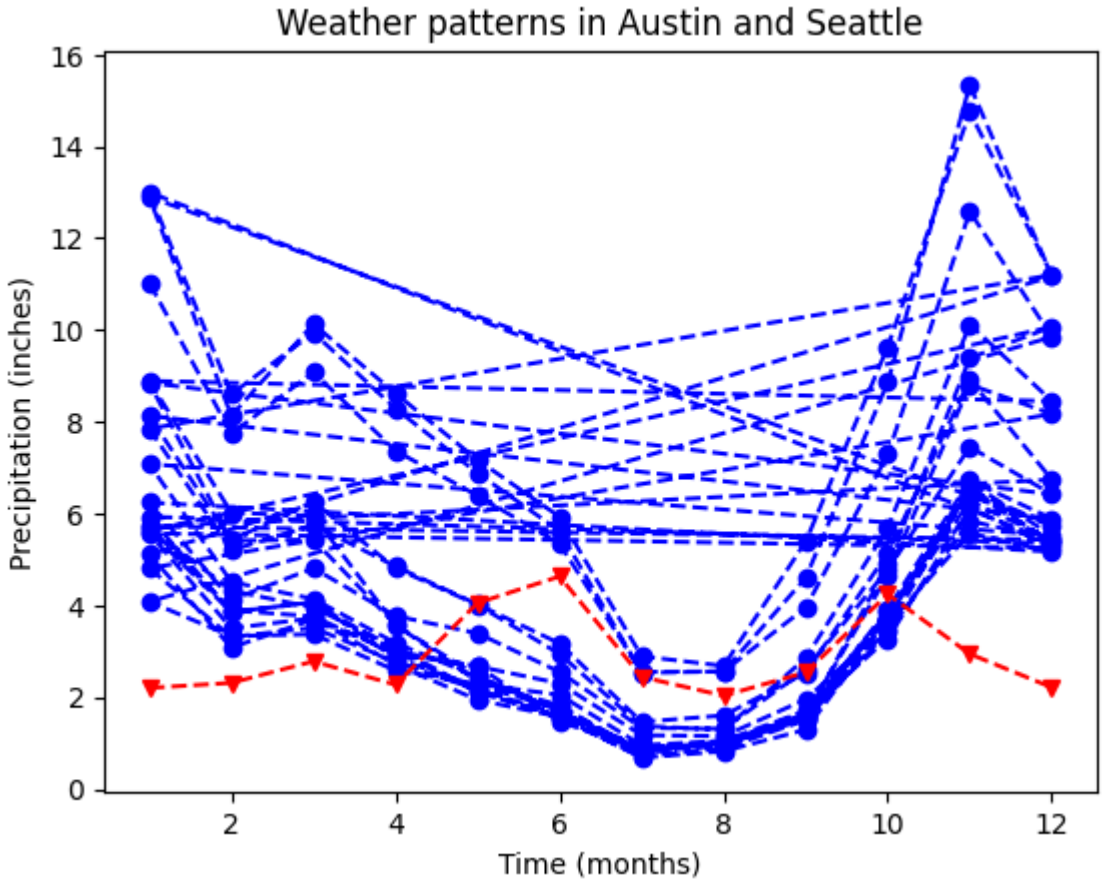
```
#-------------------------------------------------------------------------
#-------------------------------------------------------------------------
#second part of the code
# Plot Seattle data, setting data appearance
ax.plot(seattle_weather["DATE"], seattle_weather["MLY-PRCP-NORMAL"], color='b', marker='o', linestyle='--')

# Plot Austin data, setting data appearance
ax.plot(austin_weather["DATE"], austin_weather["MLY-PRCP-NORMAL"], color='r', marker='v', linestyle='--')

# Call show to display the resulting plot
plt.show();
```



Customizing axis labels and adding titles

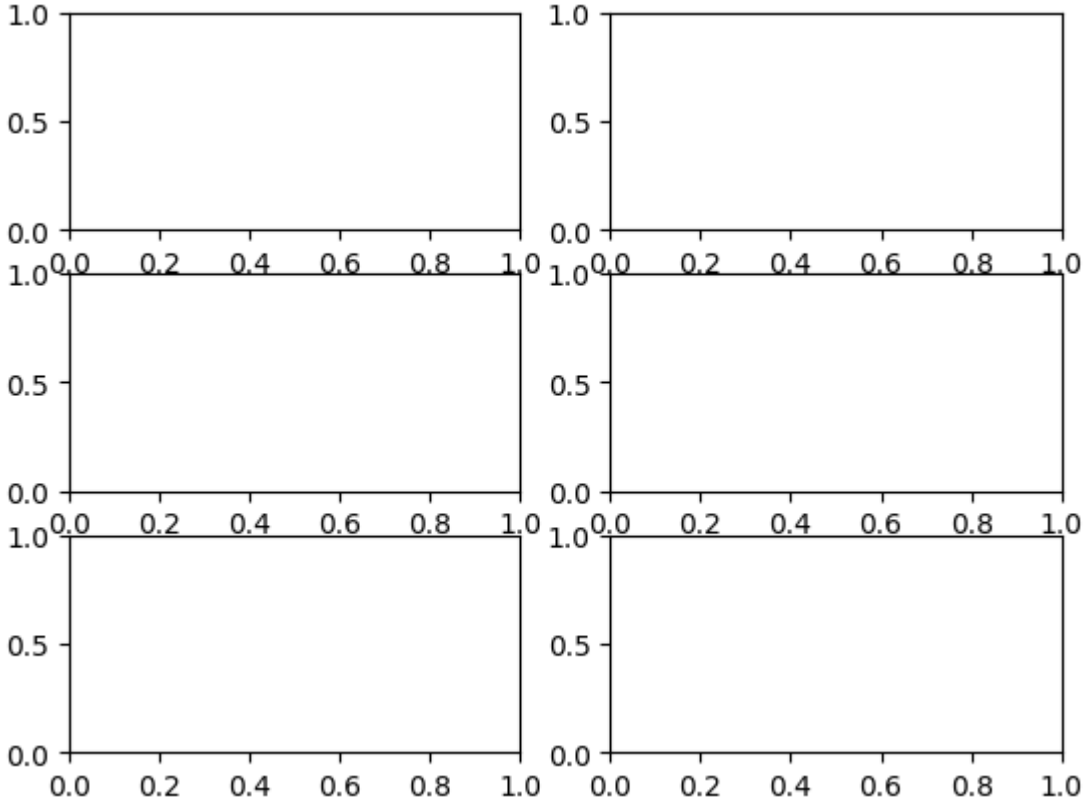
In [ ]:
```
import pandas as pd

# Specify the file paths using double backslashes
austin_weather = pd.read_csv('C:\\Users\\yeiso\\OneDrive - Douglas College\\0. DOUGLAS COLLEGE\\3. Fund Machine Learning\\0. Python Course
```

```
seattle_weather = pd.read_csv('C:\\Users\\yeiso\\OneDrive - Douglas College\\0. DOUGLAS COLLEGE\\3. Fund Machine Learning\\0. Python Course

# Import the matplotlib.pyplot submodule and name it plt
import matplotlib.pyplot as plt

# Create a Figure and an Axes with plt.subplots
fig, ax = plt.subplots()

#------------------------------------------------------------------------
#------------------------------------------------------------------------
#second part of the code
# Plot Seattle data, setting data appearance
ax.plot(seattle_weather["DATE"], seattle_weather["MLY-PRCP-NORMAL"], color='b', marker='o', linestyle='--')

# Plot Austin data, setting data appearance
ax.plot(austin_weather["DATE"], austin_weather["MLY-PRCP-NORMAL"], color='r', marker='v', linestyle='--')

# Customize the x-axis label
ax.set_xlabel('Time (months)')

# Customize the y-axis label
ax.set_ylabel('Precipitation (inches)')

# Add the title
ax.set_title('Weather patterns in Austin and Seattle')

# Display the figure
plt.show()
```



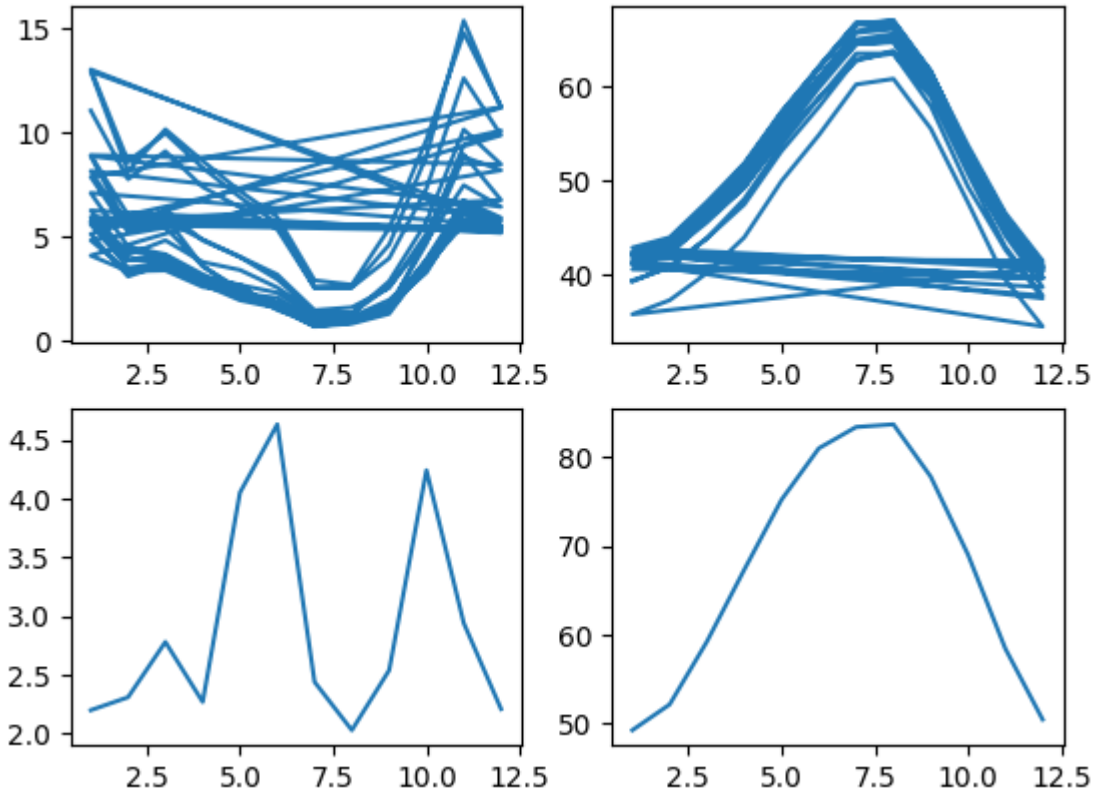
Creating a grid of subplots

In [ ]:
```
fig, ax = plt.subplots(3, 2)
```



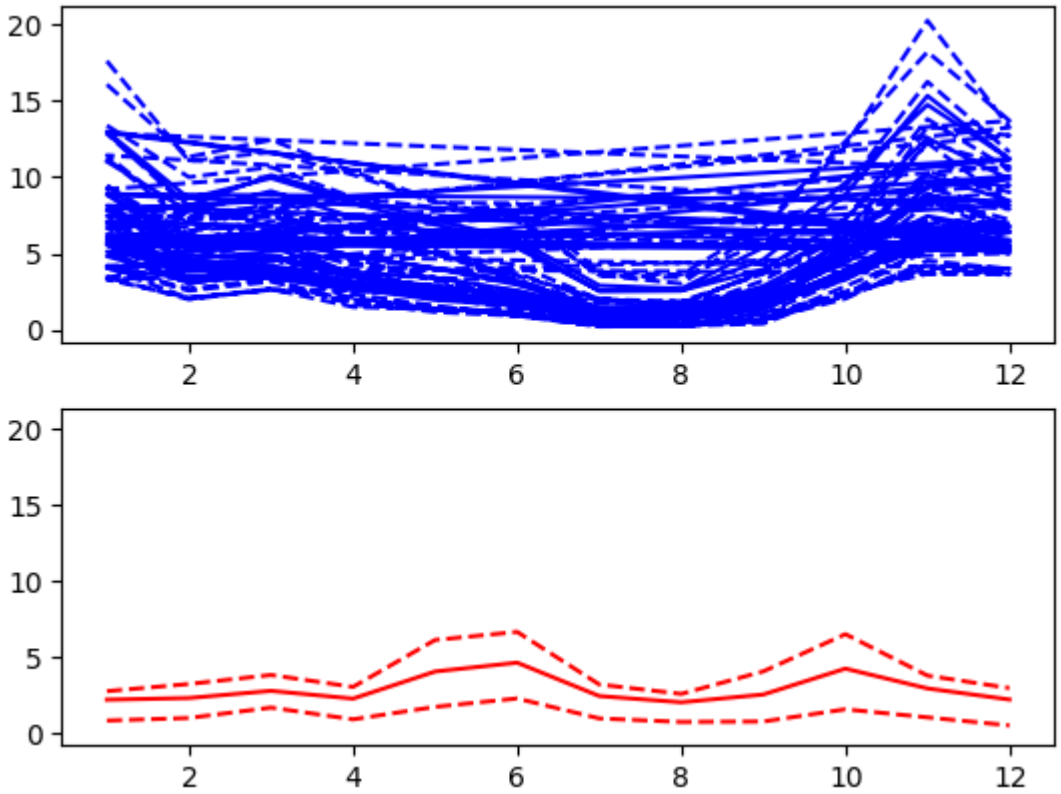
Creating small multiples with plt.subplots

In [ ]:
```
# Create a Figure and an array of subplots with 2 rows and 2 columns
fig, ax = plt.subplots(2, 2)

# Addressing the top left Axes as index 0, 0, plot month and Seattle precipitation
ax[0, 0].plot(seattle_weather['DATE'], seattle_weather['MLY-PRCP-NORMAL'])

# In the top right (index 0,1), plot month and Seattle temperatures
ax[0, 1].plot(seattle_weather['DATE'], seattle_weather['MLY-TAVG-NORMAL'])

# In the bottom left (1, 0) plot month and Austin precipitations
ax[1, 0].plot(austin_weather['DATE'], austin_weather['MLY-PRCP-NORMAL'])

# In the bottom right (1, 1) plot month and Austin temperatures
ax[1, 1].plot(austin_weather['DATE'], austin_weather['MLY-TAVG-NORMAL'])

plt.show()
```



Small multiples with shared y axis

In [ ]:
```
# Create a figure and an array of axes: 2 rows, 1 column with shared y axis
fig, ax = plt.subplots(2, 1, sharey=True)

# Plot Seattle precipitation data in the top axes
ax[0].plot(seattle_weather['DATE'], seattle_weather['MLY-PRCP-NORMAL'], color = 'b')
ax[0].plot(seattle_weather['DATE'], seattle_weather['MLY-PRCP-25PCTL'], color = 'b', linestyle = '--')
ax[0].plot(seattle_weather['DATE'], seattle_weather['MLY-PRCP-75PCTL'], color = 'b', linestyle = '--')

# Plot Austin precipitation data in the bottom axes
ax[1].plot(austin_weather['DATE'], austin_weather['MLY-PRCP-NORMAL'], color = 'r')
ax[1].plot(austin_weather['DATE'], austin_weather['MLY-PRCP-25PCTL'], color = 'r', linestyle = '--')
ax[1].plot(austin_weather['DATE'], austin_weather['MLY-PRCP-75PCTL'], color = 'r', linestyle = '--')

plt.show()
```



Chapter 2 -Plotting time-series

Time series data is data that is recorded. Visualizing this type of data helps clarify trends and illuminates relationships between data.

Read data with a time index

In [ ]:
```
# Import pandas as pd
import pandas as pd

# Read the data from file using read_csv
climate_change = pd.read_csv('C:\\Users\\yeiso\\OneDrive - Douglas College\\0. DOUGLAS COLLEGE\\3. Fund Machine Learning\\0. Python Course
climate_change.head()
```

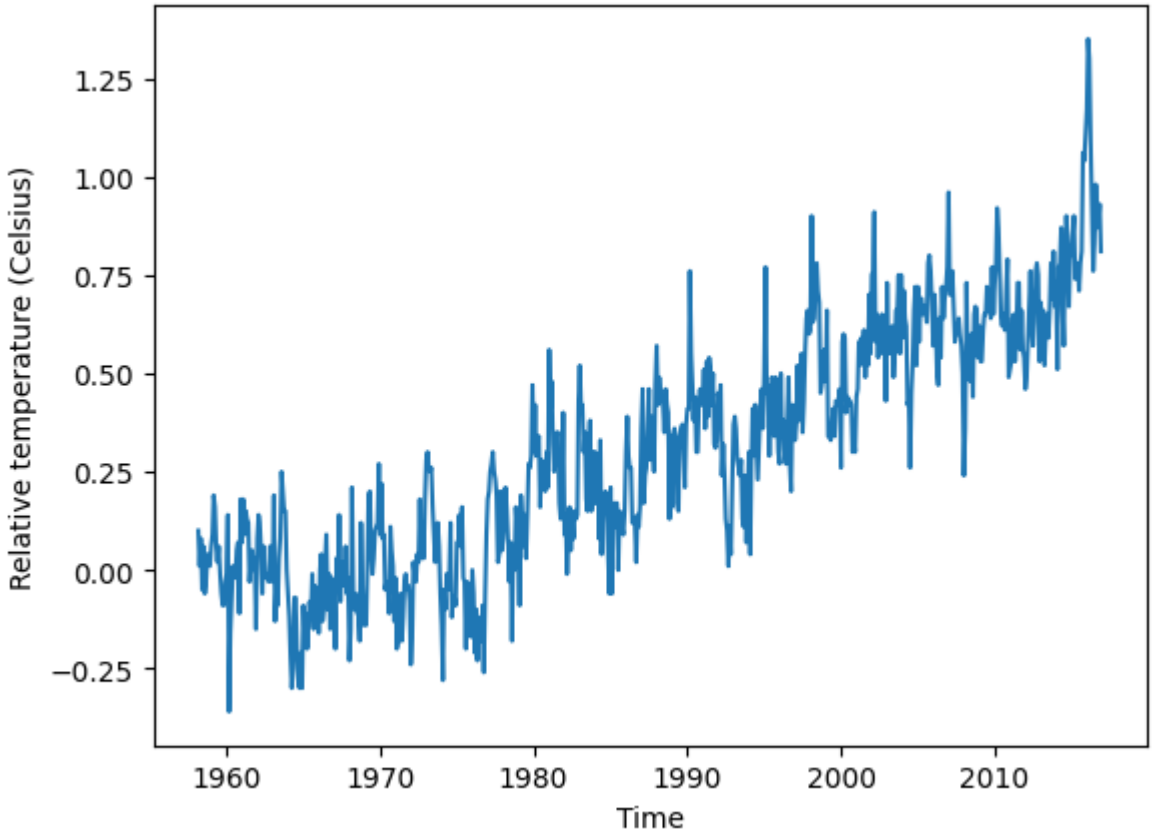
Out[ ]:

date	co2	relative_temp
1958-03-06	315.71	0.10
1958-04-06	317.45	0.01
1958-05-06	317.50	0.08
1958-06-06	NaN	-0.05
1958-07-06	315.86	0.06

Plot time-series data

In [ ]:

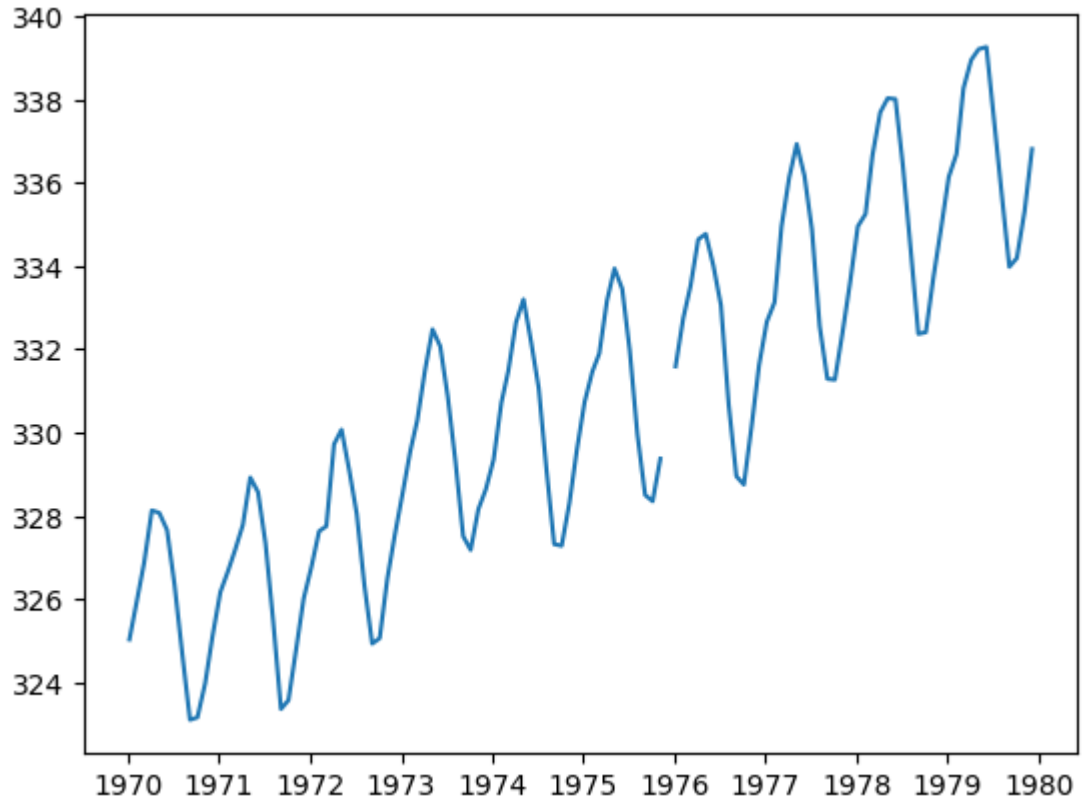
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()

# Add the time-series for "relative_temp" to the plot
ax.plot(climate_change.index, climate_change['relative_temp'])

# Set the x-axis label
ax.set_xlabel('Time')

# Set the y-axis label
ax.set_ylabel('Relative temperature (Celsius)')

# Show the figure
plt.show()
```



Using a time index to zoom in

In [ ]:

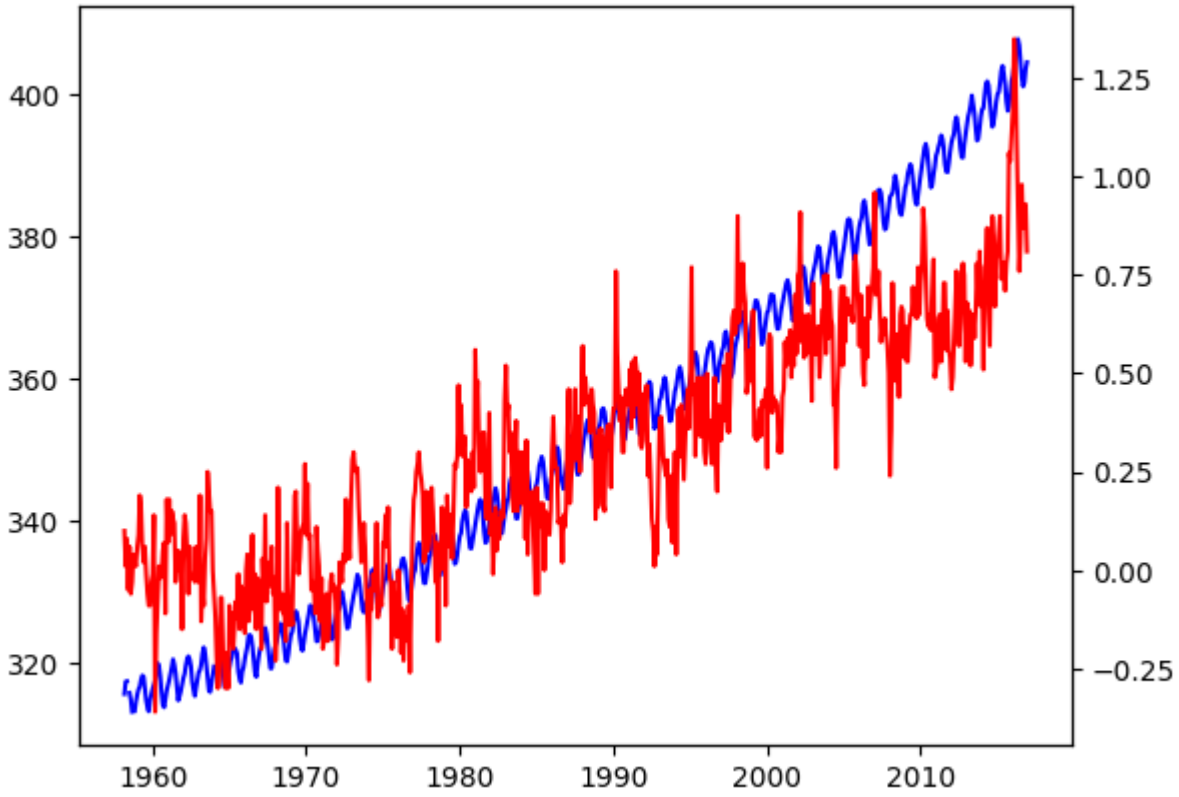
```
import matplotlib.pyplot as plt

# Use plt.subplots to create fig and ax
fig, ax = plt.subplots()

# Create variable seventies with data from "1970-01-01" to "1979-12-31"
seventies = climate_change["1970-01-01":"1979-12-31"]

# Add the time-series for "co2" data from seventies to the plot
ax.plot(seventies.index, seventies["co2"])

# Show the figure
plt.show()
```



Plotting two variables

In [ ]:

```
import matplotlib.pyplot as plt

# Initalize a Figure and Axes
fig, ax = plt.subplots()

# Plot the CO2 variable in blue
ax.plot(climate_change.index, climate_change['co2'], color='b')

# Create a twin Axes that shares the x-axis
ax2 = ax.twinx()

# Plot the relative temperature in red
ax2.plot(climate_change.index, climate_change['relative_temp'], color='r')

plt.show()
```

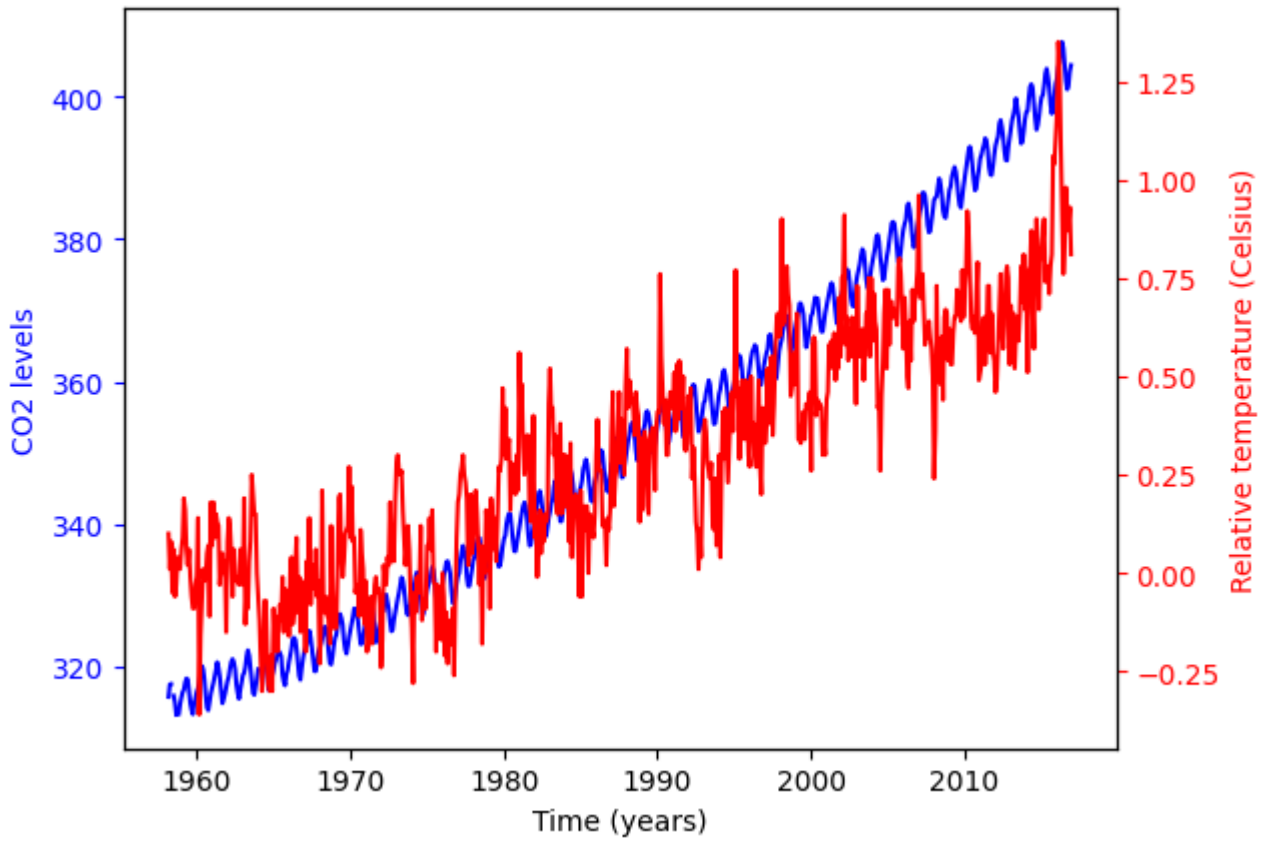


Defining a function that plots time-series data

In [ ]:
```
# Define a function called plot_timeseries
def plot_timeseries(axes, x, y, color, xlabel, ylabel):

  # Plot the inputs x,y in the provided color
  axes.plot(x, y, color=color)

  # Set the x-axis label
  axes.set_xlabel(xlabel)

  # Set the y-axis label
  axes.set_ylabel(ylabel, color=color)

  # Set the colors tick params for y-axis
  axes.tick_params('y', colors=color)
```

Using a plotting function

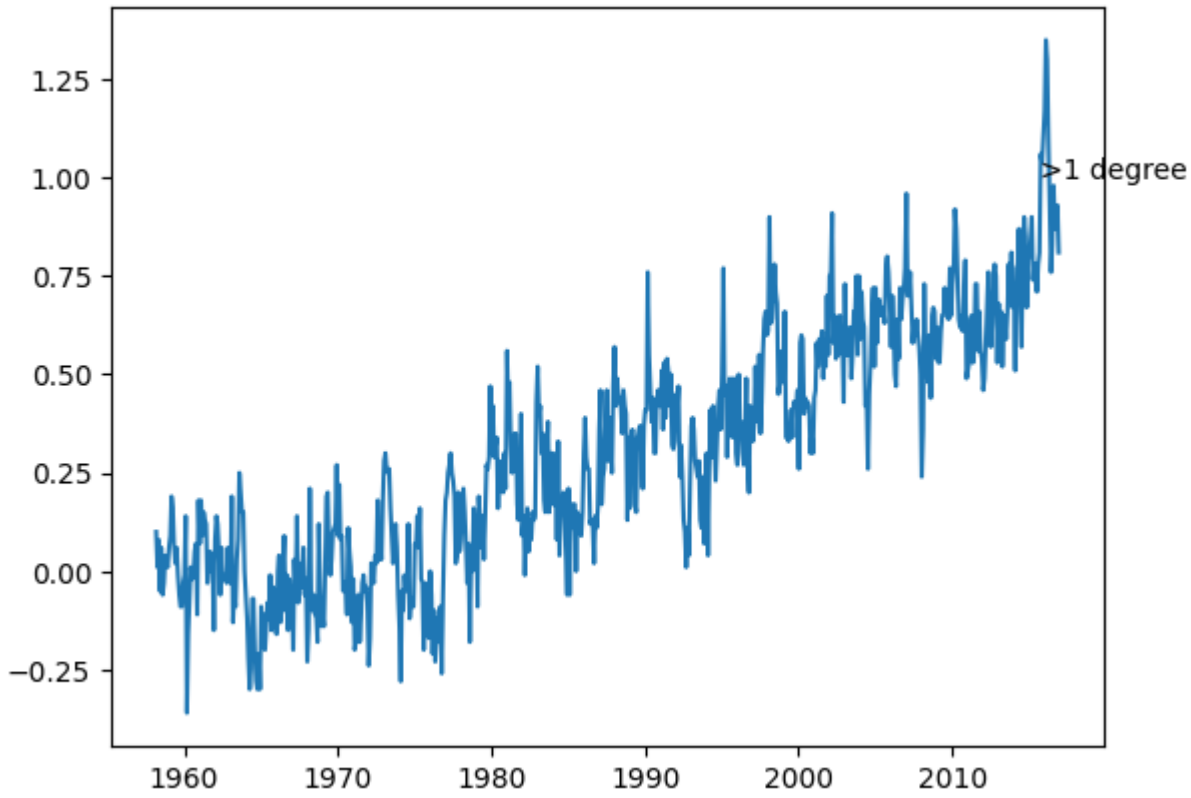
In [ ]:
```
fig, ax = plt.subplots()

# Plot the CO2 levels time-series in blue
plot_timeseries(ax, climate_change.index, climate_change['co2'], "blue", "Time (years)", "CO2 levels")

# Create a twin Axes object that shares the x-axis
ax2 = ax.twinx()

# Plot the relative temperature data in red
plot_timeseries(ax2, climate_change.index, climate_change['relative_temp'], "red", "Time (years)", "Relative temperature (Celsius)")

plt.show()
```



Annotating a plot of time-series data

In [ ]:
```
fig, ax = plt.subplots()
```

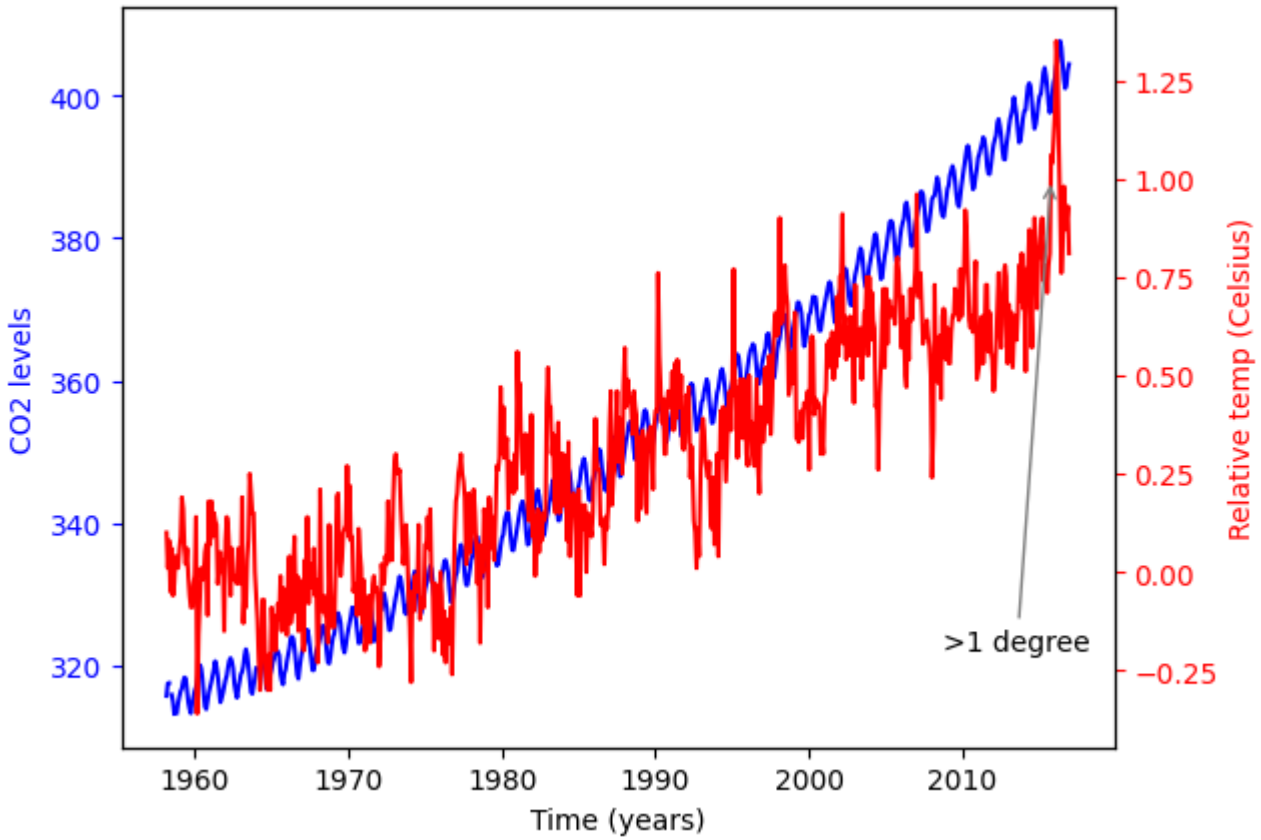
```
# Plot the relative temperature data
ax.plot(climate_change.index, climate_change.relative_temp)

# Annotate the date at which temperatures exceeded 1 degree
ax.annotate('>1 degree', xy=(pd.Timestamp('2015-10-06'), 1))

plt.show()
```



Plotting time-series: putting it all together

In [ ]:
```
fig, ax = plt.subplots()

# Plot the CO2 levels time-series in blue
plot_timeseries(ax, climate_change.index, climate_change.co2, 'blue', 'Time (years)', 'CO2 levels')

# Create an Axes object that shares the x-axis
ax2 = ax.twinx()

# Plot the relative temperature data in red
```

```
plot_timeseries(ax2, climate_change.index, climate_change.relative_temp, 'red', 'Time (years)', 'Relative temp (Celsius)')

# Annotate point with relative temperature >1 degree
ax2.annotate(">1 degree", xy=(pd.Timestamp('2015-10-06'),1), xytext=(pd.Timestamp('2008-10-06'),-0.2), arrowprops={'arrowstyle': '->', 'col

plt.show()
```



Chapter 3 - Quantitative comparisons and statistical visualizations

Visualizations can be used to compare data in a quantitative manner. This chapter explains several methods for quantitative visualizations.

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt



# Specify the file path using double backslashes
```

```
medals = pd.read_csv('C:\\Users\\yeiso\\OneDrive - Douglas College\\0. DOUGLAS COLLEGE\\3. Fund Machine Learning\\0. Python Course DataCamp

# Rename the 'Unnamed: 0' column to 'Country'
medals.rename(columns={'Unnamed: 0': 'Country'}, inplace=True)

# Display the first few rows of the DataFrame
medals.head()
```

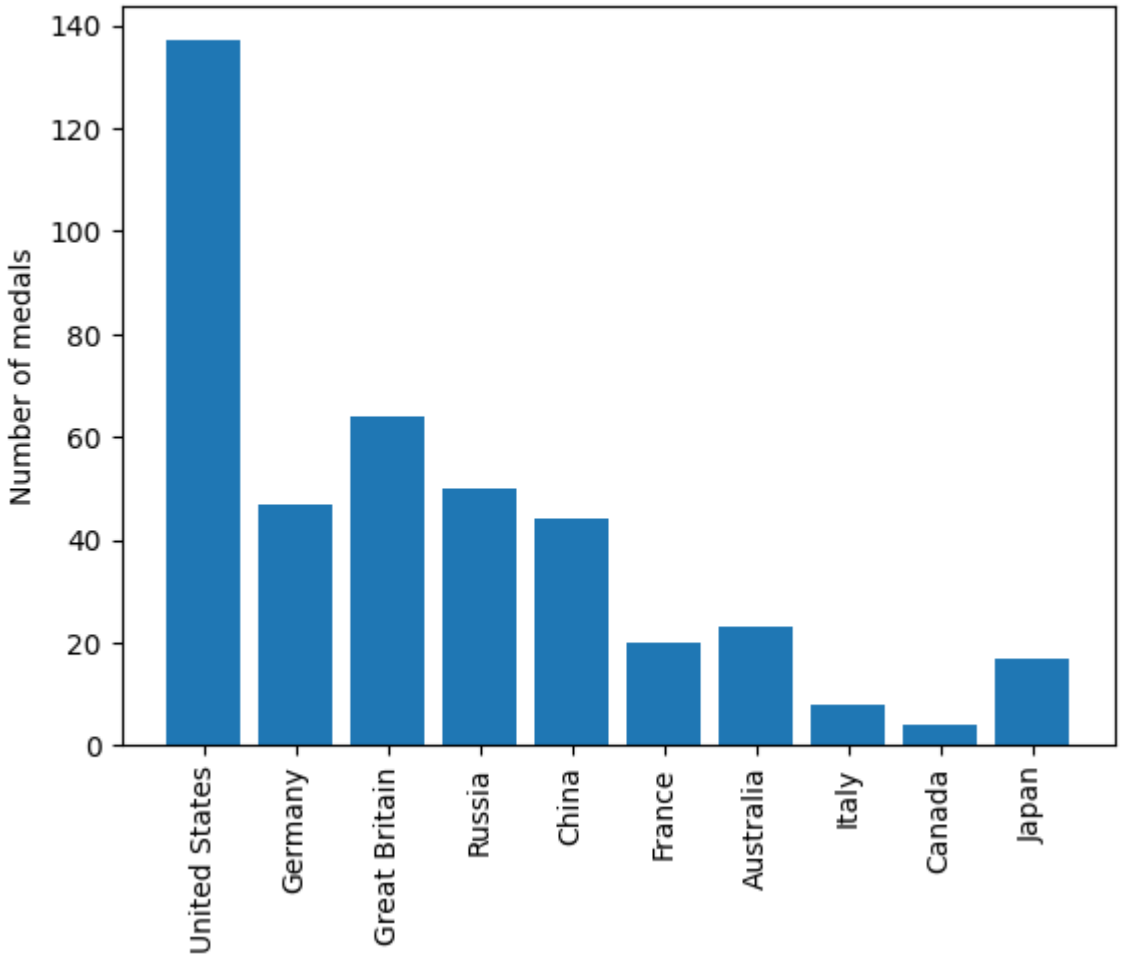
Out[ ]:

	Country	Bronze	Gold	Silver
0	United States	67	137	52
1	Germany	67	47	43
2	Great Britain	26	64	55
3	Russia	35	50	28
4	China	35	44	30

Bar chart

In [ ]:
```
fig, ax = plt.subplots()


# Set 'Country' as the index
medals.set_index('Country', inplace=True)


# Plot a bar-chart of gold medals as a function of country
ax.bar(medals.index, medals.Gold)

# Set the x-axis tick labels to the country names
ax.set_xticklabels(medals.index, rotation=90)

# Set the y-axis label
ax.set_ylabel('Number of medals')

plt.show()
```

```
C:\Users\yeiso\AppData\Local\Temp\ipykernel_18080\130371325.py:12: UserWarning: set_ticklabels() should only be used with a fixed number of ticks, i.e. after set_ticks() or using a FixedLocator.
  ax.set_xticklabels(medals.index, rotation=90)
```

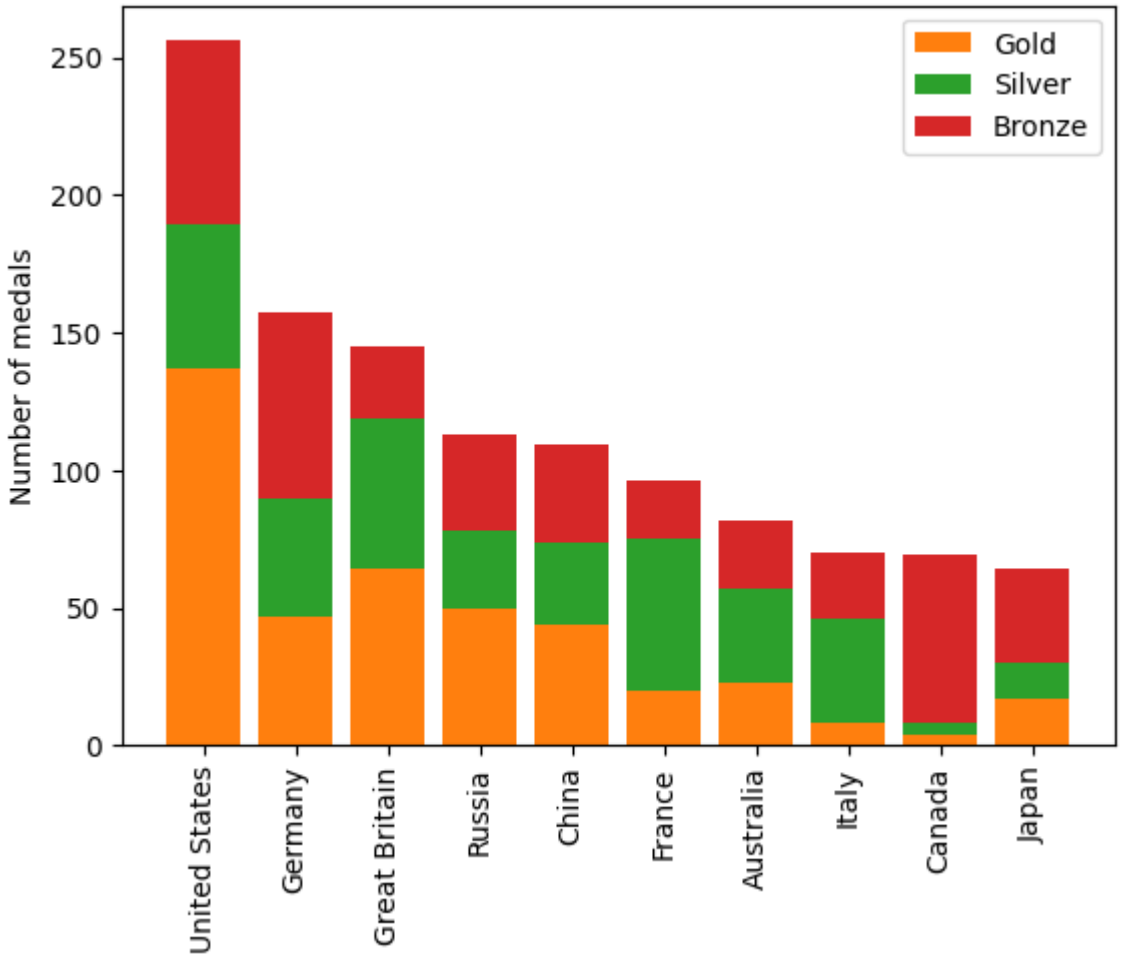


Stacked bar chart

In [ ]:
```
fig, ax = plt.subplots()

# Plot a bar-chart of gold medals as a function of country
ax.bar(medals.index, medals.Gold)

# Set the x-axis tick labels to the country names
ax.set_xticklabels(medals.index, rotation=90)

# Set the y-axis label
ax.set_ylabel('Number of medals')
```

```python
#second part of the code
# Add bars for "Gold" with the label "Gold"
ax.bar(medals.index, medals.Gold, label='Gold')

# Stack bars for "Silver" on top with label "Silver"
ax.bar(medals.index, medals.Silver, bottom=medals.Gold, label='Silver')

# Stack bars for "Bronze" on top of that with label "Bronze"
ax.bar(medals.index, medals.Bronze, bottom=medals.Gold + medals.Silver, label='Bronze')

# Display the legend
ax.legend()

plt.show()
```

```
C:\Users\yeiso\AppData\Local\Temp\ipykernel_18080\1355922929.py:7: UserWarning: set_ticklabels() should only be used with a fixed number of ticks, i.e. after set_ticks() or using a FixedLocator.
  ax.set_xticklabels(medals.index, rotation=90)
```



Creating histograms

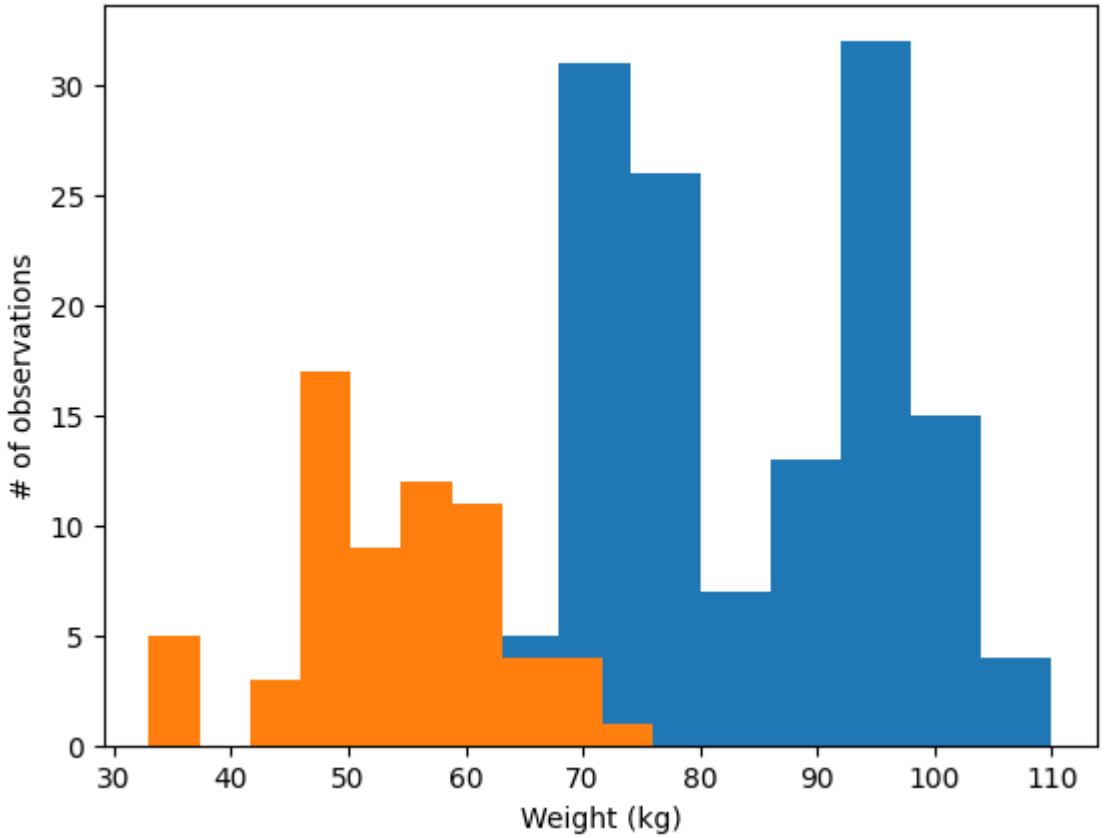
In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Specify the file path using double backslashes
summer2016 = pd.read_csv('C:\\Users\\yeiso\\OneDrive - Douglas College\\0. DOUGLAS COLLEGE\\3. Fund Machine Learning\\0. Python Course Data

# Display the first few rows of the DataFrame
summer2016.head()
```

Out[ ]:

	Unnamed: 0	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
0	158	62	Giovanni Abagnale	M	21.0	198.0	90.0	Italy	ITA	2016 Summer	2016	Summer	Rio de Janeiro	Rowing	Rowing Men's Coxless Pairs	Bronze
1	161	65	Patimat Abakarova	F	21.0	165.0	49.0	Azerbaijan	AZE	2016 Summer	2016	Summer	Rio de Janeiro	Taekwondo	Taekwondo Women's Flyweight	Bronze
2	175	73	Luc Abalo	M	31.0	182.0	86.0	France	FRA	2016 Summer	2016	Summer	Rio de Janeiro	Handball	Handball Men's Handball	Silver
3	450	250	Saeid Morad Abdevali	M	26.0	170.0	80.0	Iran	IRI	2016 Summer	2016	Summer	Rio de Janeiro	Wrestling	Wrestling Men's Middleweight, Greco-Roman	Bronze
4	794	455	Denis Mikhaylovich Ablyazin	M	24.0	161.0	62.0	Russia	RUS	2016 Summer	2016	Summer	Rio de Janeiro	Gymnastics	Gymnastics Men's Team All-Around	Silver

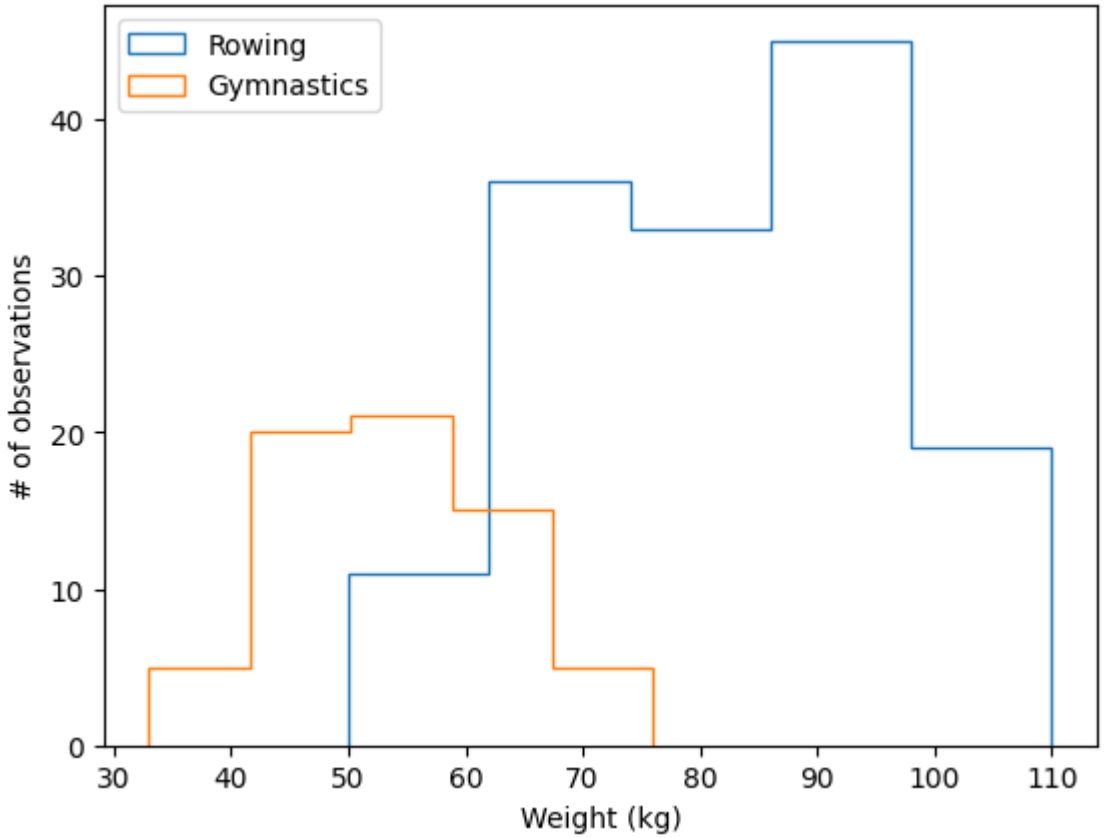
In [ ]:

```
# Select data for mens_rowing
mens_rowing = summer2016[summer2016['Sport'] == "Rowing"]

# Select data for mens_gymnastics
mens_gymnastics = summer2016[summer2016['Sport'] == "Gymnastics"]



#second part of the code

fig, ax = plt.subplots()
# Plot a histogram of "Weight" for mens_rowing

ax.hist(mens_rowing.Weight)

# Compare to histogram of "Weight" for mens_gymnastics
ax.hist(mens_gymnastics.Weight)

# Set the x-axis label to "Weight (kg)"
ax.set_xlabel('Weight (kg)')

# Set the y-axis label to "# of observations"
ax.set_ylabel('# of observations')

plt.show()
```



"Step" histogram

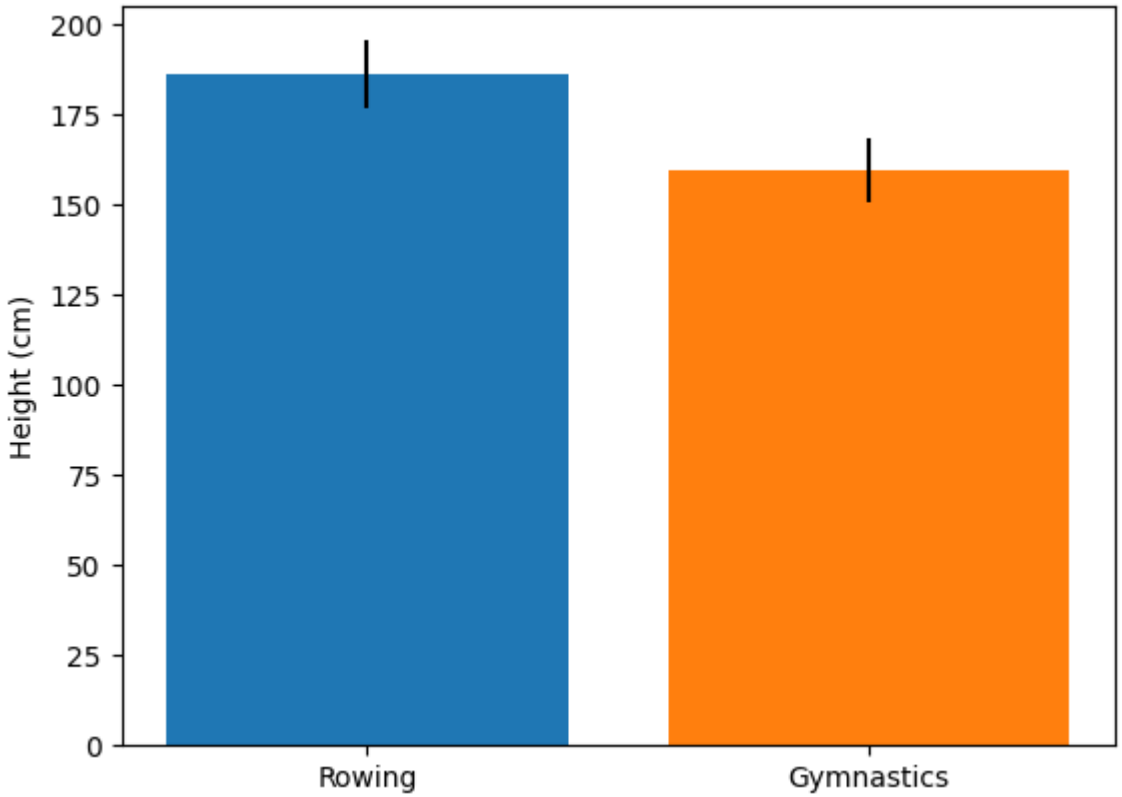
In [ ]:
```
fig, ax = plt.subplots()

# Plot a histogram of "Weight" for mens_rowing
ax.hist(mens_rowing.Weight, label='Rowing', histtype='step', bins=5)

# Compare to histogram of "Weight" for mens_gymnastics
ax.hist(mens_gymnastics.Weight, label='Gymnastics', histtype='step', bins=5)

ax.set_xlabel("Weight (kg)")
ax.set_ylabel("# of observations")

# Add the legend and show the Figure
ax.legend()
plt.show()
```



Adding error-bars to a bar chart

In [ ]:

```
fig, ax = plt.subplots()

# Add a bar for the rowing "Height" column mean/std
ax.bar("Rowing", mens_rowing.Height.mean(), yerr=mens_rowing.Height.std())

# Add a bar for the gymnastics "Height" column mean/std
ax.bar("Gymnastics", mens_gymnastics.Height.mean(), yerr=mens_gymnastics.Height.std())

# Label the y-axis
ax.set_ylabel("Height (cm)")

plt.show()
```

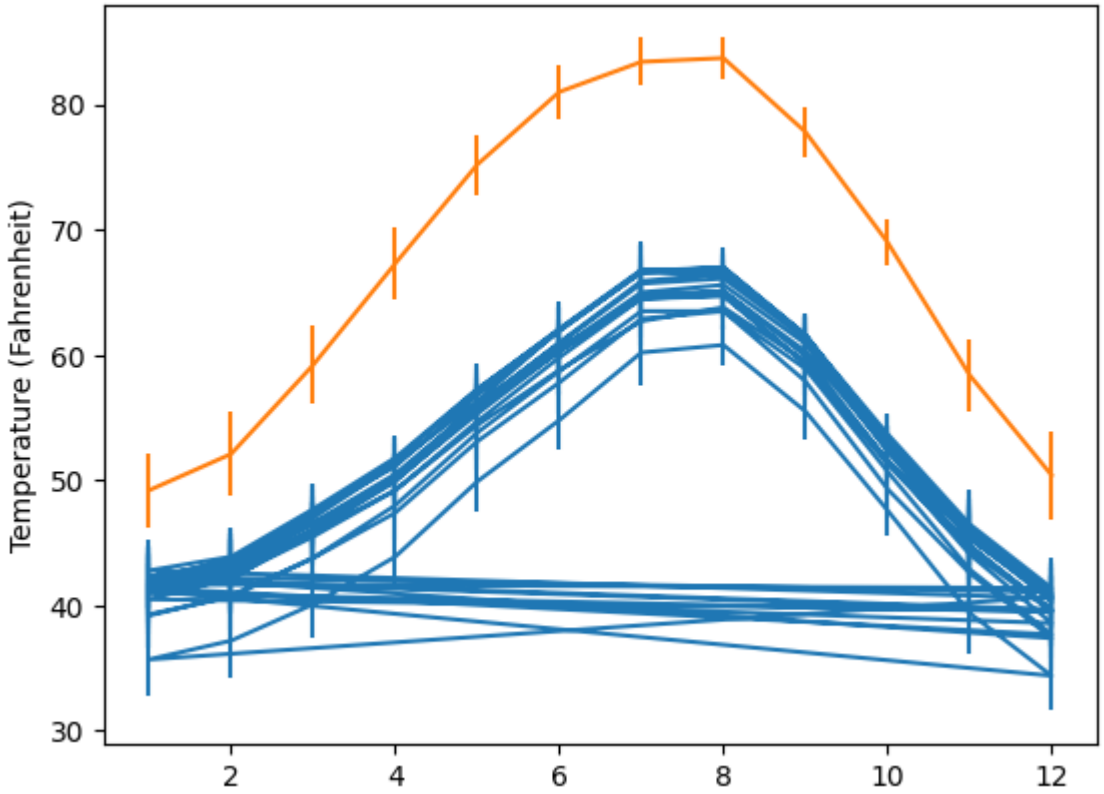


Adding error-bars to a plot

In [ ]:

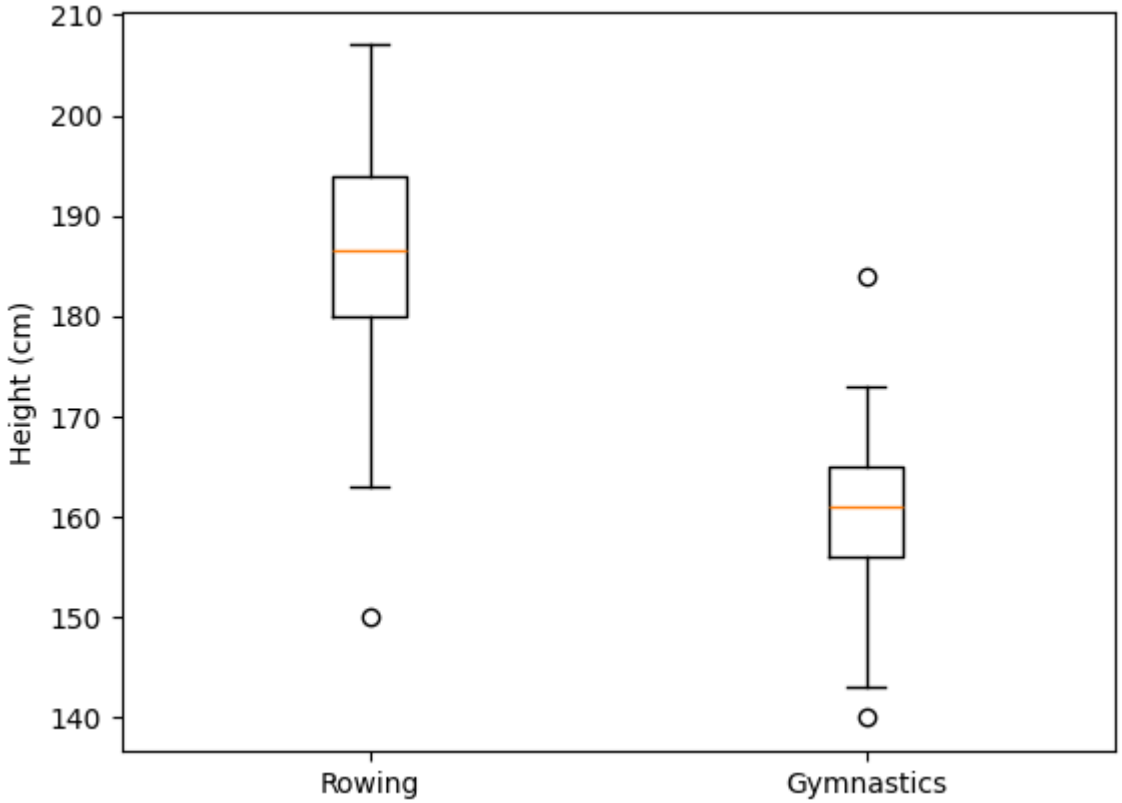
```
# Import pandas as pd
import pandas as pd

# Read the data from file using read_csv
austin_weather = pd.read_csv('C:\\Users\\yeiso\\OneDrive - Douglas College\\0. DOUGLAS COLLEGE\\3. Fund Machine Learning\\0. Python Course
seattle_weather = pd.read_csv('C:\\Users\\yeiso\\OneDrive - Douglas College\\0. DOUGLAS COLLEGE\\3. Fund Machine Learning\\0. Python Course

fig, ax = plt.subplots()

# Add Seattle temperature data in each month with error bars
ax.errorbar(seattle_weather.DATE, seattle_weather['MLY-TAVG-NORMAL'], yerr=seattle_weather['MLY-TAVG-STDDEV'])

# Add Austin temperature data in each month with error bars
ax.errorbar(austin_weather.DATE, austin_weather['MLY-TAVG-NORMAL'], yerr=austin_weather['MLY-TAVG-STDDEV'])

# Set the y-axis label
ax.set_ylabel('Temperature (Fahrenheit)')
```

```
plt.show()
```



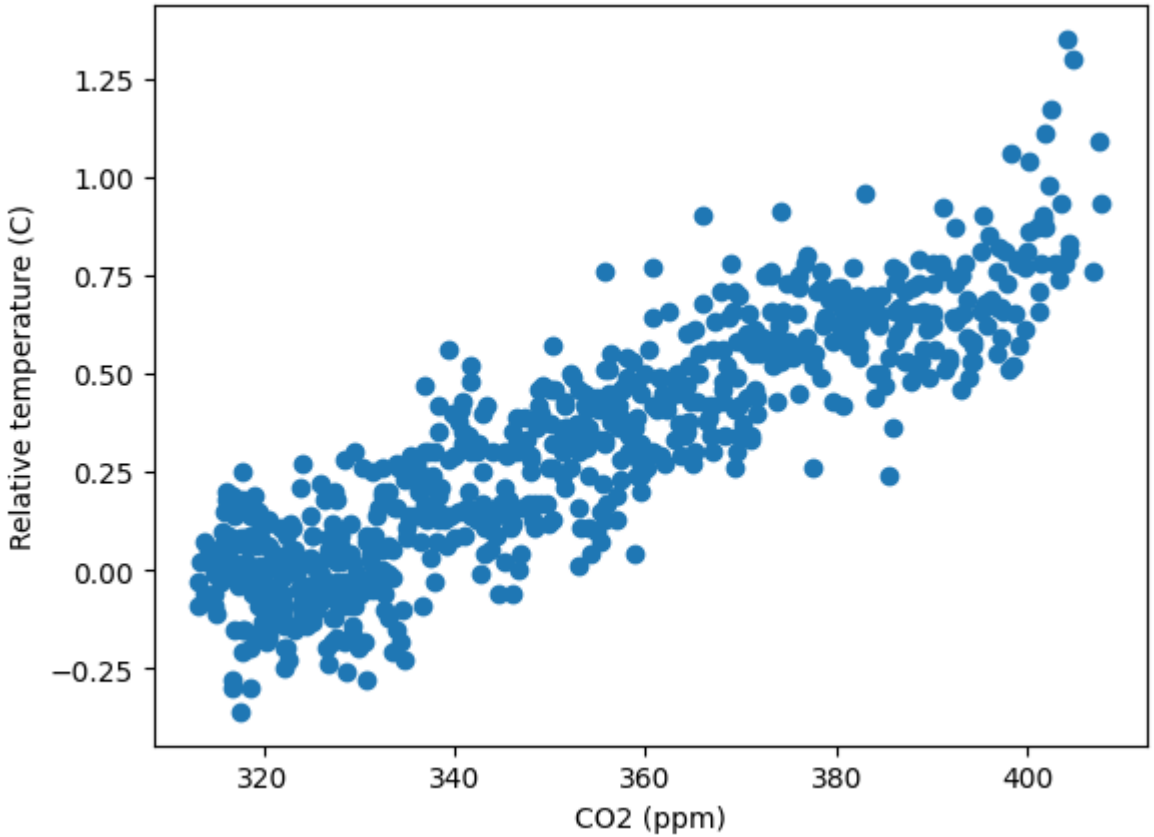
Creating boxplots

In [ ]:
```
fig, ax = plt.subplots()

# Add a boxplot for the "Height" column in the DataFrames
ax.boxplot([mens_rowing.Height, mens_gymnastics.Height])

# Add x-axis tick labels:
ax.set_xticklabels(['Rowing', 'Gymnastics'])

# Add a y-axis label
ax.set_ylabel('Height (cm)')

plt.show()
```



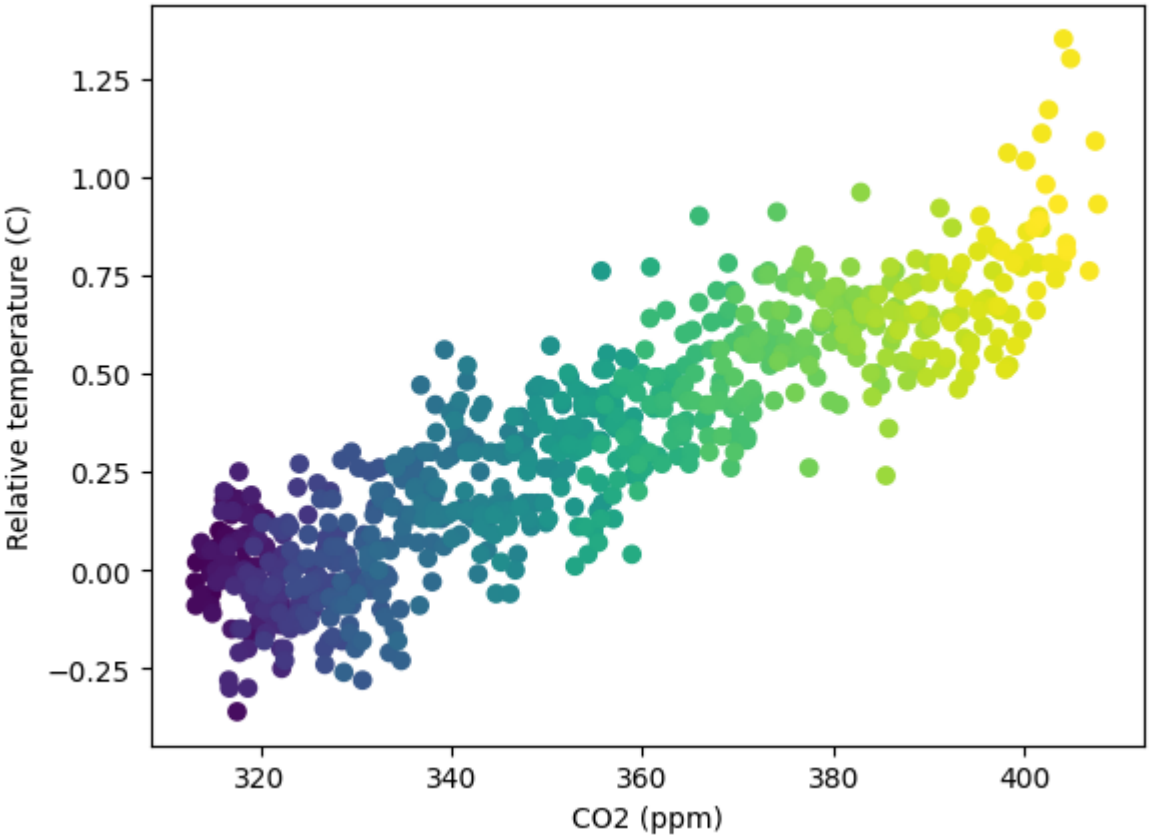
Simple scatter plot

In [ ]:
```
fig, ax = plt.subplots()

# Add data: "co2" on x-axis, "relative_temp" on y-axis
ax.scatter(climate_change.co2, climate_change.relative_temp)

# Set the x-axis label to "CO2 (ppm)"
ax.set_xlabel('CO2 (ppm)')

# Set the y-axis label to "Relative temperature (C)"
ax.set_ylabel('Relative temperature (C)')

plt.show()
```



Encoding time by color

In [ ]:
```
fig, ax = plt.subplots()

# Add data: "co2", "relative_temp" as x-y, index as color
ax.scatter(climate_change.co2, climate_change.relative_temp, c=climate_change.index)

# Set the x-axis label to "CO2 (ppm)"
ax.set_xlabel('CO2 (ppm)')

# Set the y-axis label to "Relative temperature (C)"
ax.set_ylabel('Relative temperature (C)')

plt.show()
```



Chapter 4 - Sharing visualizations with others

This chapter shows you how to share your visualizations with others: how to save your figures as files, how to adjust their look and feel, and how to automate their creation based on input data.

Selecting a style for printing

This chapter shows you how to share your visualizations with others: how to save your figures as files, how to adjust their look and feel, and how to automate their creation based on input data.

Selecting a style for printing

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

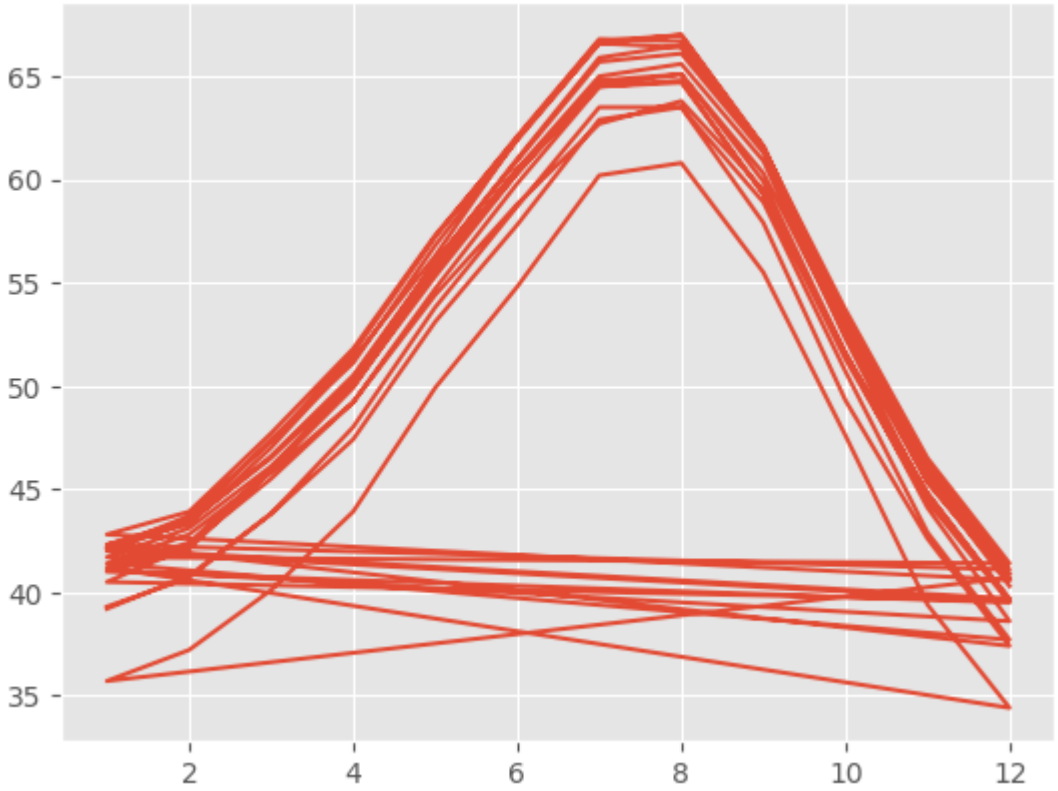
In [ ]:
```
# Read the data from file using read_csv
austin_weather = pd.read_csv('C:\\Users\\yeiso\\OneDrive - Douglas College\\0. DOUGLAS COLLEGE\\3. Fund Machine Learning\\0. Python Course
seattle_weather = pd.read_csv('C:\\Users\\yeiso\\OneDrive - Douglas College\\0. DOUGLAS COLLEGE\\3. Fund Machine Learning\\0. Python Course
```

In [ ]:
```
'grayscale'
```

Out[ ]:
```
'grayscale'
```

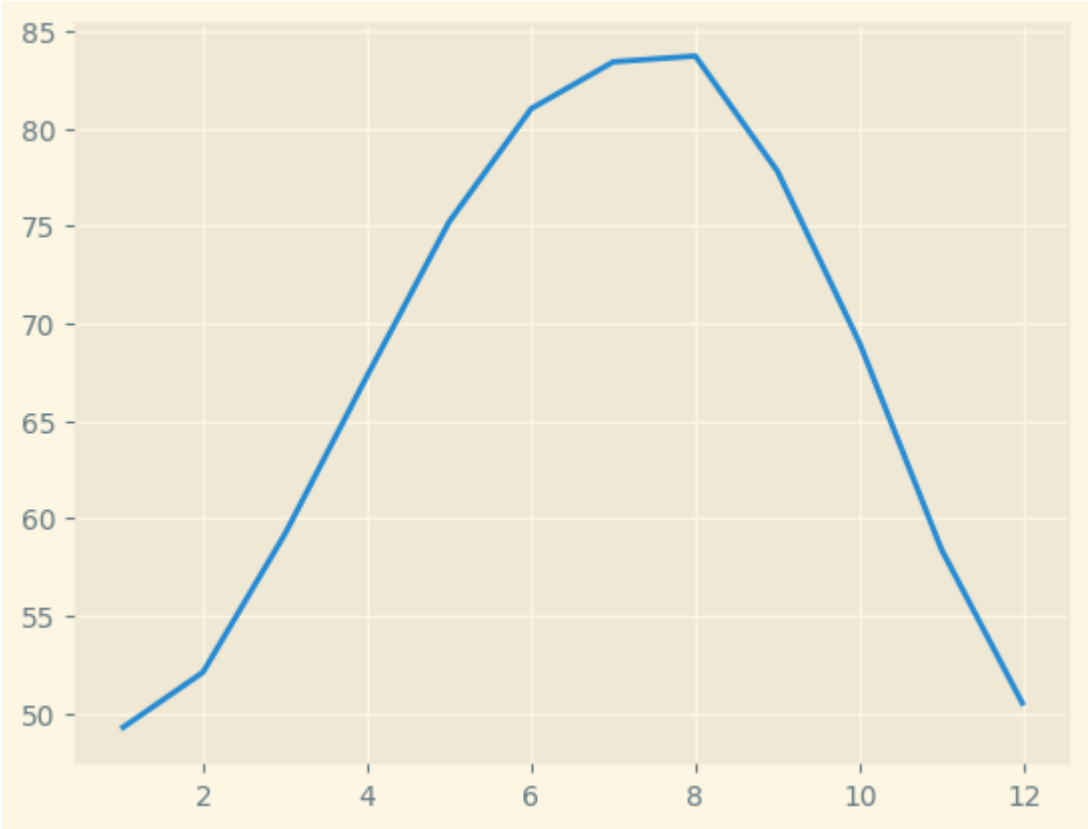
Switching between styles

In [ ]:
```
# Use the "ggplot" style and create new Figure/Axes
plt.style.use('ggplot')
fig, ax = plt.subplots()
ax.plot(seattle_weather["DATE"], seattle_weather["MLY-TAVG-NORMAL"])
plt.show()
```



In [ ]:
```
# Use the "Solarize_Light2" style and create new Figure/Axes
plt.style.use('Solarize_Light2')
```

```
fig, ax = plt.subplots()
ax.plot(austin_weather["DATE"], austin_weather["MLY-TAVG-NORMAL"])
plt.show()
```



Saving a file several times

In [ ]:
```
# Show the figure
plt.show()

# Save as a PNG file
fig.savefig('my_figure.png')
# Save as a PNG file with 300 dpi

fig.savefig('my_figure_300dpi.png', dpi=300)
```

Save a figure with different sizes

In [ ]:
```
# Set figure dimensions and save as a PNG
fig.set_size_inches([3,5])
fig.savefig('figure_3_5.png')

# Set figure dimensions and save as a PNG
fig.set_size_inches([5,3])
fig.savefig('figure_5_3.png')
```

Unique values of a column

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Specify the file path using double backslashes
summer_2016_medals = pd.read_csv('C:\\Users\\yeiso\\OneDrive - Douglas College\\0. DOUGLAS COLLEGE\\3. Fund Machine Learning\\0. Python Cou

# Display the first few rows of the DataFrame
summer_2016_medals.head()
```

Out[ ]:

	Unnamed: 0	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
0	158	62	Giovanni Abagnale	M	21.0	198.0	90.0	Italy	ITA	2016 Summer	2016	Summer	Rio de Janeiro	Rowing	Rowing Men's Coxless Pairs	Bronze
1	161	65	Patimat Abakarova	F	21.0	165.0	49.0	Azerbaijan	AZE	2016 Summer	2016	Summer	Rio de Janeiro	Taekwondo	Taekwondo Women's Flyweight	Bronze
2	175	73	Luc Abalo	M	31.0	182.0	86.0	France	FRA	2016 Summer	2016	Summer	Rio de Janeiro	Handball	Handball Men's Handball	Silver
3	450	250	Saeid Morad Abdevali	M	26.0	170.0	80.0	Iran	IRI	2016 Summer	2016	Summer	Rio de Janeiro	Wrestling	Wrestling Men's Middleweight, Greco-Roman	Bronze
4	794	455	Denis Mikhaylovich Ablyazin	M	24.0	161.0	62.0	Russia	RUS	2016 Summer	2016	Summer	Rio de Janeiro	Gymnastics	Gymnastics Men's Team All-Around	Silver

In [ ]:
```
# Extract the "Sport" column
sports_column = summer_2016_medals['Sport']

# Find the unique values of the "Sport" column
sports = sports_column.unique()
```

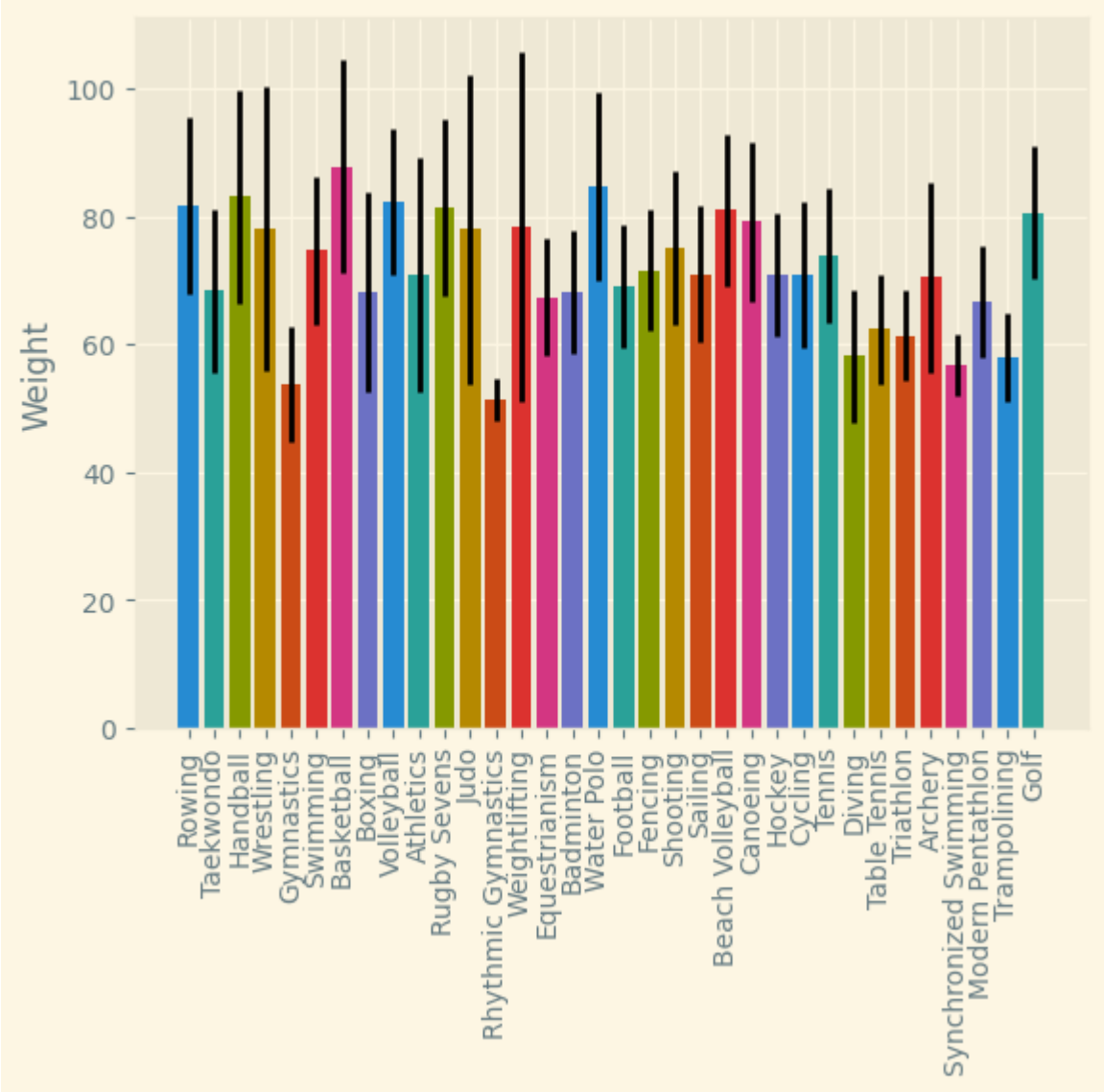
```
# Print out the unique sports values
print(sports)
```

```
['Rowing' 'Taekwondo' 'Handball' 'Wrestling' 'Gymnastics' 'Swimming'
 'Basketball' 'Boxing' 'Volleyball' 'Athletics' 'Rugby Sevens' 'Judo'
 'Rhythmic Gymnastics' 'Weightlifting' 'Equestrianism' 'Badminton'
 'Water Polo' 'Football' 'Fencing' 'Shooting' 'Sailing' 'Beach Volleyball'
 'Canoeing' 'Hockey' 'Cycling' 'Tennis' 'Diving' 'Table Tennis'
 'Triathlon' 'Archery' 'Synchronized Swimming' 'Modern Pentathlon'
 'Trampolining' 'Golf']
```

Automate your visualization

In [ ]:
```
fig, ax = plt.subplots()

# Loop over the different sports branches
for sport in sports:
  # Extract the rows only for this sport
  sport_df = summer_2016_medals[summer_2016_medals['Sport'] == sport]
  # Add a bar for the "Weight" mean with std y error bar
  ax.bar(sport, sport_df["Weight"].mean(), yerr=sport_df["Weight"].std())

ax.set_ylabel("Weight")
ax.set_xticklabels(sports, rotation=90)

# Save the figure to file
fig.savefig('sports_weights.png')
```

```
C:\Users\yeiso\AppData\Local\Temp\ipykernel_18080\1401696868.py:11: UserWarning: set_ticklabels() should only be used with a fixed number o
f ticks, i.e. after set_ticks() or using a FixedLocator.
  ax.set_xticklabels(sports, rotation=90)
```


Weight
0
20
40
60
80
100
Rowing
Taekwondo
Handball
Wrestling
Gymnastics
Swimming
Basketball
Boxing
Volleyball
Athletics
Rugby Sevens
Judo
Rhythmic Gymnastics
Weightlifting
Equestrianism
Badminton
Water Polo
Football
Fencing
Shooting
Sailing
Beach Volleyball
Canoeing
Hockey
Cycling
Tennis
Diving
Table Tennis
Triathlon
Archery
Synchronized Swimming
Modern Pentathlon
Trampolining
Golf