SI 305 Discussion 5: Visualization in Matplotlib

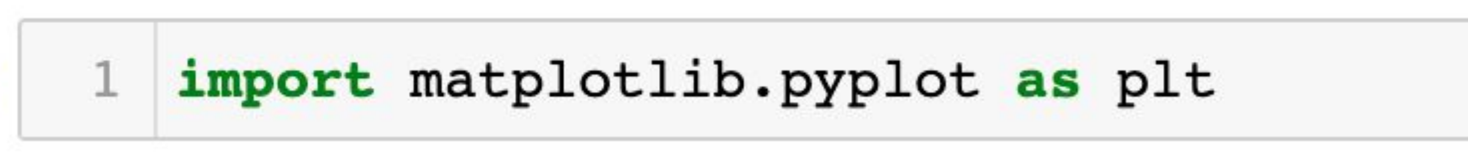
Coulton | Arpitha | Haley

What is matplotlib?

- Open source library
- Supports many types of visualizations
  - e.g. bar charts, histograms, scatterplots
- Many more sophisticated plotting libraries (e.g. proplot, seaborn) are built on top of matplotlib

Importing matplotlib

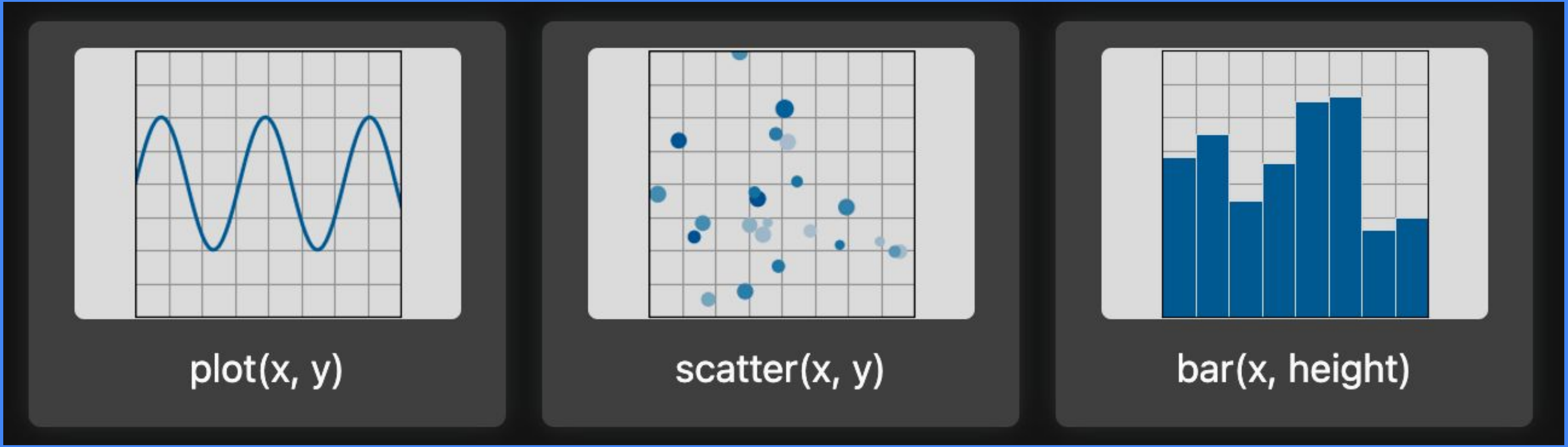
- Unlike pandas, we don’t import matplotlib directly
  - We want to import the pyplot module



```python
import matplotlib.pyplot as plt
```

Types of plots — pairwise data

When we want see the relationship between two variables



plot(x, y)

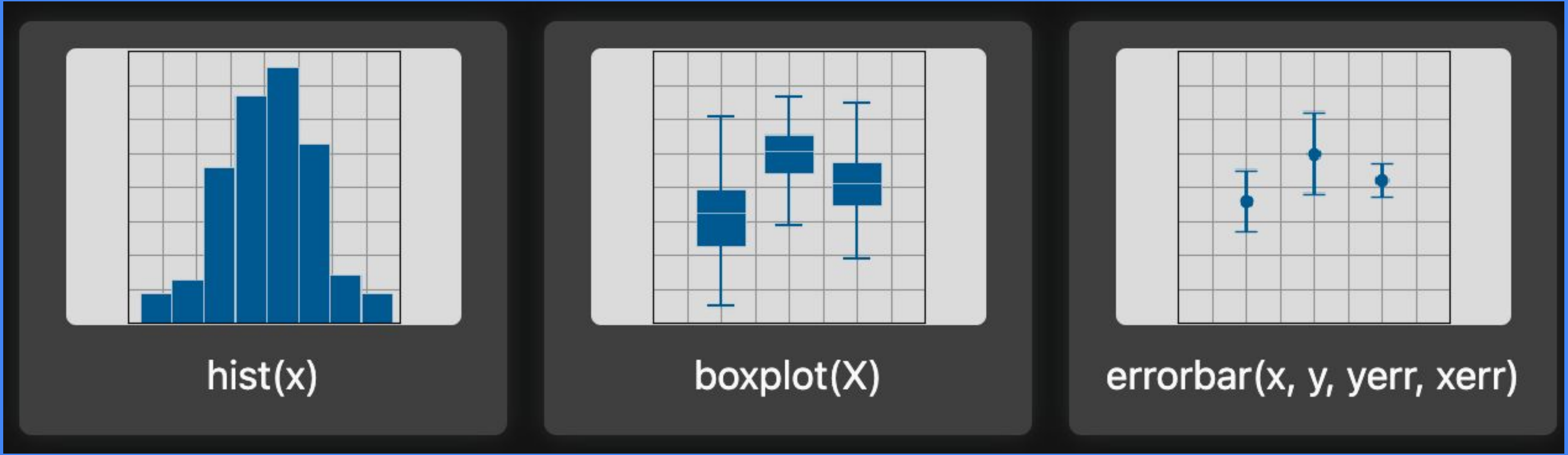
scatter(x, y)

bar(x, height)

See matplotlib documentation for a full list

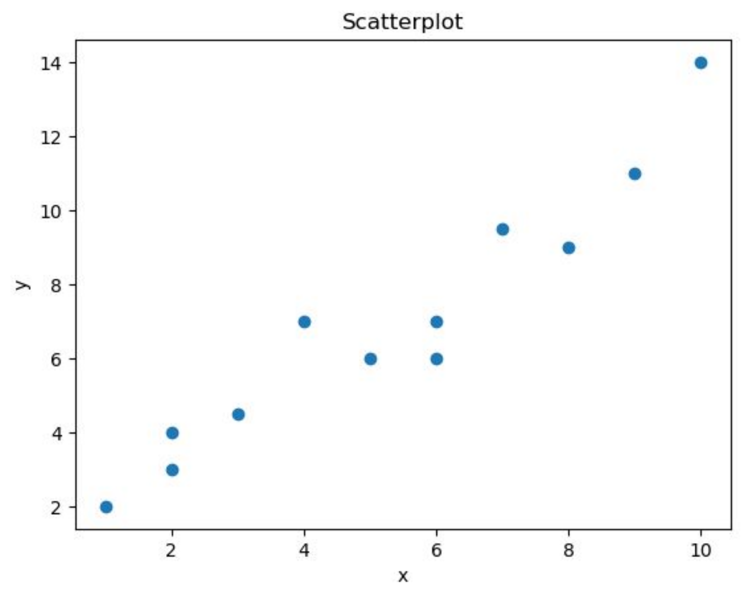
Types of plots — statistical distributions

When we want to understand the underlying distribution of a variable

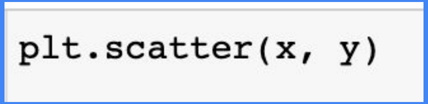


See matplotlib documentation for a full list

How can we recreate this plot?

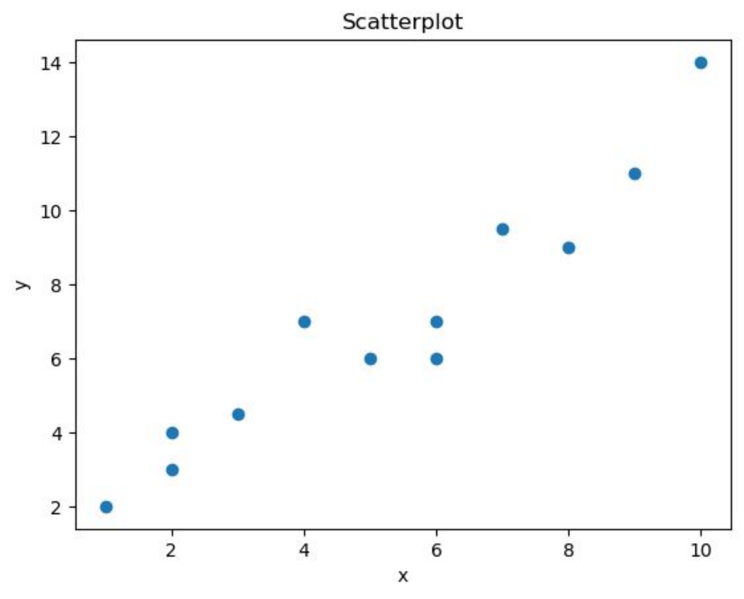


1) Specific the kind of plot we want and the data on each access

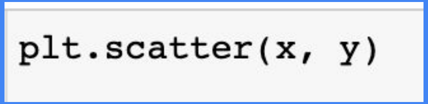


```
plt.scatter(x, y)
```

How can we recreate this plot?

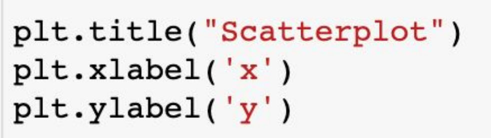


1) Specific the kind of plot we want and the data on each access



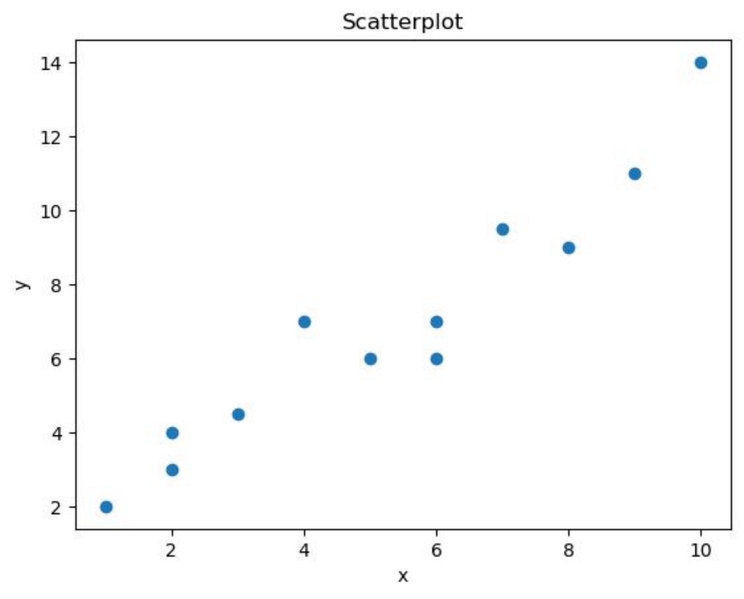
```
plt.scatter(x, y)
```

2) Specific the kind of plot we want and the data on each access

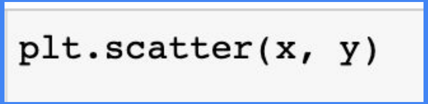


```
plt.title("Scatterplot")
plt.xlabel('x')
plt.ylabel('y')
```

How can we recreate this plot?

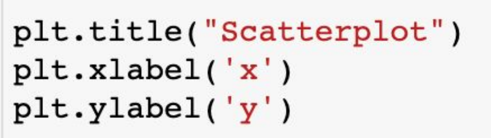


1) Specific the kind of plot we want and the data on each access



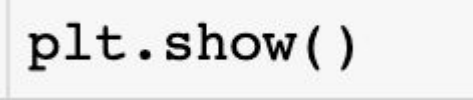
```
plt.scatter(x, y)
```

2) Specific the kind of plot we want and the data on each access



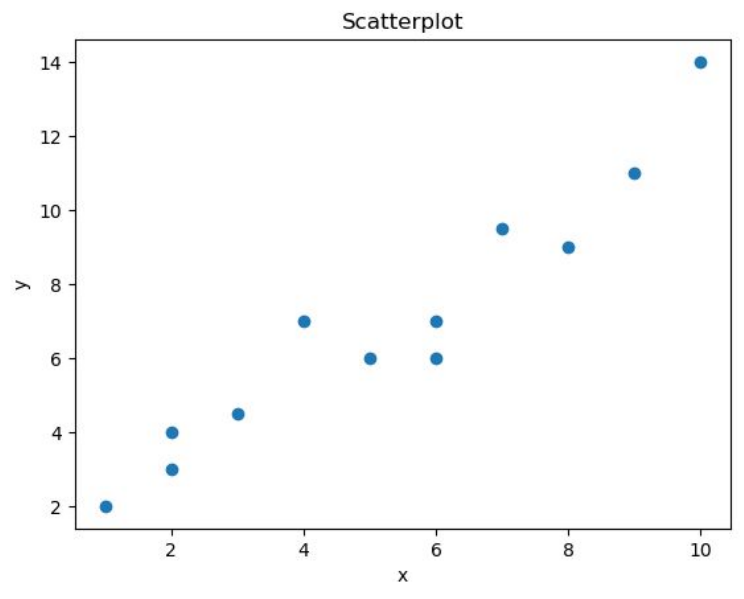
```
plt.title("Scatterplot")
plt.xlabel('x')
plt.ylabel('y')
```

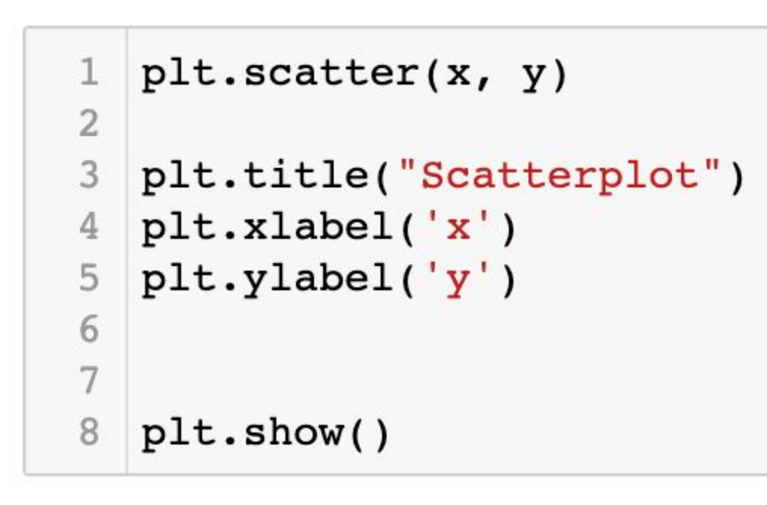
3) Show the plot



```
plt.show()
```

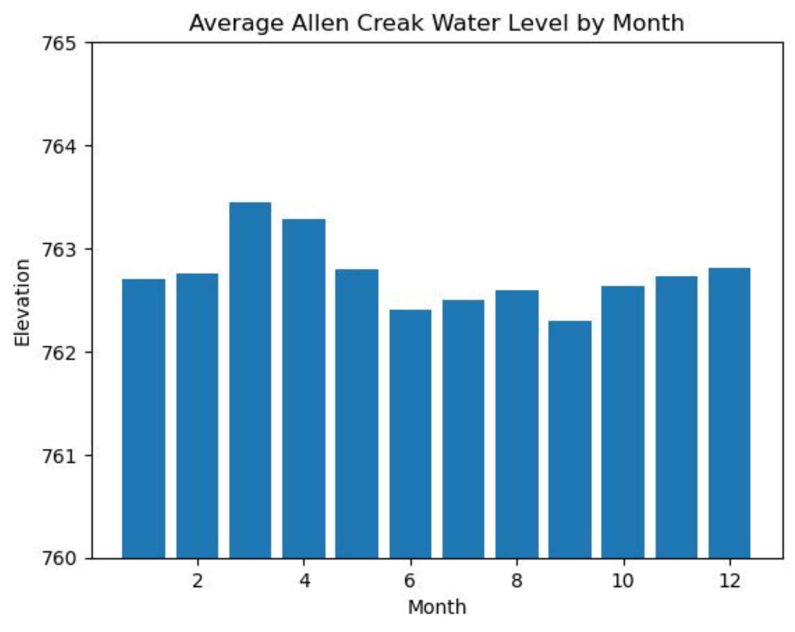
How can we recreate this plot?



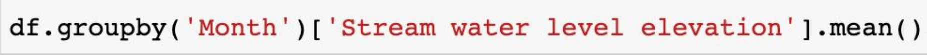


```
plt.scatter(x, y)

plt.title("Scatterplot")
plt.xlabel('x')
plt.ylabel('y')


plt.show()
```

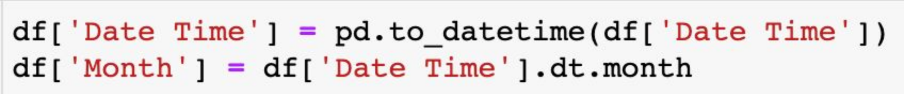
Combining data cleaning & plotting



1) Calculate the average water level each month

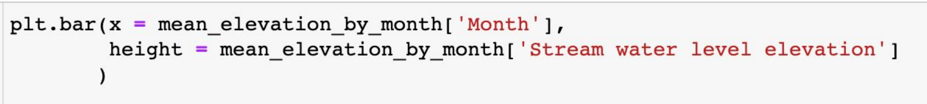


```
df.groupby('Month')['Stream water level elevation'].mean()
```



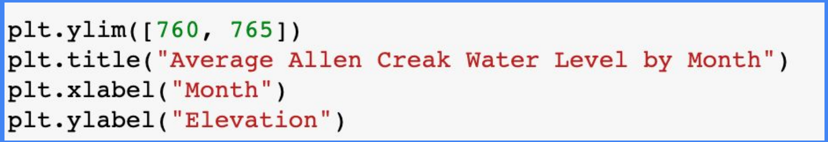
```
df['Date Time'] = pd.to_datetime(df['Date Time'])
df['Month'] = df['Date Time'].dt.month
```

2) Select plot type and specify data



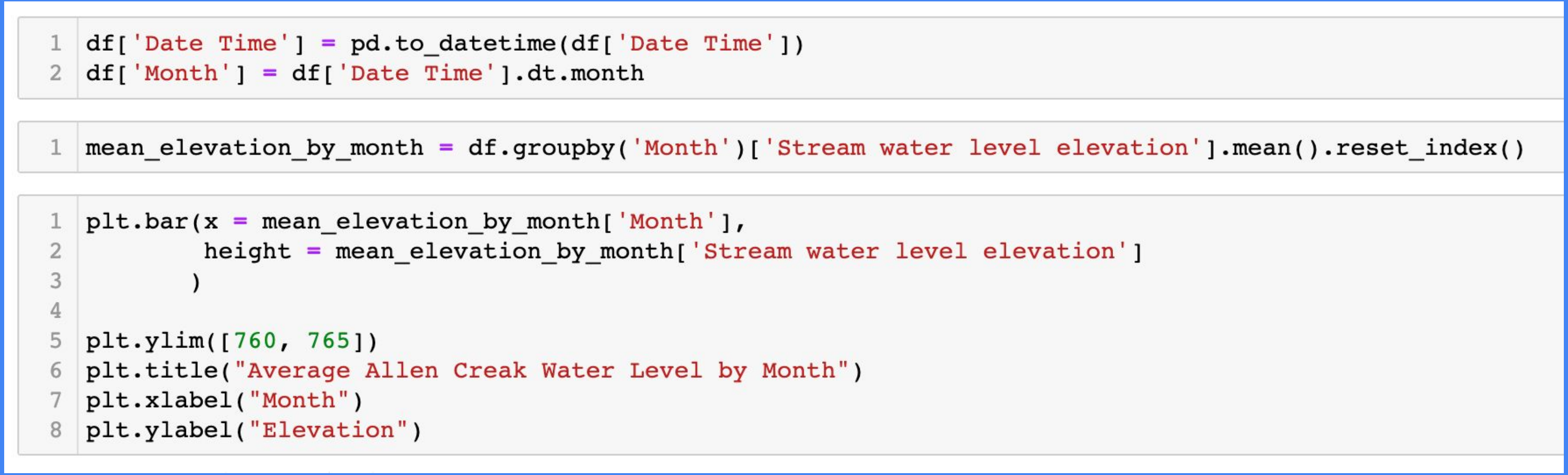
```
plt.bar(x = mean_elevation_by_month['Month'],
        height = mean_elevation_by_month['Stream water level elevation']
       )
```

3) Adjust chart



```
plt.ylim([760, 765])
plt.title("Average Allen Creak Water Level by Month")
plt.xlabel("Month")
plt.ylabel("Elevation")
```

Combining data cleaning & plotting

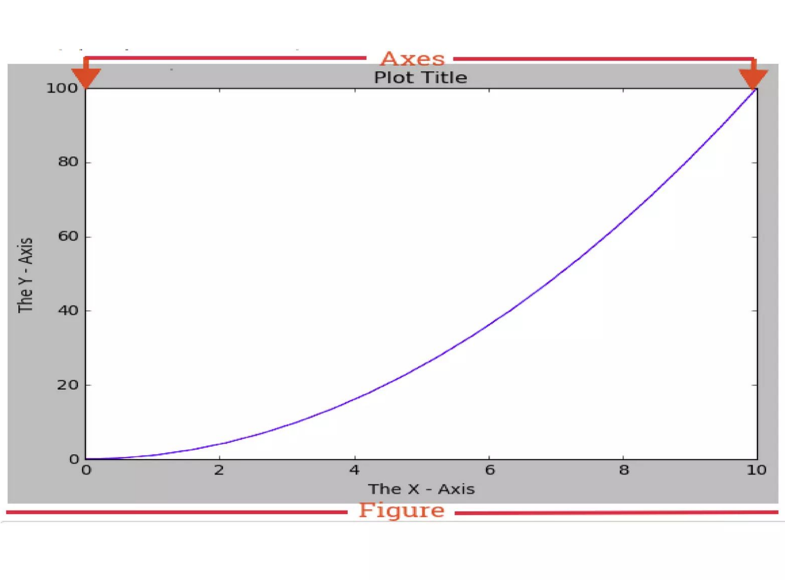


```python
df['Date Time'] = pd.to_datetime(df['Date Time'])
df['Month'] = df['Date Time'].dt.month
```

```python
mean_elevation_by_month = df.groupby('Month')['Stream water level elevation'].mean().reset_index()
```

```python
plt.bar(x = mean_elevation_by_month['Month'],
        height = mean_elevation_by_month['Stream water level elevation']
       )

plt.ylim([760, 765])
plt.title("Average Allen Creak Water Level by Month")
plt.xlabel("Month")
plt.ylabel("Elevation")
```

Figure vs Axes



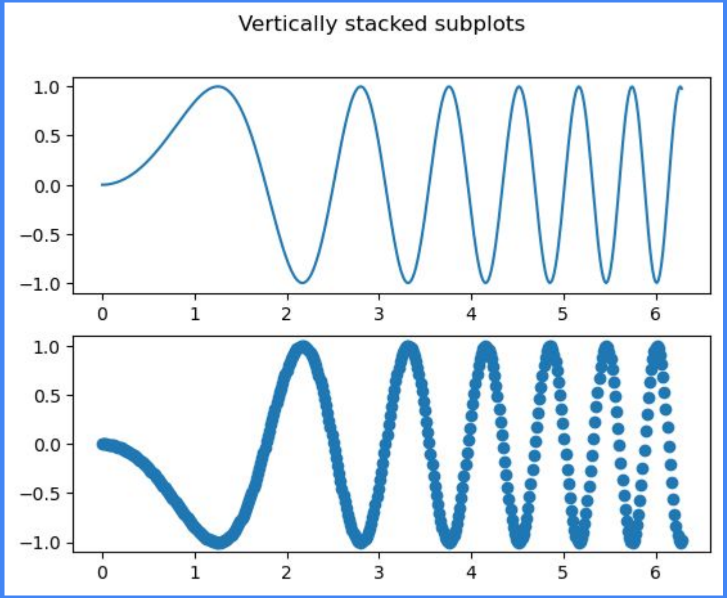
A **figure** is the container for a matplotlib graphic

An **axes** is an individual plot

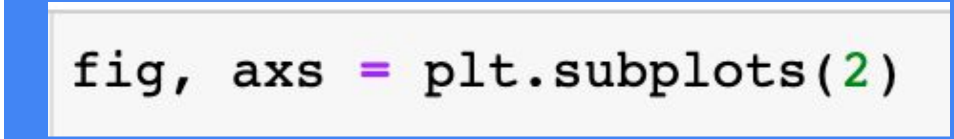
- We use things like .scatter on the axes
- A figure can have multiple axes

This is important when we're creating subplots

Creating multiple plots

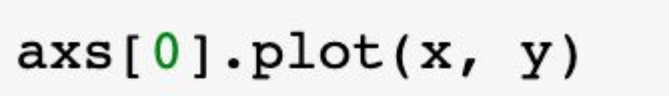


1) Set the figure and axes



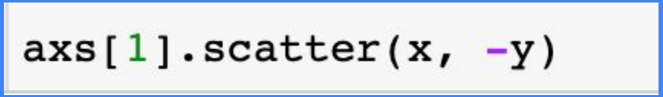
```
fig, axs = plt.subplots(2)
```

2) Assign a plot to the first axes



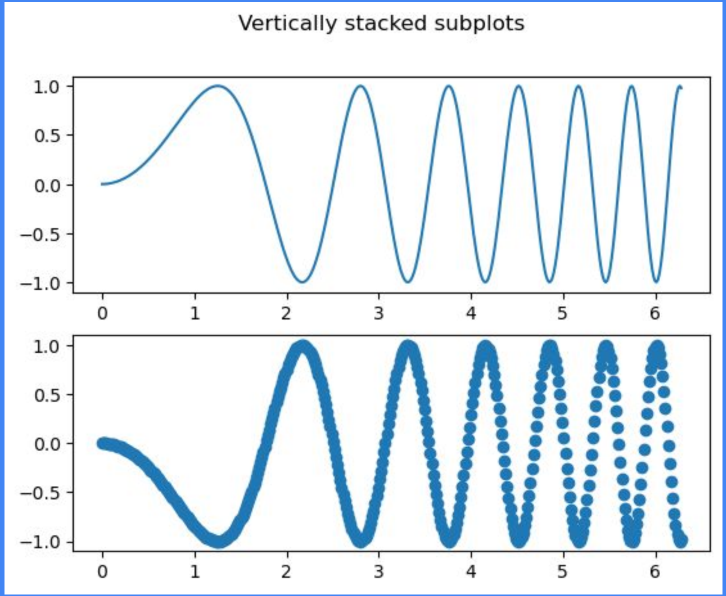
```
axs[0].plot(x, y)
```

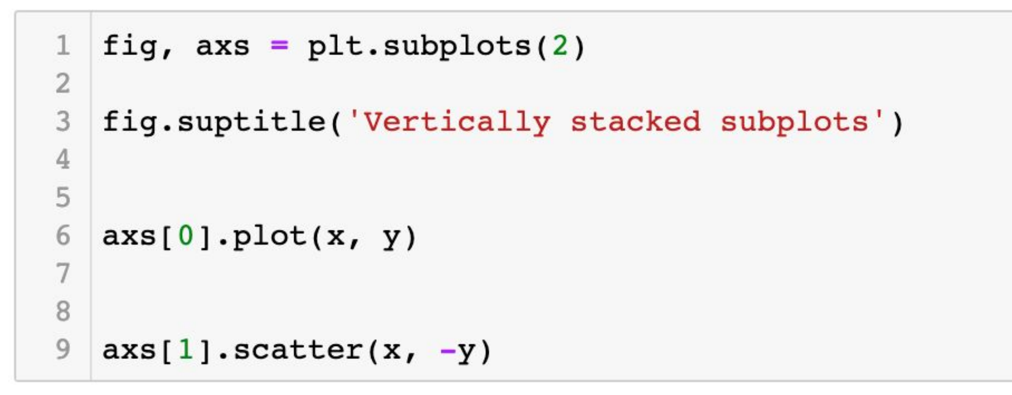
3) Assign a plot to the second axes



```
axs[1].scatter(x, -y)
```

Creating multiple plots





```
fig, axs = plt.subplots(2)

fig.suptitle('Vertically stacked subplots')


axs[0].plot(x, y)


axs[1].scatter(x, -y)
```

Coding Style

- pep8 is the official style guide for Python code
    - Defines things like maximum line length, use of whitespace in expressions, naming conventions for variables

- You don’t need to know or follow every rule in pep8, **but your code should follow best practices**
    - Break up long lines of code, add spaces between function definitions for readability, import all modules at the beginning of a file, use white space appropriately, etc.
    - See examples here