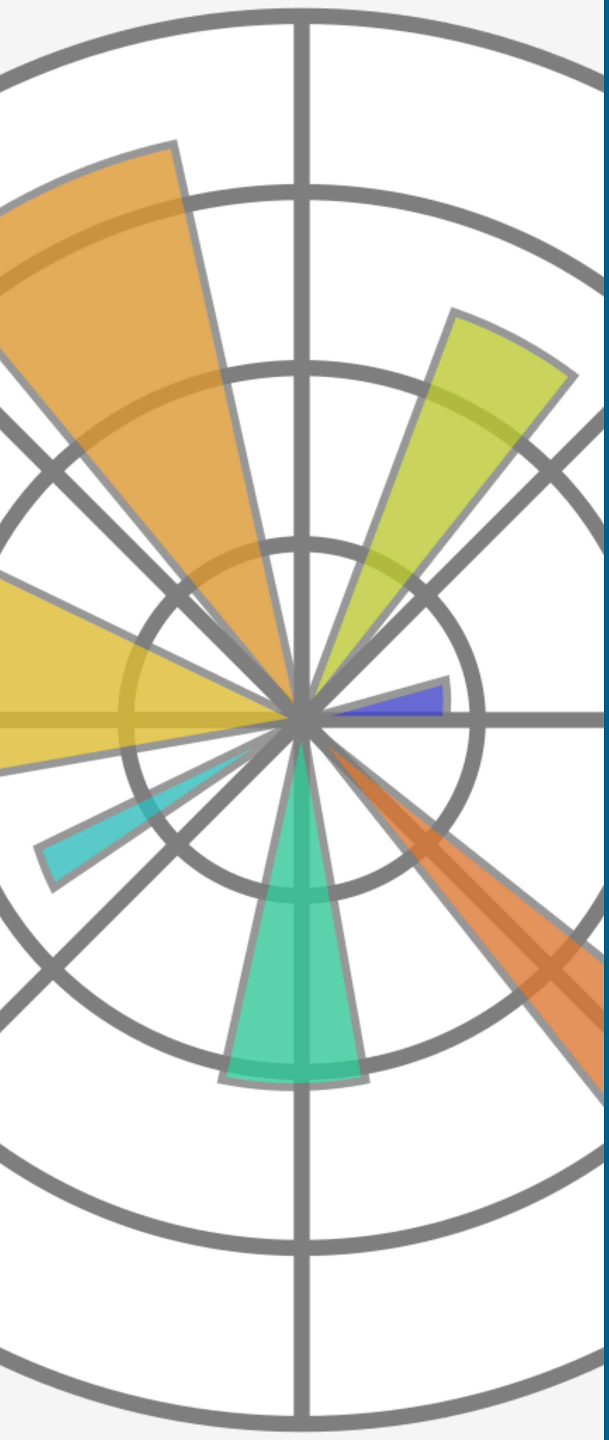
MARCH 26TH, 2025

Plotting

CE 311K - L29



Matplotlib

Matplotlib is the most widely used library for plotting in Python

Extensive support for 2D plots (and some 3D functionality)

Ability to customize every element of a plot

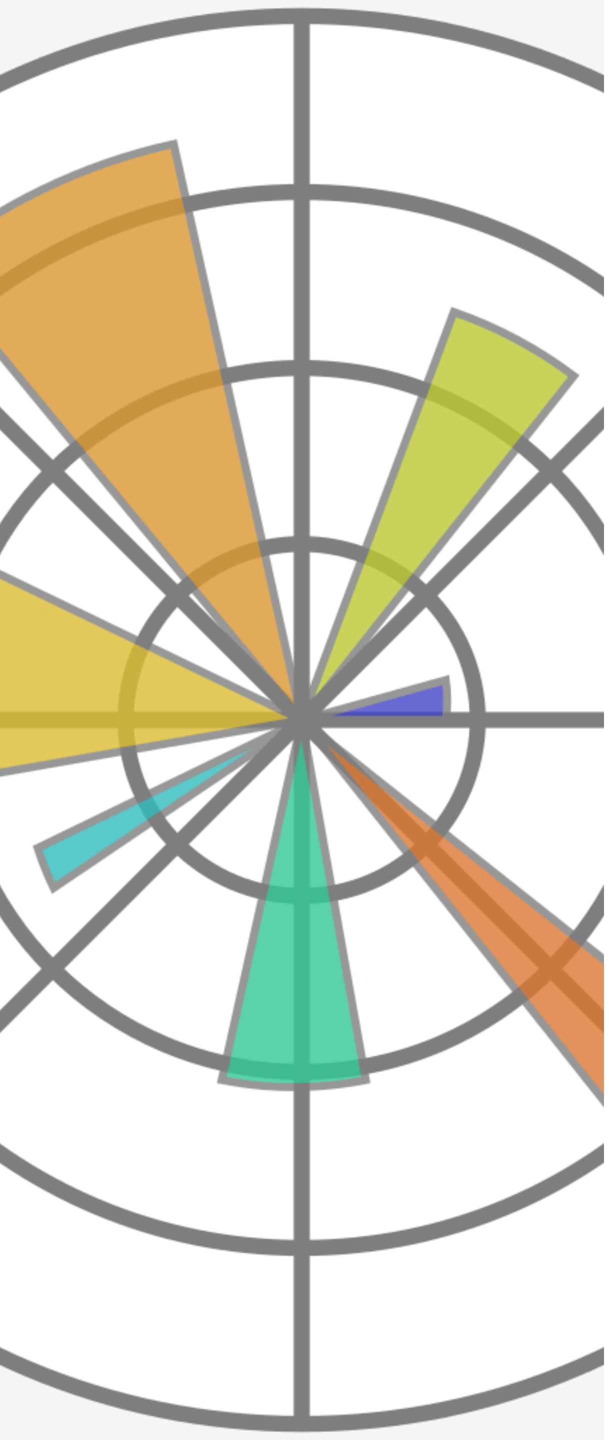
Integration with other common libraries for seamless data visualization

---

You typically import its *pyplot* module under the alias *plt*

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

See the documentation here: https://matplotlib.org/



Figures

A **figure** is the top-level container for all elements of a plot

plt.figure(): Creates a new figure or activates an existing one

```
plt.figure(num=1, figsize=(8, 6), dpi=100)
```

num: sets the figure ID, overwriting any existing figure with that ID number

figsize=(width, height): Sets the size of the figure in inches

dpi: Sets the resolution (dots per inch)

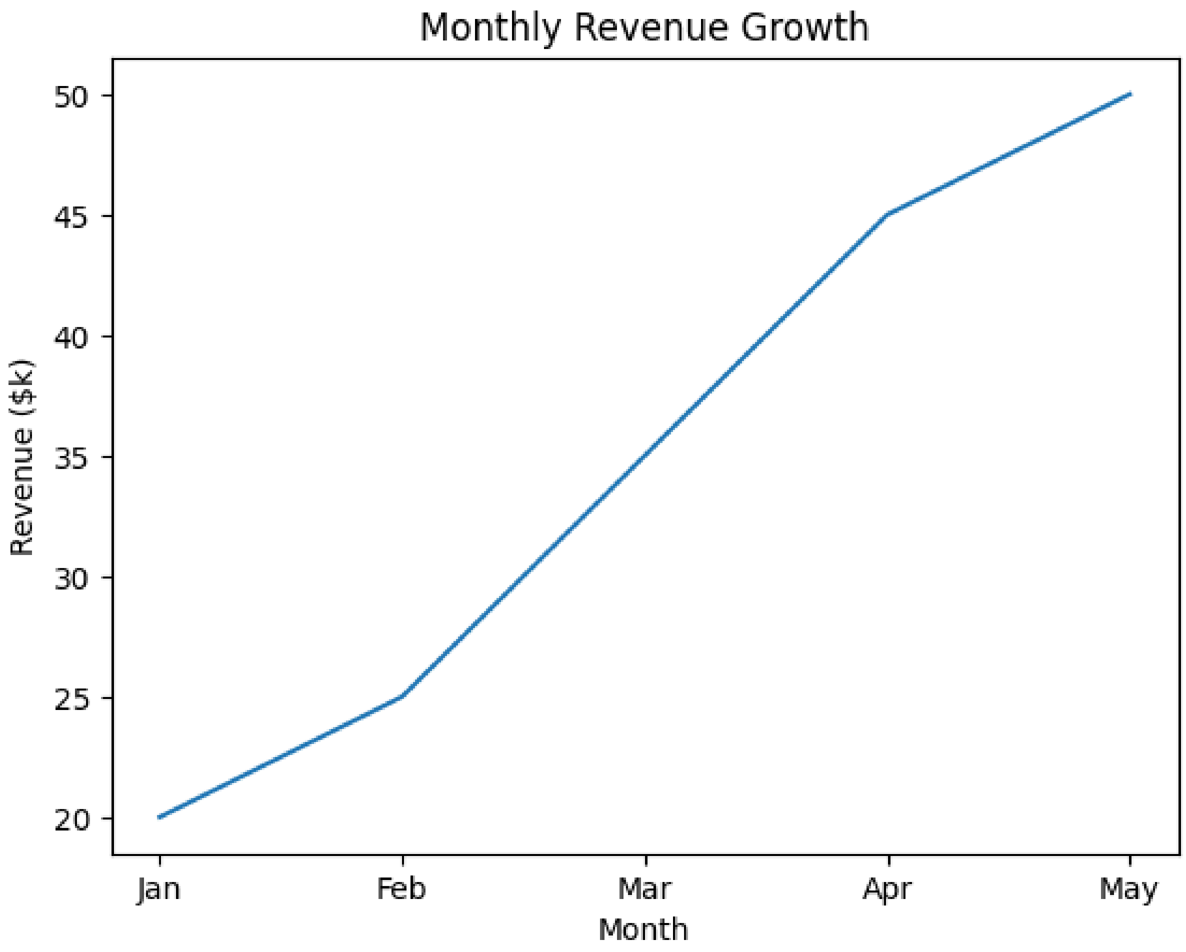
Line Plot

plt.plot(): creates data points connected by lines

Commonly used for trends or time series

```python
# Monthly revenue ($ in thousands)
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May']
revenue = [20, 25, 35, 45, 50]

# Create a simple line plot
plt.figure(1)
plt.plot(months, revenue)
plt.title("Monthly Revenue Growth")
plt.xlabel("Month")
plt.ylabel("Revenue ($k)")
plt.show()
plt.close(1)
```



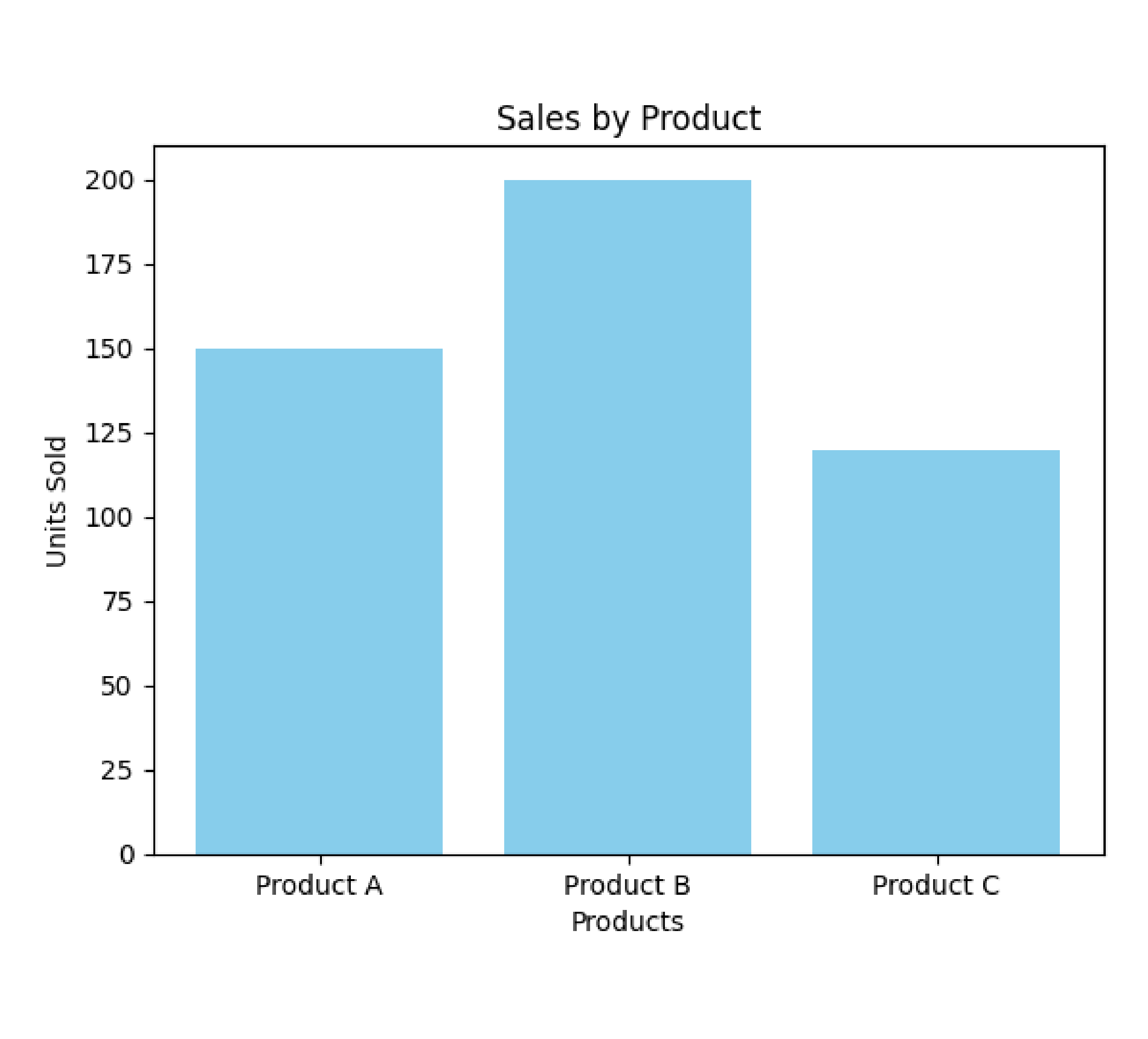
Bar Chart

plt.bar(): Displays categorical data using rectangular bars

Useful for comparisons

```python
# Sales of products in units
products = ['Product A', 'Product B', 'Product C']
sales = [150, 200, 120]

# Create a bar chart
plt.figure(2)
plt.bar(products, sales, color='skyblue')
plt.title("Sales by Product")
plt.xlabel("Products")
plt.ylabel("Units Sold")
plt.show()
plt.close(2)
```



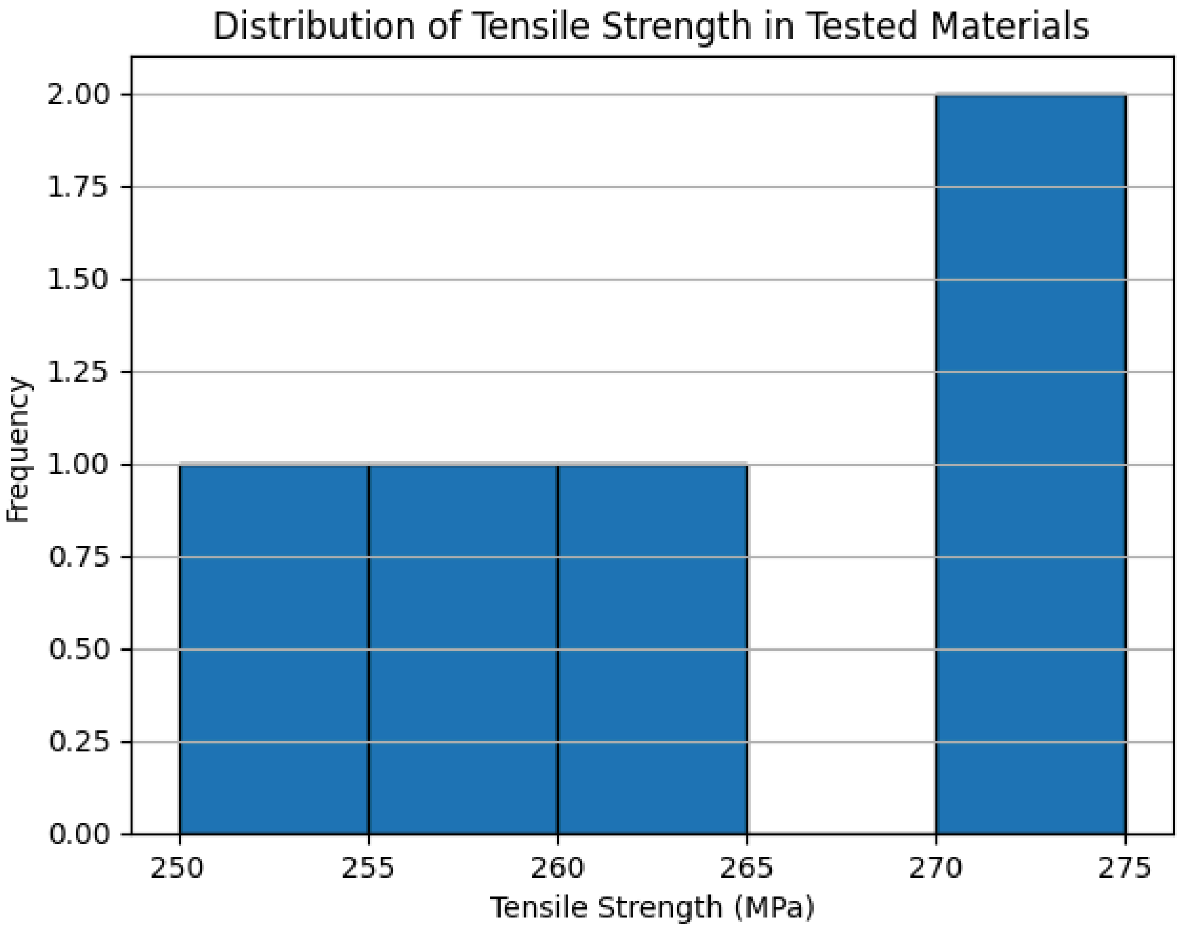
Histogram

plt.hist(): Groups data into intervals (bins) and displays frequency of data points in each bin

Ideal for understanding data distributions or patterns

```
# Tensile strength values (in MPa)
tensile_strength = [250, 260, 255, 270, 275]

# Create a histogram
plt.figure(3)
plt.hist(tensile_strength, bins=5, edgecolor='black')
plt.title("Distribution of Tensile Strength in Tested Materials")
plt.xlabel("Tensile Strength (MPa)")
plt.ylabel("Frequency")
plt.grid(axis='y')
plt.show()
plt.close(3)
```



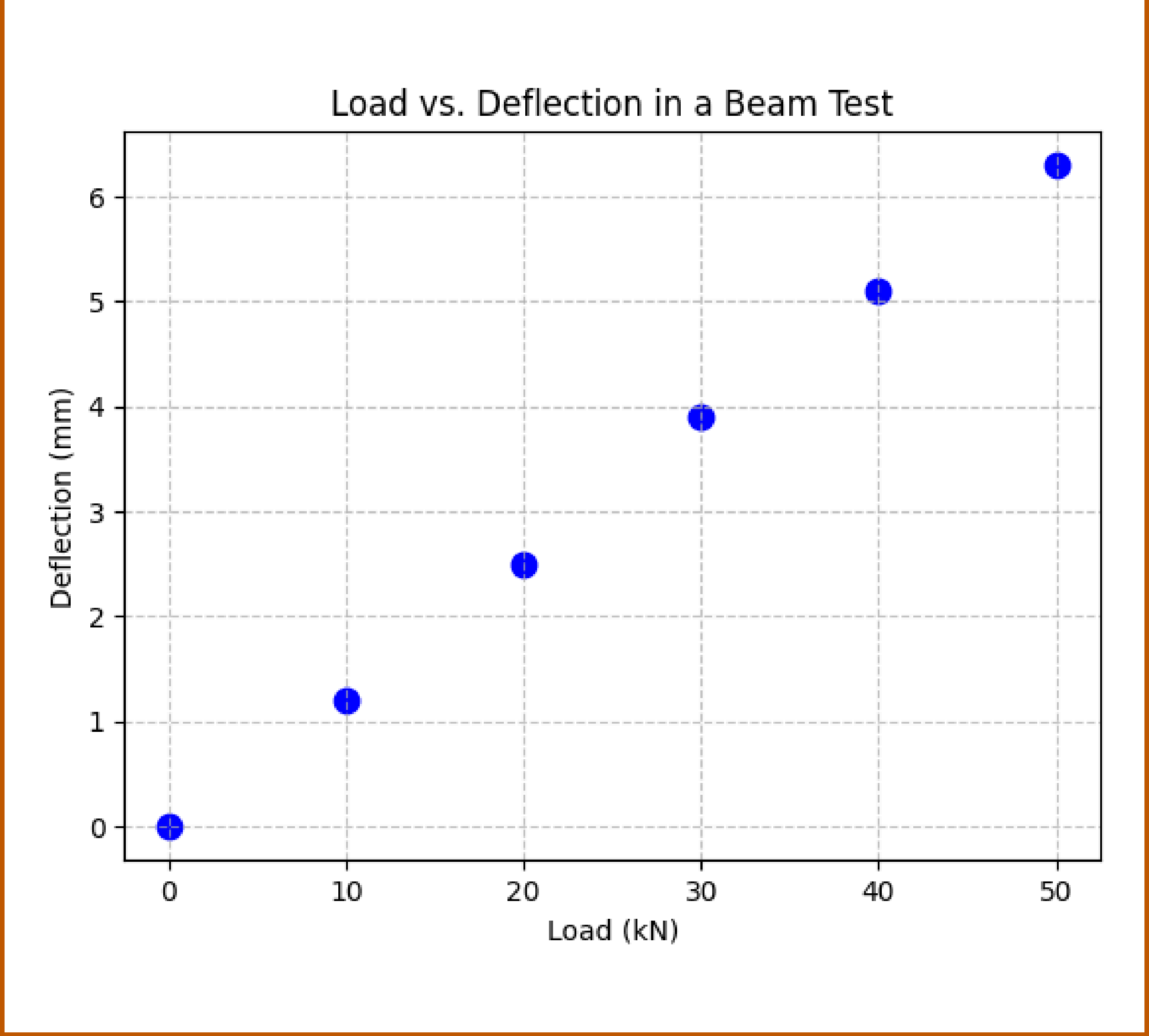
Scatterplot

plt.scatter(): Plots individual data points

Ideal for showing relationships between two variables

```
# Load vs. Deflection in a beam test
load = [0, 10, 20, 30, 40, 50]  # (kN)
deflection = [0, 1.2, 2.5, 3.9, 5.1, 6.3]  # (mm)

# Create a scatter plot
plt.figure()
plt.scatter(load, deflection, s=80, color='blue')
plt.title("Load vs. Deflection in a Beam Test")
plt.xlabel("Load (kN)")
plt.ylabel("Deflection (mm)")
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
plt.close()
```



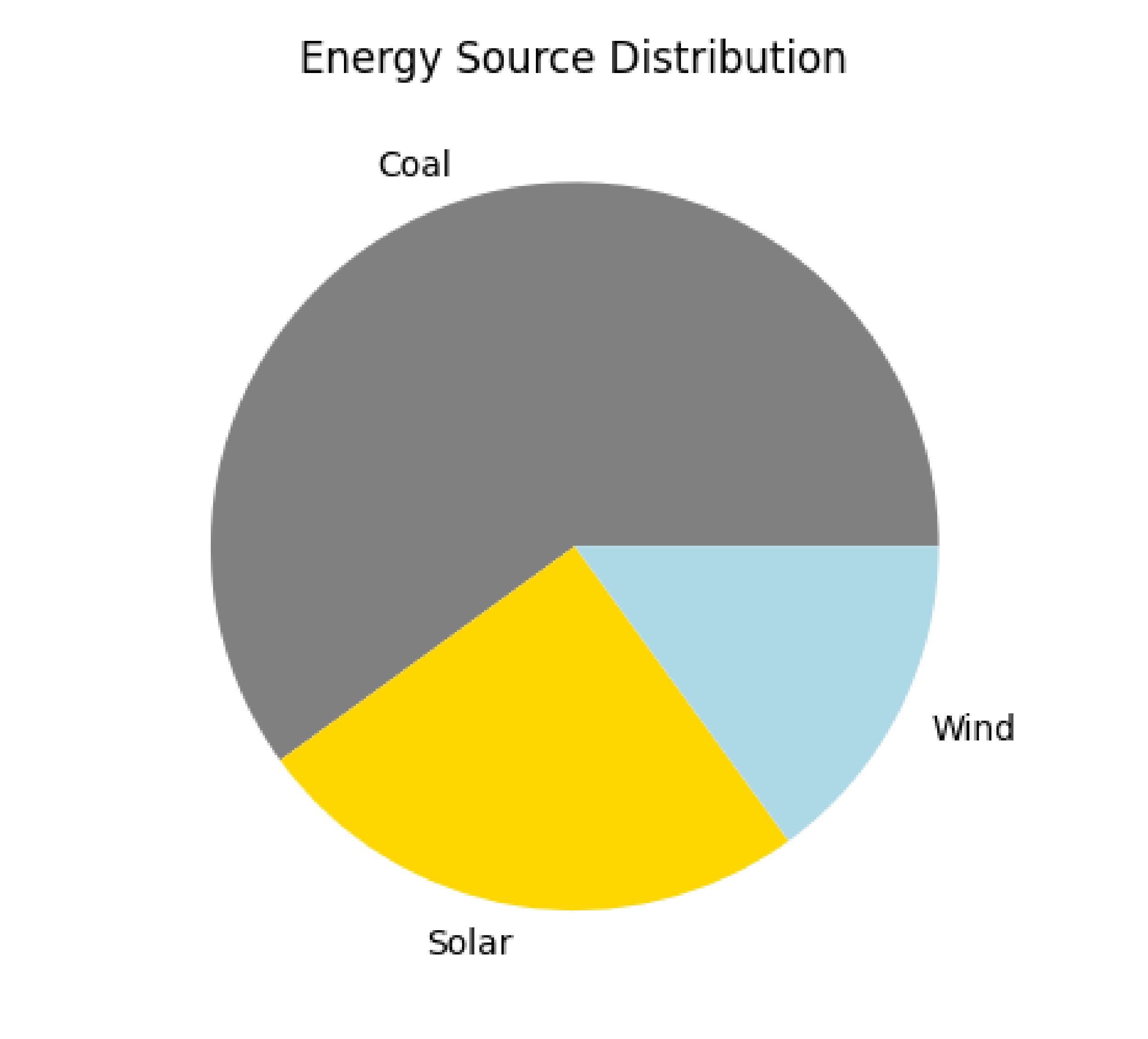
Pie Chart

plt.pie(): Displays data as slices of a circle, representing a category's contribution

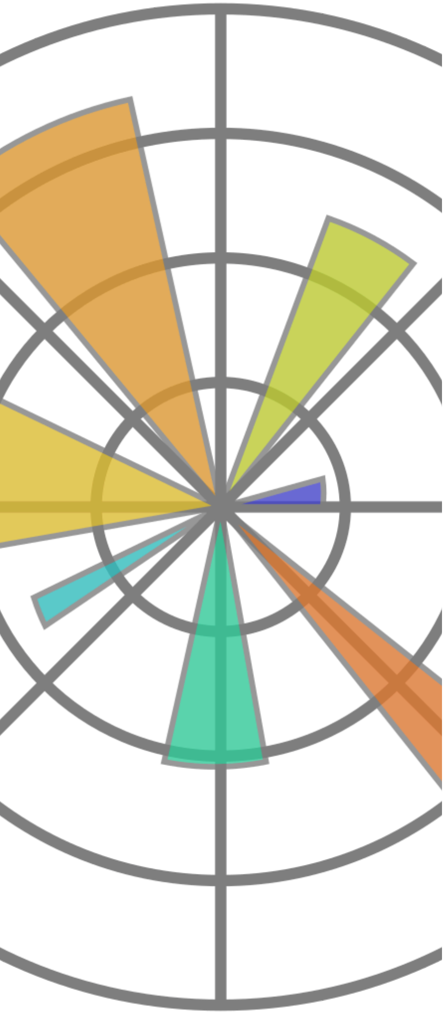
Ideal for showing percentage-based data distributions

```
# Energy source distribution
labels = ['Coal', 'Solar', 'Wind']
sizes = [60, 25, 15]  # Percent contribution of each energy source

# Create a pie chart
plt.figure()
plt.pie(sizes, labels=labels, colors=['grey', 'gold', 'lightblue'])
plt.title("Energy Source Distribution")
plt.show()
plt.close()
```



Summary



figure(), *show()*, and *close()*

figure() defines a figure to use

show() will dsiplay the figure

close() will close the figure

plot(), *bar()*, and *scatter()*

plot() provides a simple line plot

bar() creates a bar chart

scatter() shows individual points

Customize Plots

title() to provide a figure name

xlabel() and *ylabel()* allow you to name the x and y axes respectively

hist() and *pie()*

hist() creates a discrete distribution

pie() shows relative contribution