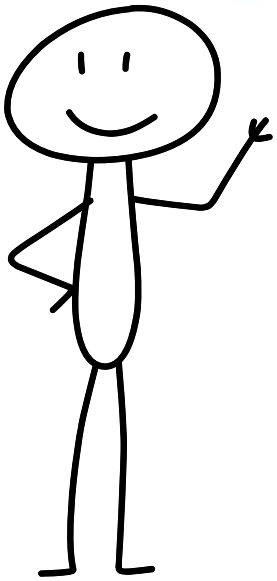
matplotlib



@pycode.hubb

MATPLOTLIB CHEAT SHEET

Basic to Advance

Everything You Need to Know

python™

NEXT

What is Matplotlib?

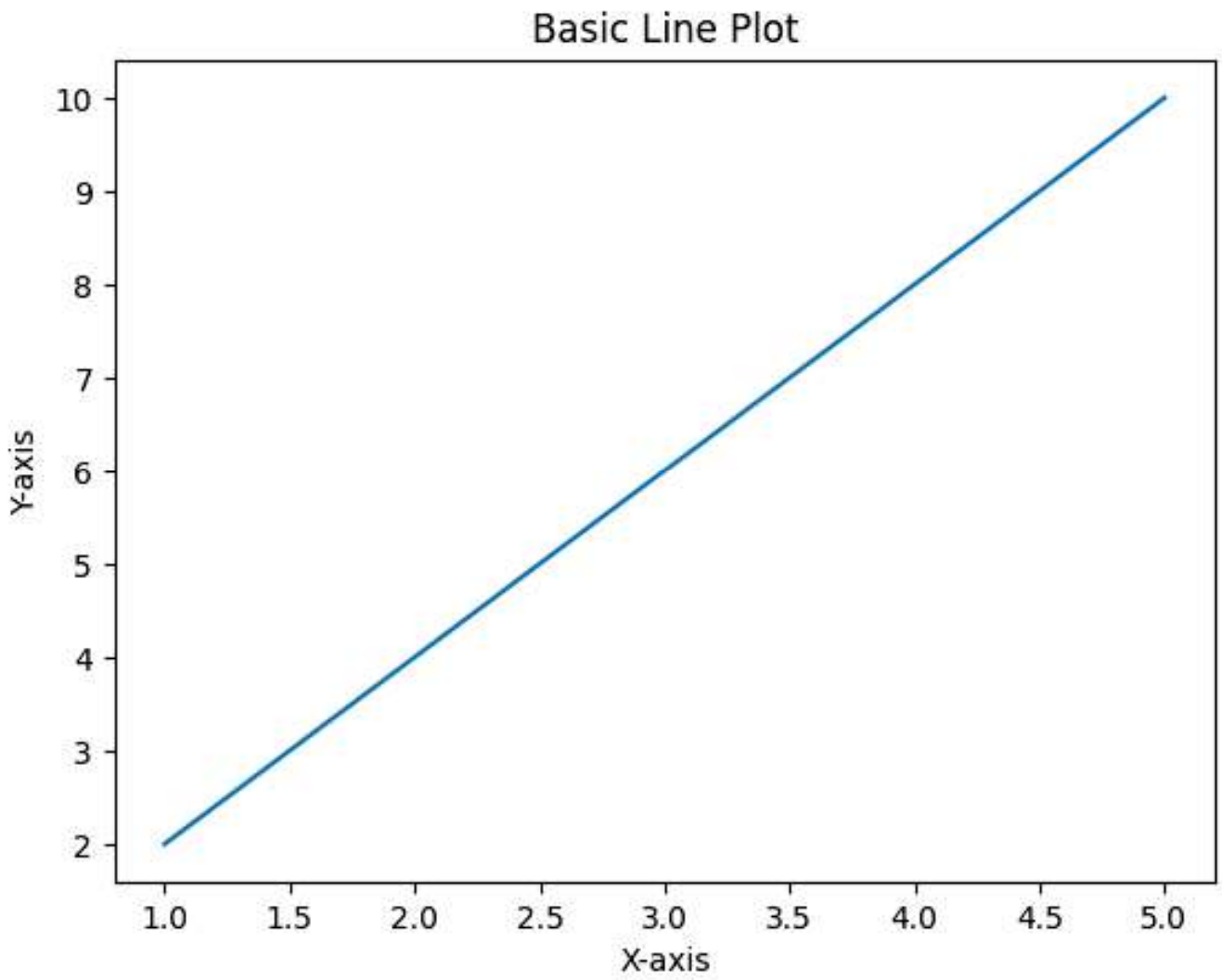
Matplotlib is a powerful Python library for creating a wide range of data visualizations, from simple line charts to complex 3D plots.

How does it work?

We import matplotlib.pyplot and use simple functions like plot(), bar(), and scatter() to visualize your data with full customization.

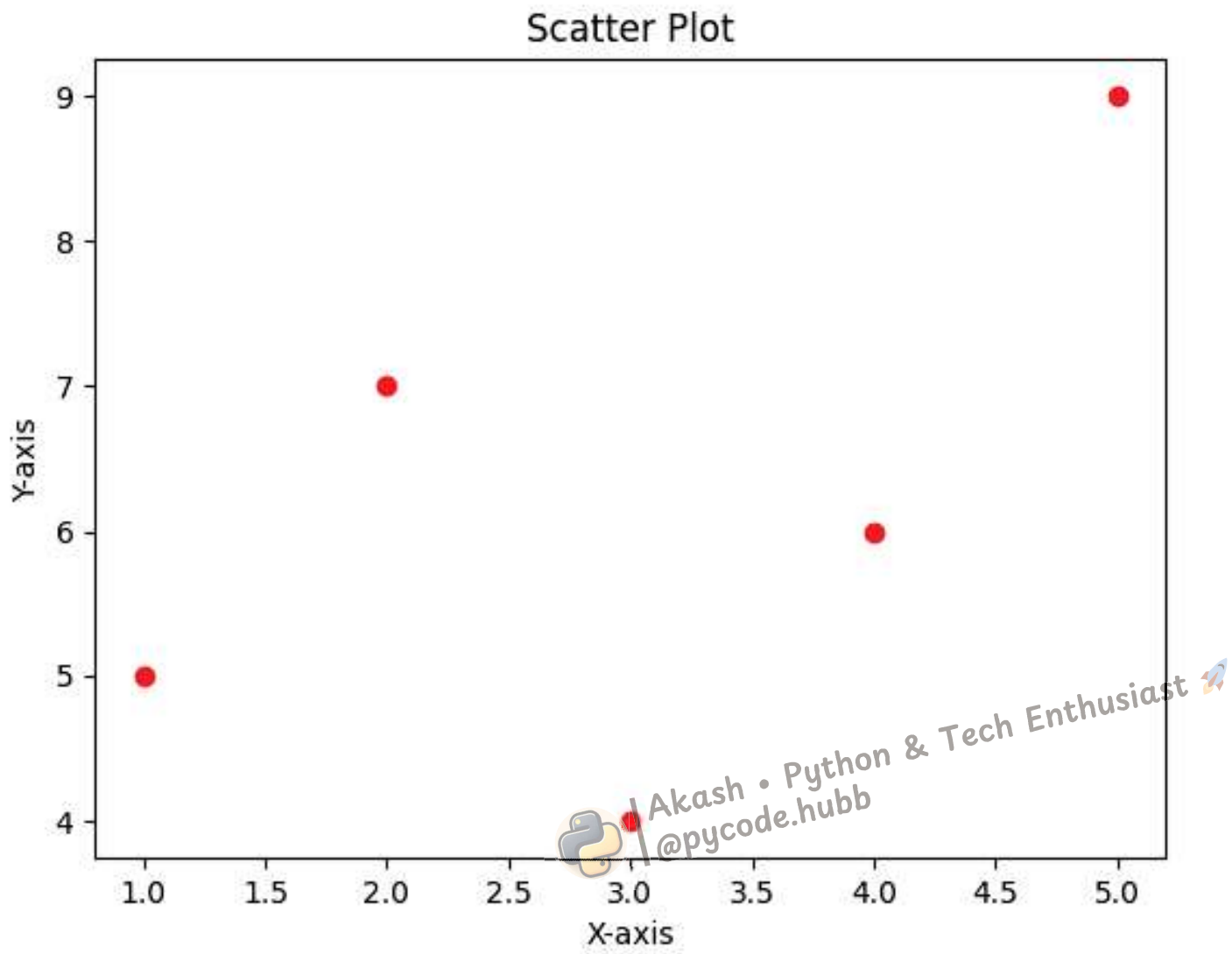
Level 1: Basic Line Plot:

```python
# Importing Matplotlib
import matplotlib.pyplot as plt

# Data for plotting
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Creating a basic line plot
plt.plot(x, y)

# Adding title and labels
plt.title('Basic Line Plot')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

# Displaying the plot
plt.show()
```

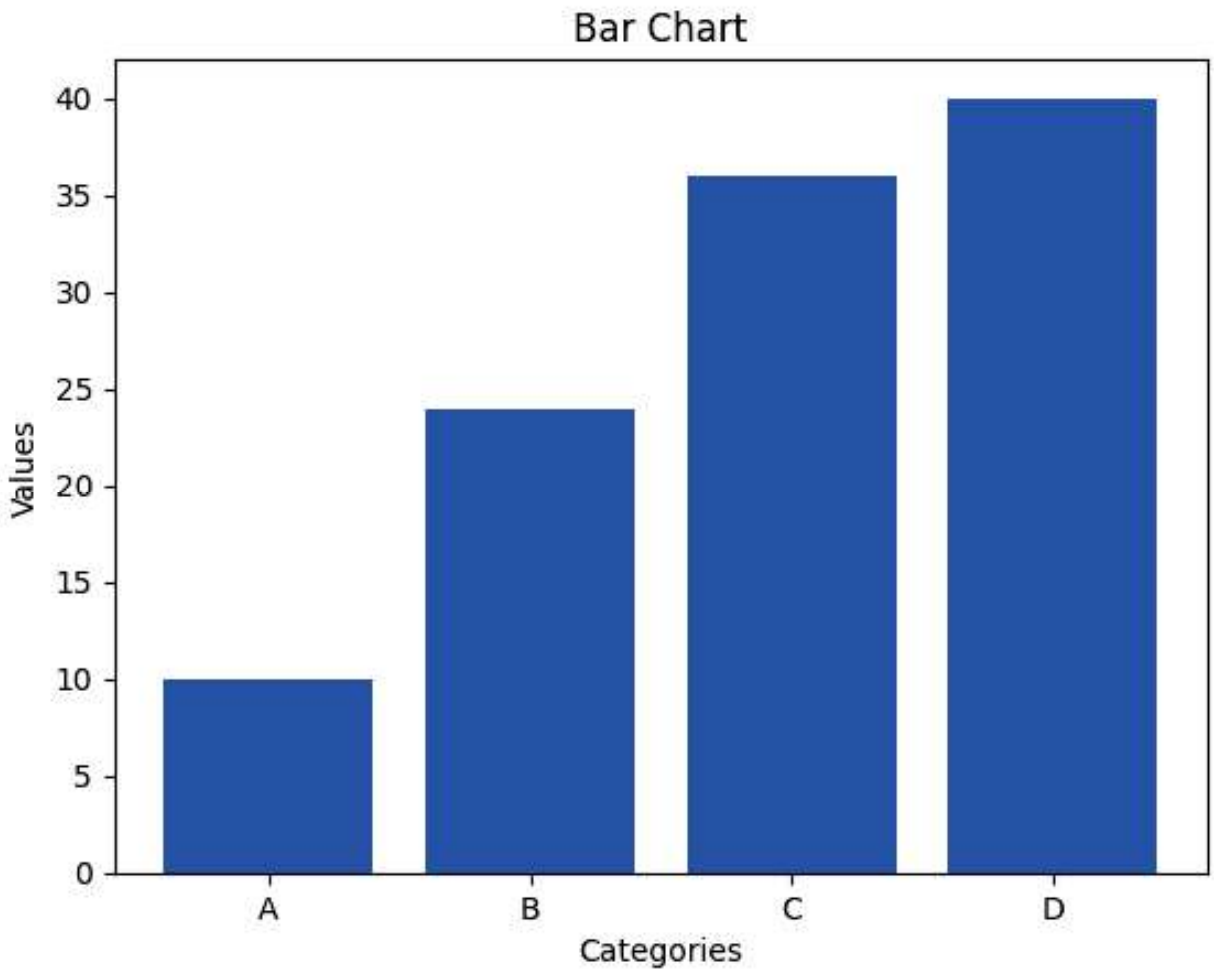
Level 2: Scatter Plot:

```
# Creating data for the scatter plot
x = [1, 2, 3, 4, 5]
y = [5, 7, 4, 6, 9]

# Creating a scatter plot with red dots
plt.scatter(x, y, color='red')

# Adding title and labels
plt.title('Scatter Plot')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

# Displaying the plot
plt.show()
```



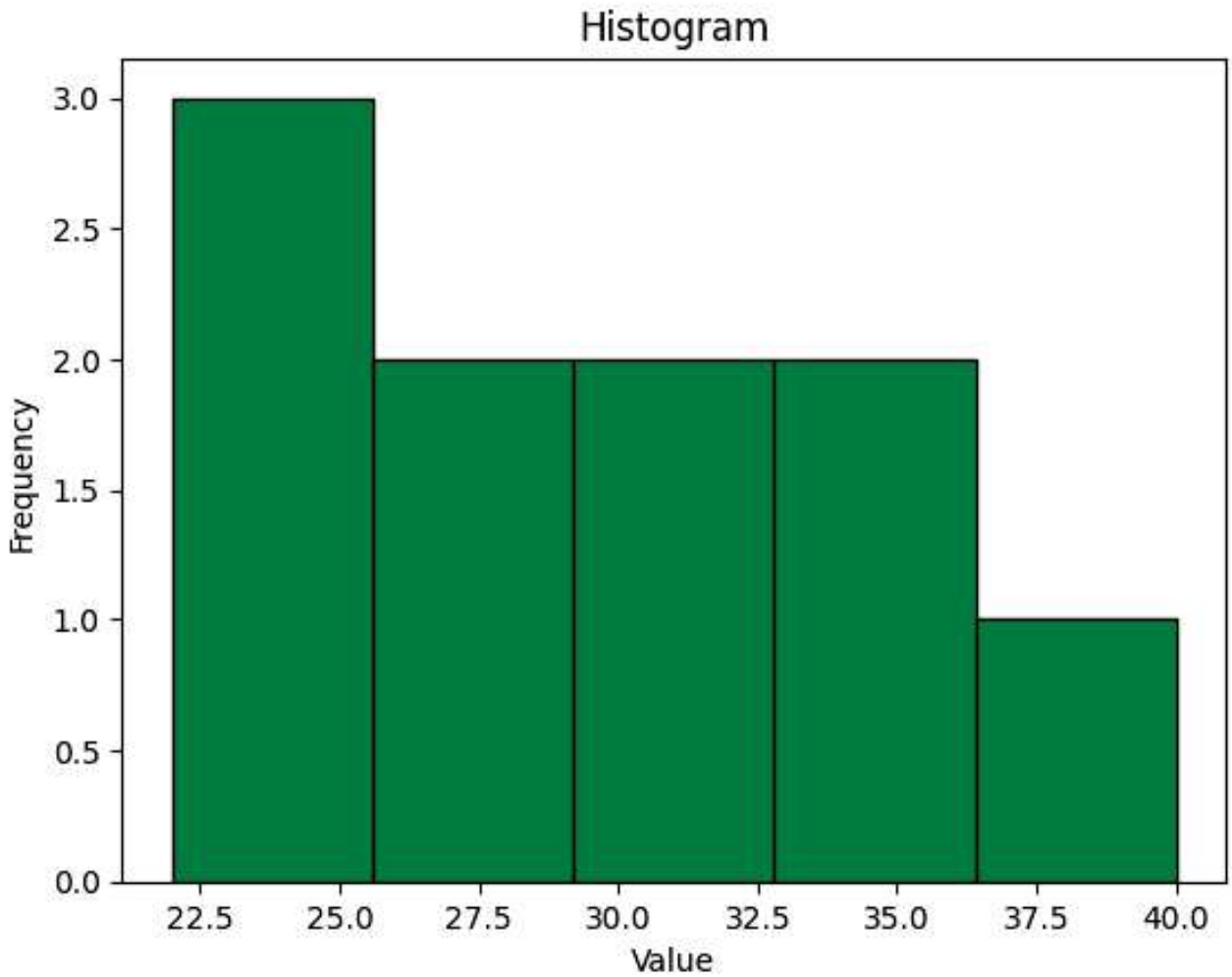
Level 3: Bar Chart:

```python
# Data for the bar chart
labels = ['A', 'B', 'C', 'D']
values = [10, 24, 36, 40]

# Creating a bar chart with blue bars
plt.bar(labels, values, color='blue')

# Adding title and labels
plt.title('Bar Chart')
plt.xlabel('Categories')
plt.ylabel('Values')

# Displaying the plot
plt.show()
```

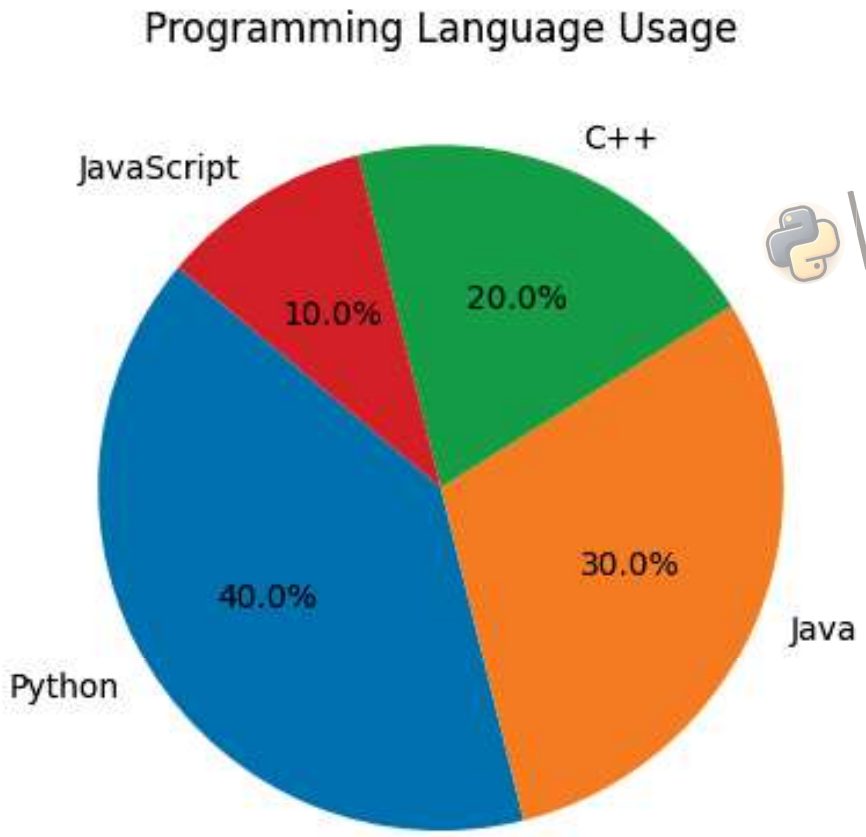
Level 4: Histogram:

```python
# Data for the histogram
data = [22, 25, 25, 26, 28, 30, 32, 35, 35, 40]

# Creating a histogram with 5 bins and black edges
plt.hist(data, bins=5, color='green', edgecolor='black')

# Adding title and labels
plt.title('Histogram')
plt.xlabel('Value')
plt.ylabel('Frequency')

# Displaying the plot
plt.show()
```

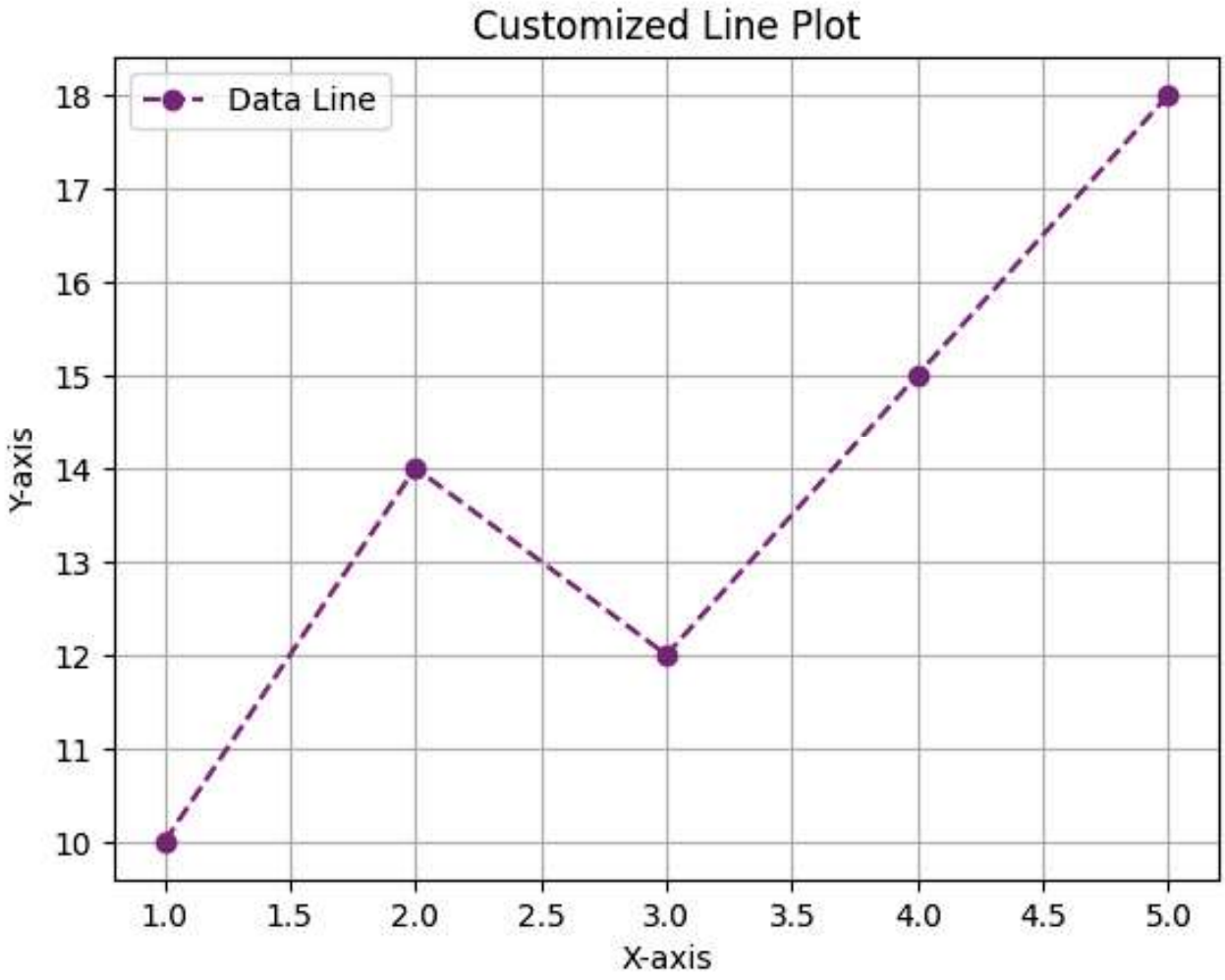
Level 5: Pie Chart:

```
# Data for the pie chart
labels = ['Python', 'Java', 'C++', 'JavaScript']
sizes = [40, 30, 20, 10]

# Creating a pie chart with percentage labels and start angle
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)

# Adding a title
plt.title('Programming Language Usage')

# Displaying the plot
plt.show()
```



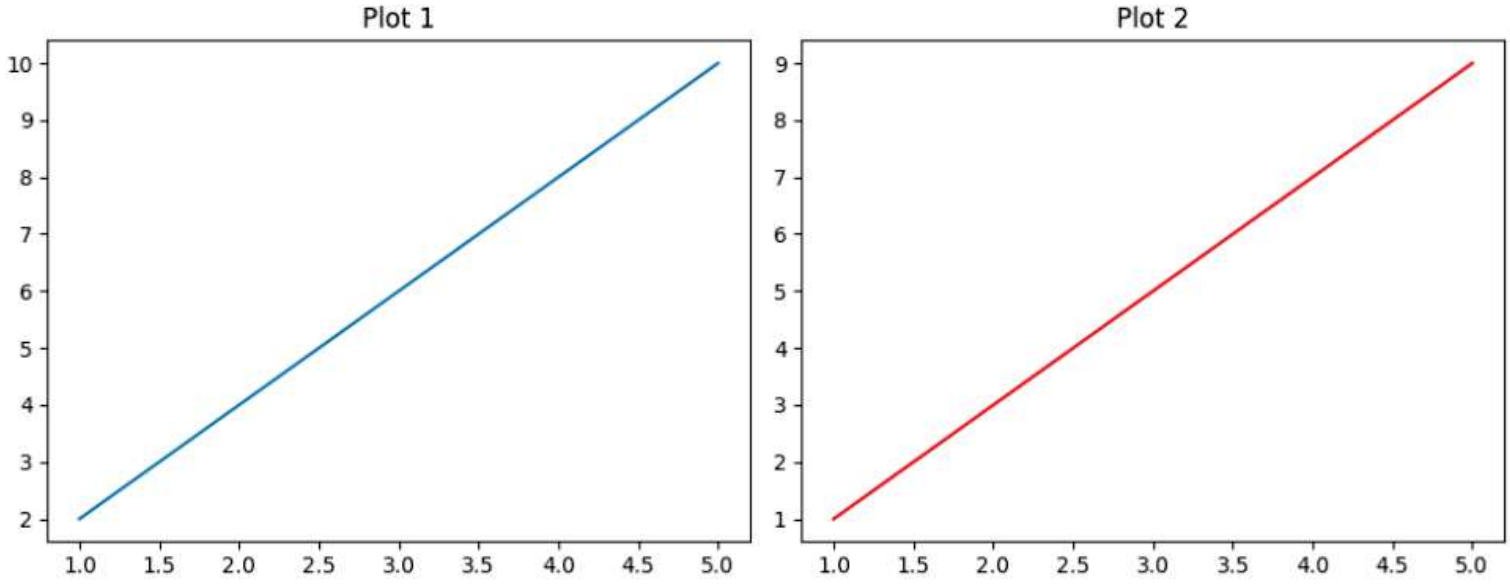
Level 6: Customizing Plots:

```
# Data for the plot
x = [1, 2, 3, 4, 5]
y = [10, 14, 12, 15, 18]

# Creating a customized line plot
plt.plot(x, y, color='purple', linestyle='--', marker='o', label='Data Line')

# Adding title, labels, grid, and legend
plt.title('Customized Line Plot')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.grid(True)
plt.legend()

# Displaying the plot
plt.show()
```

Level 7: Subplots:

```python
# Data for the subplots
x = [1, 2, 3, 4, 5]
y1 = [2, 4, 6, 8, 10]
y2 = [1, 3, 5, 7, 9]

# Creating a figure with 2 subplots side by side
plt.figure(figsize=(10, 4))

# First subplot
plt.subplot(1, 2, 1)
plt.plot(x, y1)
plt.title('Plot 1')

# Second subplot
plt.subplot(1, 2, 2)
plt.plot(x, y2, color='red')
plt.title('Plot 2')

# Adjusting layout to avoid overlap
plt.tight_layout()

# Displaying the plot
plt.show()
```

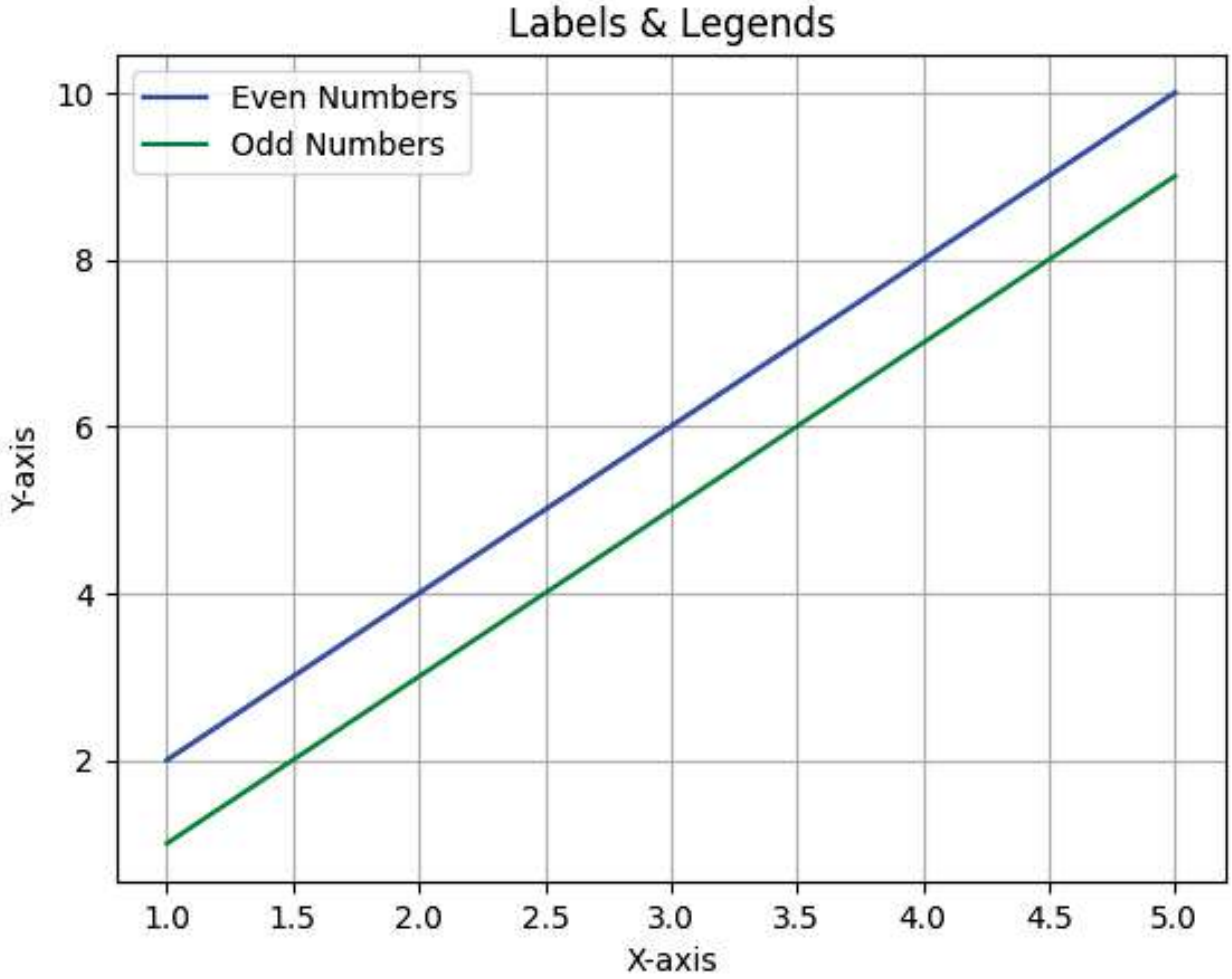



Level 8: Figure Size and DPI:

```
# Setting figure size and DPI for high resolution
plt.figure(figsize=(10, 6), dpi=100)

# Data for the plot
x = [1, 2, 3, 4, 5]
y = [10, 20, 15, 25, 30]

# Creating the plot with custom size
plt.plot(x, y, color='orange', linewidth=2)
plt.title('Figure Size & DPI Example')

# Displaying the plot
plt.show()
```

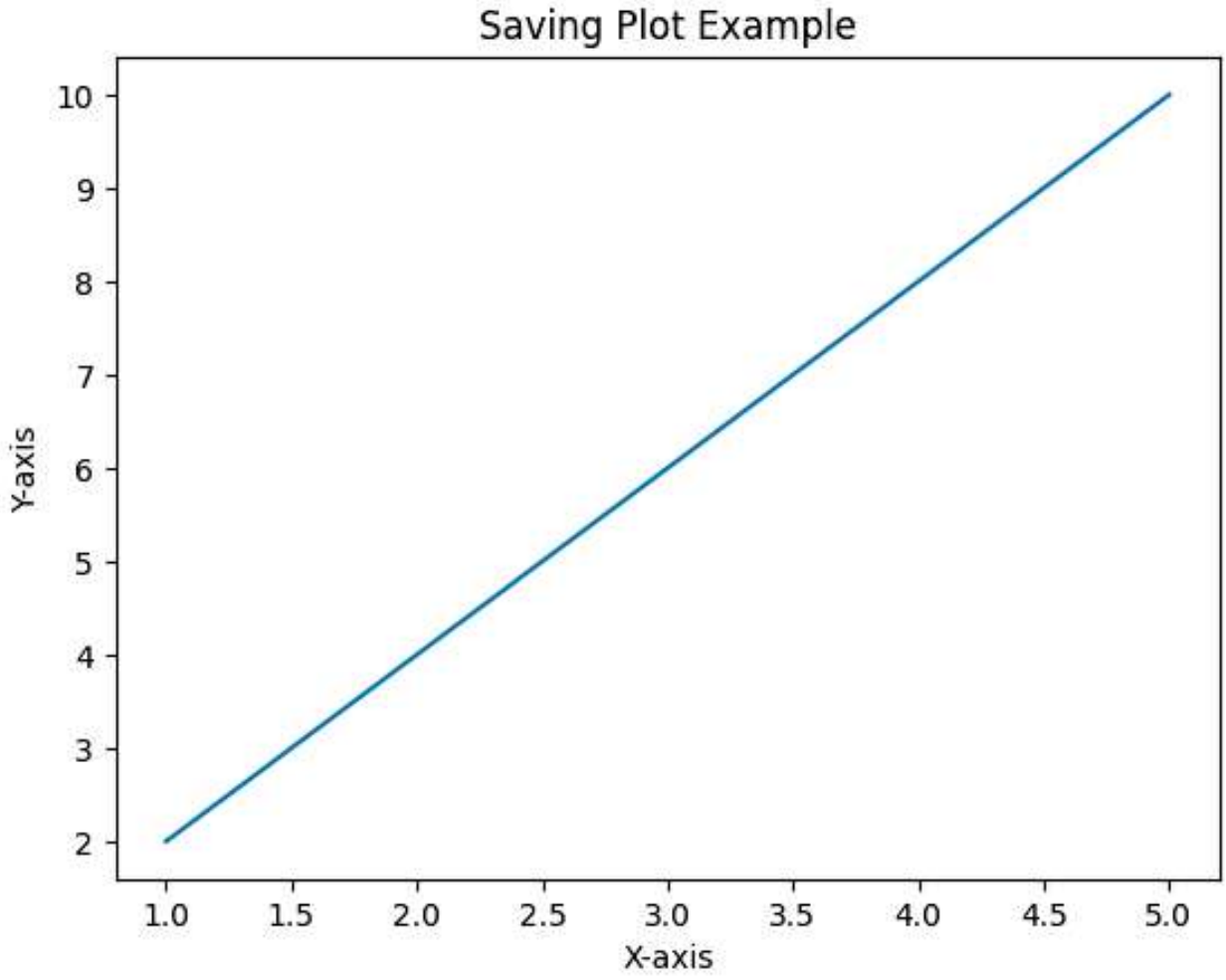

Level 9: Adding Labels and Legends:

```
# Data for plotting
x = [1, 2, 3, 4, 5]
y1 = [2, 4, 6, 8, 10]
y2 = [1, 3, 5, 7, 9]

# Plotting two lines with labels
plt.plot(x, y1, label='Even Numbers', color='blue')
plt.plot(x, y2, label='Odd Numbers', color='green')

# Adding title, labels, legend, and grid
plt.title('Labels & Legends')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend(loc='upper left')
plt.grid(True)

# Displaying the plot
plt.show()
```

Level 10: Saving Plots:

```python
# Data for the plot
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Creating the plot
plt.plot(x, y)
plt.title('Saving Plot Example')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

# Saving the plot as a PNG file with high resolution
plt.savefig('my_plot.png', dpi=300, bbox_inches='tight')

# Displaying the plot
plt.show()
```

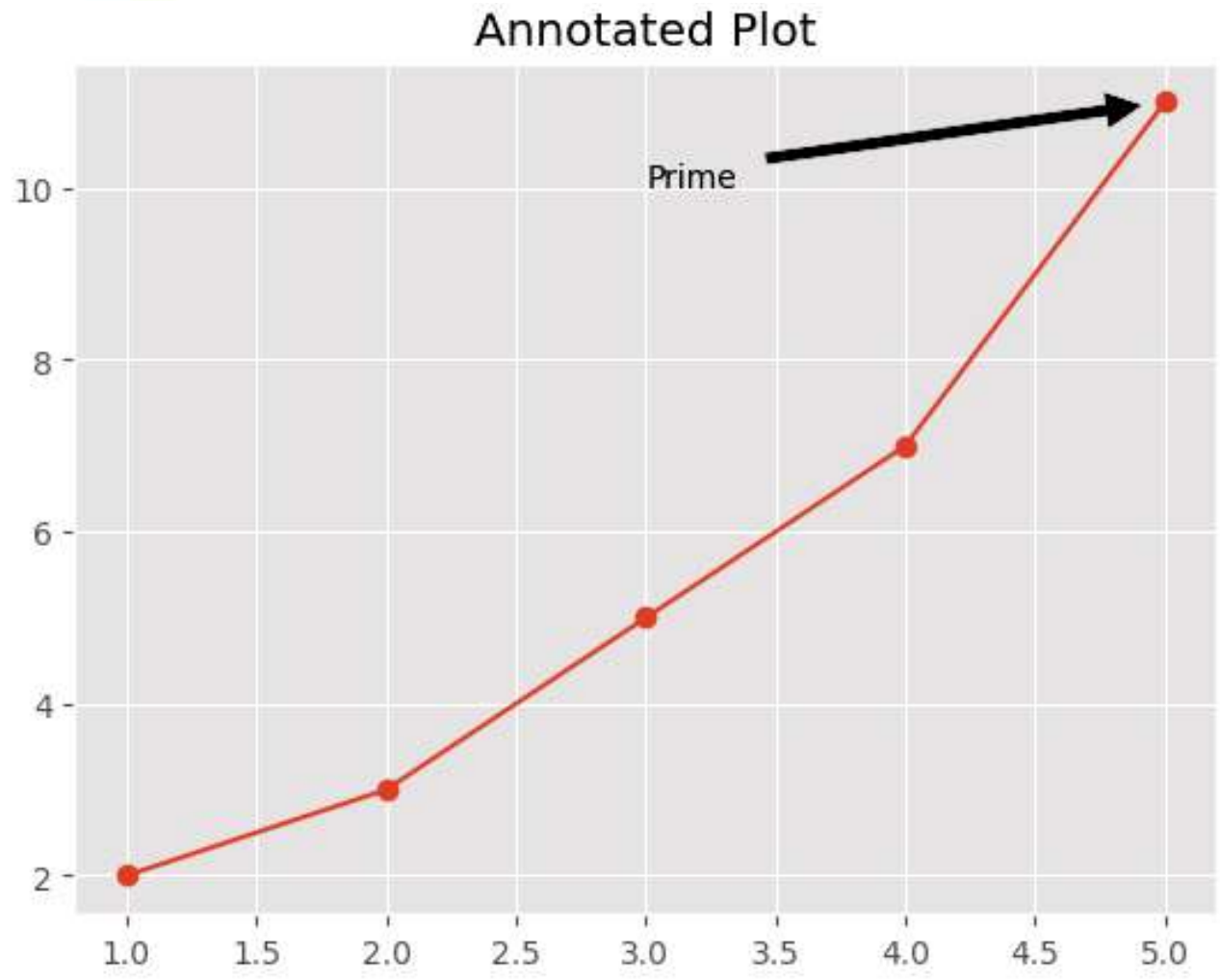



Level 11: Plot Styling:

```
# Setting a built-in style
plt.style.use('ggplot')

# Data for the plot
x = [1, 2, 3, 4, 5]
y = [5, 10, 15, 10, 5]

# Creating the plot with a styled background
plt.plot(x, y, color='orange', linewidth=2)
plt.title('Styled Plot')

# Displaying the plot
plt.show()
```

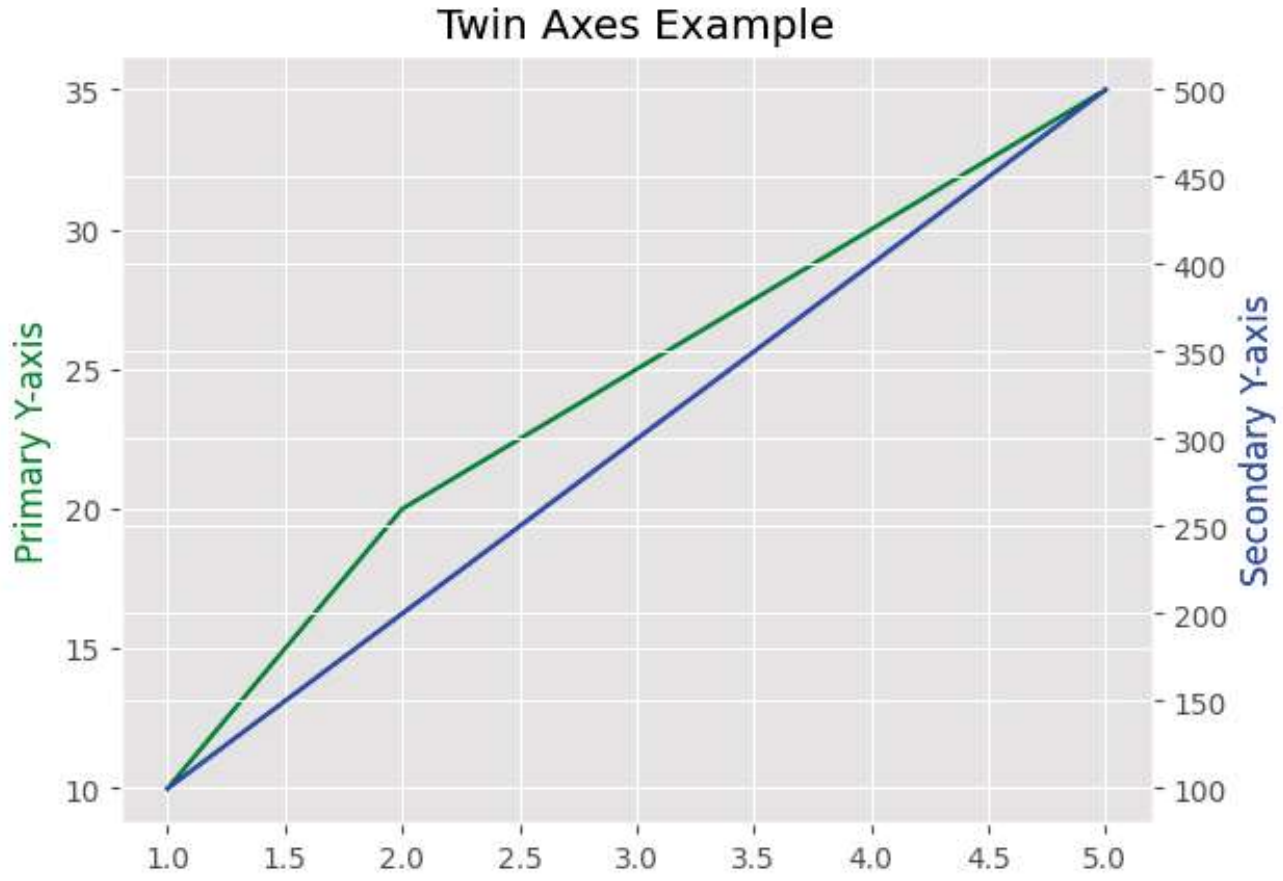

Level 12: Annotations:

```
# Data for the plot
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]

# Creating the plot
plt.plot(x, y, marker='o')

# Adding an annotation with an arrow
plt.annotate('Prime', xy=(5, 11), xytext=(3, 10),
             arrowprops=dict(facecolor='black', shrink=0.05))

# Adding title and displaying the plot
plt.title('Annotated Plot')
plt.show()
```

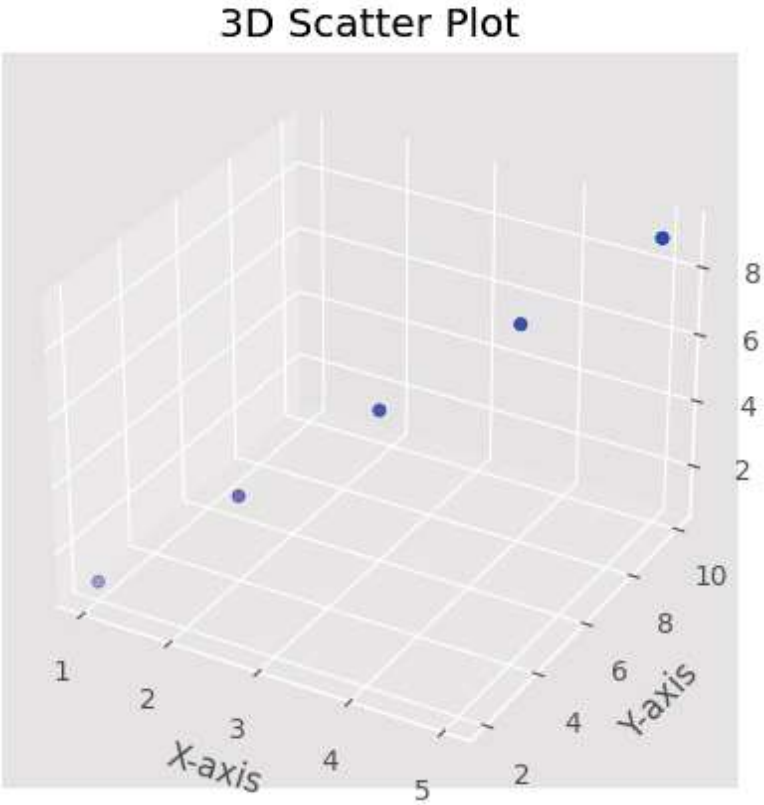
Level 13: Twin Axes (Two Y-axes):

```
# Data for twin axes
x = [1, 2, 3, 4, 5]
y1 = [10, 20, 25, 30, 35]
y2 = [100, 200, 300, 400, 500]

# Creating the figure and primary axis
fig, ax1 = plt.subplots()

# Plotting on the first Y-axis
ax1.plot(x, y1, color='green')
ax1.set_ylabel('Primary Y-axis', color='green')

# Creating a secondary Y-axis
ax2 = ax1.twinx()
ax2.plot(x, y2, color='blue')
ax2.set_ylabel('Secondary Y-axis', color='blue')

# Adding title and displaying the plot
plt.title('Twin Axes Example')
plt.show()
```

Level 14: 3D Plot:

```python
# Importing 3D plotting toolkit
from mpl_toolkits.mplot3d import Axes3D

# Creating a 3D figure
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Data for the 3D plot
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]
z = [1, 3, 5, 7, 9]

# Creating a 3D scatter plot
ax.scatter(x, y, z, color='blue')

# Adding axis labels and title
ax.set_title('3D Scatter Plot')
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')

# Displaying the plot
plt.show()
```

Matplotlib for beginners

Matplotlib is a library for making 2D plots in Python. It is designed with the philosophy that you should be able to create simple plots with just a few commands:

1 Initialize

```
import numpy as np
import matplotlib.pyplot as plt
```

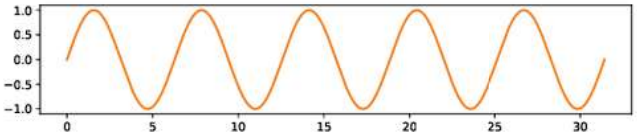
2 Prepare

```
X = np.linspace(0, 10*np.pi, 1000)
Y = np.sin(X)
```

3 Render

```
fig, ax = plt.subplots()
ax.plot(X, Y)
plt.show()
```

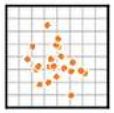
4 Observe



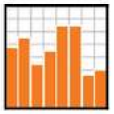
Choose

Matplotlib offers several kind of plots (see Gallery):

```
X = np.random.uniform(0, 1, 100)
Y = np.random.uniform(0, 1, 100)
ax.scatter(X, Y)
```



```
X = np.arange(10)
Y = np.random.uniform(1, 10, 10)
ax.bar(X, Y)
```

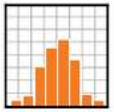


```
Z = np.random.uniform(0, 1, (8, 8))

ax.imshow(Z)
```

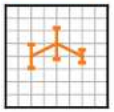
```
Z = np.random.uniform(0, 1, (8, 8))

ax.contourf(Z)
```

```
Z = np.random.uniform(0, 1, 4)

ax.pie(Z)
```

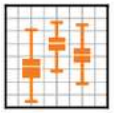
```
Z = np.random.normal(0, 1, 100)

ax.hist(Z)
```



```
X = np.arange(5)
Y = np.random.uniform(0, 1, 5)
ax.errorbar(X, Y, Y/4)
```



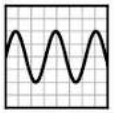
```
Z = np.random.normal(0, 1, (100, 3))

ax.boxplot(Z)
```



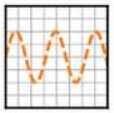
Tweak

You can modify pretty much anything in a plot, including limits, colors, markers, line width and styles, ticks and ticks labels, titles, etc.

```
X = np.linspace(0, 10, 100)
Y = np.sin(X)
ax.plot(X, Y, color="black")
```

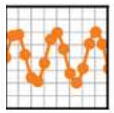


```
X = np.linspace(0, 10, 100)
Y = np.sin(X)
ax.plot(X, Y, linestyle="--")
```



```
X = np.linspace(0, 10, 100)
Y = np.sin(X)
ax.plot(X, Y, linewidth=5)
```

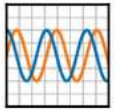
```
X = np.linspace(0, 10, 100)
Y = np.sin(X)
ax.plot(X, Y, marker="o")
```



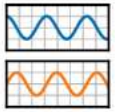
Organize

You can plot several data on the same figure, but you can also split a figure in several subplots (named *Axes*):

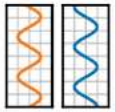
```
X = np.linspace(0, 10, 100)
Y1, Y2 = np.sin(X), np.cos(X)
ax.plot(X, Y1, X, Y2)
```



```
fig, (ax1, ax2) = plt.subplots(2, 1)
ax1.plot(X, Y1, color="C1")
ax2.plot(X, Y2, color="C0")
```

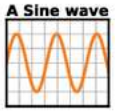


```
fig, (ax1, ax2) = plt.subplots(1, 2)
ax1.plot(Y1, X, color="C1")
ax2.plot(Y2, X, color="C0")
```



Label (everything)

```
ax.plot(X, Y)
fig.suptitle(None)
ax.set_title("A Sine wave")
```




```
ax.plot(X, Y)
ax.set_ylabel(None)
ax.set_xlabel("Time")
```

Explore

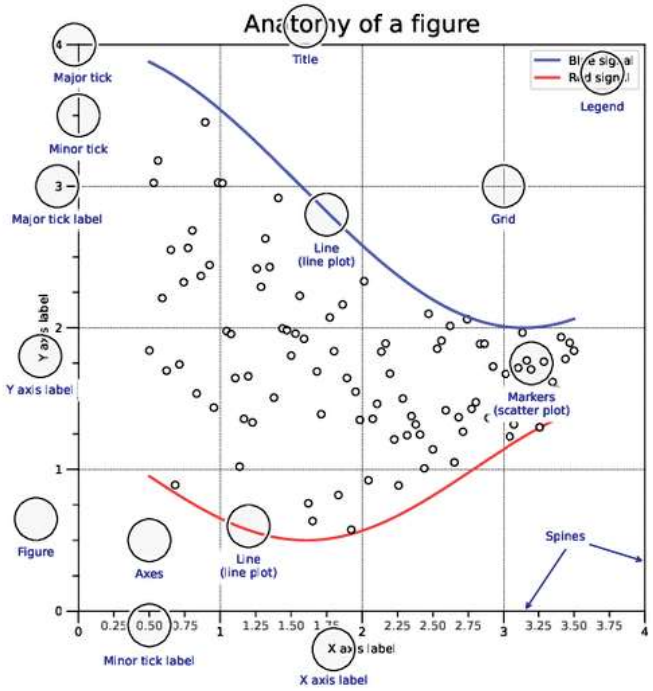
Figures are shown with a graphical user interface that allows to zoom and pan the figure, to navigate between the different views and to show the value under the mouse.

Save (bitmap or vector format)

```
fig.savefig("my-first-figure.png", dpi=300)
fig.savefig("my-first-figure.pdf")
```

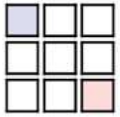
Matplotlib for intermediate users

A matplotlib figure is composed of a hierarchy of elements that forms the actual figure. Each element can be modified.

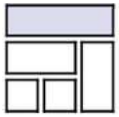


Figure, axes & spines

```
fig, axs = plt.subplots(3, 3)
axs[0, 0].set_facecolor("#ddddff")
axs[2, 2].set_facecolor("#ffffdd")
```



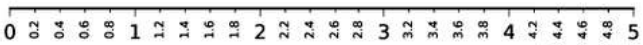
```
gs = fig.add_gridspec(3, 3)
ax = fig.add_subplot(gs[0, :])
ax.set_facecolor("#ddddff")
```



```
fig, ax = plt.subplots()
ax.spines["top"].set_color("None")
ax.spines["right"].set_color("None")
```

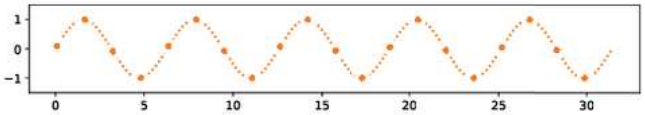
Ticks & labels

```
from mpl.ticker import MultipleLocator as ML
from mpl.ticker import ScalarFormatter as SF
ax.xaxis.set_minor_locator(ML(0.2))
ax.xaxis.set_minor_formatter(SF())
ax.tick_params(axis='x',which='minor',rotation=90)
```



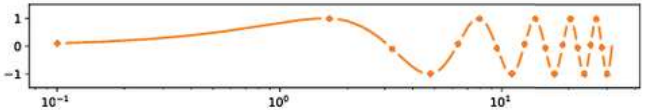
Lines & markers

```
X = np.linspace(0.1, 10*np.pi, 1000)
Y = np.sin(X)
ax.plot(X, Y, "C1o:", markevery=50, mec="1.0")
```



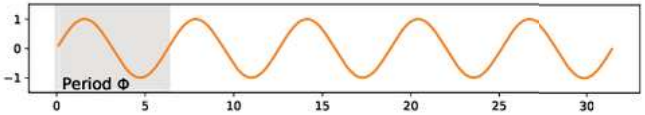
Scales & projections

```
fig, ax = plt.subplots()
ax.set_xscale("log")
ax.plot(X, Y, "C1o-", markevery=50, mec="1.0")
```



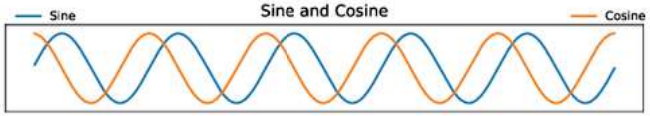
Text & ornaments

```
ax.fill_betweenx([-1, 1], [0], [2*np.pi])
ax.text(0, -1, r" Period $\Phi$")
```



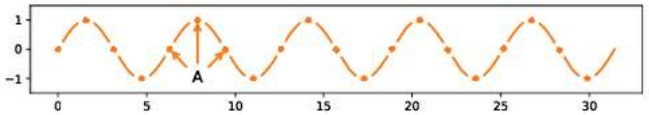
Legend

```
ax.plot(X, np.sin(X), "C0", label="Sine")
ax.plot(X, np.cos(X), "C1", label="Cosine")
ax.legend(bbox_to_anchor=(0,1,1,.1), ncol=2,
          mode="expand", loc="lower left")
```



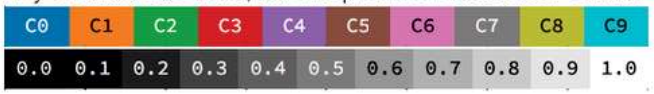
Annotation

```
ax.annotate("A", (X[250],Y[250]), (X[250],-1),
  ha="center", va="center", arrowprops={
    "arrowstyle": "->", "color": "C1"})
```



Colors

Any color can be used, but Matplotlib offers sets of colors:



Size & DPI

Consider a square figure to be included in a two-column A4 paper with 2 cm margins on each side and a column separation of 1 cm. The width of a figure is (21 - 2*2 - 1)/2 = 8 cm. One inch being 2.54 cm, figure size should be 3.15×3.15 in.

```
fig = plt.figure(figsize=(3.15, 3.15), dpi=50)
plt.savefig("figure.pdf", dpi=600)
```

ts