Matplotlib Quick Reference

Figures, axes, plots, and customization

Basic Plots

Line Plot

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [2, 4, 1, 8, 3]
plt.plot(x, y)
plt.show()
```

Quick Plot Shortcuts

```
plt.plot(y)              # x auto 0..N-1
plt.plot(x, y, "ro--")  # red circles, dashed
plt.plot(x, y, "bs-")   # blue squares, solid
```

Format String Codes

`r` `g` `b` `k`	Red, green, blue, black
`o` `s` `^` `D`	Circle, square, triangle, diamond markers
`-` `--` `-.` `:`	Solid, dashed, dash-dot, dotted lines

Subplots

Figure and Axes

```
fig, ax = plt.subplots()
ax.plot(x, y)
ax.set_title("Single Plot")
plt.show()
```

Grid of Subplots

```
fig, axes = plt.subplots(2, 2, figsize=(8, 6))
axes[0, 0].plot(x, y)
axes[0, 1].bar(x, y)
axes[1, 0].scatter(x, y)
fig.tight_layout()
```

Shared Axes

```
fig, (ax1, ax2) = plt.subplots(1, 2,
    sharey=True, figsize=(10, 4))
ax1.plot(x, y)
ax2.plot(x, y2)
```

Labels & Titles

Axis Labels and Title

```
plt.plot(x, y)
plt.xlabel("Time (s)")
plt.ylabel("Value")
plt.title("Sensor Reading")
plt.show()
```

OO-Style Labels

```
fig, ax = plt.subplots()
ax.plot(x, y)
ax.set_xlabel("X"); ax.set_ylabel("Y")
ax.set_title("My Plot")
```

Annotations

```
ax.annotate("Peak", xy=(4, 8),
    xytext=(3, 9),
    arrowprops=dict(arrowstyle="->"))
```

Customization

Colors and Styles

```
plt.plot(x, y, color="#FF5733",
    linewidth=2, linestyle="--")
plt.plot(x, y2, color="steelblue",
    marker="o", markersize=5)
```

Figure Size and DPI

```
fig, ax = plt.subplots(figsize=(10, 6), dpi=100)
plt.rcParams["figure.figsize"] = (8, 5)
```

Style Sheets

```
print(plt.style.available)  # list all
plt.style.use("seaborn-v0_8")
plt.style.use("ggplot")
```

Bar & Histogram

Bar Chart

```
labels = ["A", "B", "C", "D"]
values = [23, 45, 12, 37]
plt.bar(labels, values, color="teal")
plt.show()
```

Grouped / Stacked Bars

```
import numpy as np
x = np.arange(4); w = 0.35
plt.bar(x - w/2, v1, w, label="2024")
plt.bar(x + w/2, v2, w, label="2025")
plt.xticks(x, labels)
```

Histogram

```
data = np.random.randn(1000)
plt.hist(data, bins=30, edgecolor="black",
    alpha=0.7)
plt.show()
```

Scatter & Line

Scatter Plot

```
plt.scatter(x, y, c="red", s=50,
    alpha=0.6, edgecolors="black")
plt.show()
```

Scatter with Colormap

```
sc = plt.scatter(x, y, c=values,
    cmap="viridis", s=sizes)
plt.colorbar(sc, label="Intensity")
```

Multiple Lines

```
plt.plot(x, y1, label="Train")
plt.plot(x, y2, label="Validation")
plt.legend()
plt.show()
```

Axes & Ticks

Axis Limits and Scale

```
ax.set_xlim(0, 10)
ax.set_ylim(-1, 1)
ax.set_xscale("log")
ax.set_yscale("log")
```

Custom Ticks

```
ax.set_xticks([0, 1, 2, 3, 4])
ax.set_xticklabels(["Mon", "Tue", "Wed",
    "Thu", "Fri"], rotation=45)
```

Grid

```
ax.grid(True, linestyle="--", alpha=0.5)
ax.grid(axis="y")  # horizontal only
```

Legends

Adding Legends

```
ax.plot(x, y, label="Series A")
ax.plot(x, y2, label="Series B")
ax.legend(loc="upper right")
```

Legend Placement

``'best'``	Auto best position (default)
``'upper left'``	Top-left corner
``'lower right'``	Bottom-right corner
``'center'``	Center of axes
`bbox_to_anchor=(1, 1)`	Place outside the axes area

Legend Customization

```
ax.legend(fontsize=8, frameon=False,
    ncol=2, title="Legend")
```

Saving

Save to File

```
plt.savefig("plot.png", dpi=300,
    bbox_inches="tight")
plt.savefig("plot.pdf")
plt.savefig("plot.svg", transparent=True)
```

Supported Formats

PNG	Raster, best for web/screen
PDF	Vector, best for print/papers
SVG	Vector, scalable for web
EPS	Vector, legacy scientific journals

Save from Figure Object

```
fig, ax = plt.subplots()
ax.plot(x, y)
fig.savefig("output.png", dpi=150,
    facecolor="white")
```

Common Patterns

Twin Axes (Two Y-Scales)

```
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
ax1.plot(x, temp, "r-", label="Temp")
ax2.plot(x, pressure, "b-", label="Pressure")
```

Fill Between

```
ax.fill_between(x, y_low, y_high,
    alpha=0.3, color="blue")
```

Heatmap with imshow

```
data = np.random.rand(10, 10)
plt.imshow(data, cmap="hot",
    interpolation="nearest")
plt.colorbar()
```

Pie Chart

```
plt.pie(sizes, labels=labels,
    autopct="%1.1f%%", startangle=90)
plt.axis("equal")
```