Matplotlib Cheat Sheet

Setup Imports

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

For Jupyter Notebooks:

```
%matplotlib inline
```

Show plot:

```
plt.show()
```

Two Plotting Methods

1. Pyplot (Simple, Quick):

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

2. Object-Oriented (More Control):

```
fig, ax = plt.subplots()
ax.plot(x, y)
```

Recommended for complex plots!

Basic Plot Types

Line Plot:

```
ax.plot(x, y)
```

Scatter Plot:

```
ax.scatter(x, y, c=colors, cmap='winter')
```

Bar Chart (Vertical):

```
ax.bar(x, height)
```

Bar Chart (Horizontal):

```
ax.barh(y, width)
```

Histogram:

```
ax.hist(data, bins=20, edgecolor='black')
```

bins = number of bars (integer, not string!)

Customization

Using .set() - Multiple at once:

```
ax.set(title="My Title",
xlabel="X Label",
ylabel="Y Label")
```

Individual methods:

```
ax.set_title("Title", fontsize=14)
ax.set_xlabel("X axis")
ax.set_ylabel("Y axis")
```

Add grid:

```
ax.grid(True, alpha=0.3)
```

Add horizontal line:

```
ax.axhline(y=value, linestyle='--', color='red')
```

Figure Subplots

Single plot:

```
fig, ax = plt.subplots(figsize=(10, 6))
```

Multiple subplots (2 rows, 2 cols):

```
fig, ((ax1, ax2), (ax3, ax4)) = \
plt.subplots(nrows=2, ncols=2,
figsize=(10, 8))
```

Access subplots:

```
ax1.plot(x, y) # Top-left
ax2.scatter(x, y) # Top-right
```

fig = canvas, ax = individual plot area

Legend Colors

Add legend (must use label=):

```
ax.plot(x, y, label='Line 1')
ax.legend(loc='upper right')
```

For scatter with colors:

```
scatter = ax.scatter(x, y, c=colors)
ax.legend(*scatter.legend_elements(),
title="Categories")
```

Color parameters:

```
c='red' # Single color
cmap='winter' # Color map for gradients
```

Plotting with Pandas

Direct from DataFrame:

```
df.plot(x='col1', y='col2')
df['col'].plot(kind='hist', bins=50)
```

Different plot types:

```
df.plot.line() # Line plot
df.plot.scatter(x='a', y='b')
df.plot.bar() # Bar chart
df.plot.hist() # Histogram
```

Multiple histograms:

```
df.plot.hist(subplots=True, figsize=(10,30))
```

⚠ Data must be numeric to plot!

Styles Themes

Check available styles:

```
plt.style.available
```

Apply a style:

```
plt.style.use('ggplot')
plt.style.use('seaborn-v0_8-whitegrid')
```

Popular styles:

```
'ggplot', 'fivethirtyeight', 'bmh'
'seaborn-v0_8-whitegrid', 'classic'
```

⚠ Old 'seaborn-whitegrid' → now 'seaborn-v0_8-whitegrid' in newer versions

Saving Display

Save figure:

```
fig.savefig("plot.png")
fig.savefig("plot.pdf", dpi=300)
```

Adjust layout:

```
plt.tight_layout() # Prevent overlap
```

Reset plot:

```
fig, ax = plt.subplots() # Creates new
```

Common Parameters

```
figsize=(10, 6) # Width, height
color='red' # or c='red'
alpha=0.7 # Transparency
linewidth=2 # Line thickness
linestyle='--' # Dashed line
marker='o' # Point style
```

Scatter Plot Special Features

```
ax.scatter(x, y,
c=colors, # Color by variable
s=sizes, # Size of points
cmap='viridis', # Color map
alpha=0.6, # Transparency
edgecolors='black' # Border
)
```

c= colors each point by value (3rd dimension!)

Quick Tips

- Use `fig, ax = plt.subplots()` for better control (OO method)
- Always assign scatter to variable if using `.legend_elements()`
- `bins` must be integer, not string!
- Use `ax.set()` to set multiple properties at once

Complete Workflow Example

```
# 1. Import libraries
import matplotlib.pyplot as plt
import numpy as np

# 2. Prepare data
x = np.linspace(0, 10, 100)
y = x ** 2

# 3. Create figure and axes
fig, ax = plt.subplots(figsize=(10, 6))

# 4. Plot data
ax.plot(x, y, label='y = x²', color='blue', linewidth=2)

# 5. Customize
ax.set(title='Quadratic Function', xlabel='X', ylabel='Y')
ax.legend(loc='upper left')
ax.grid(True, alpha=0.3)

# 6. Save and display
```