Matplotlib Cheat Sheet: Beginner to Advanced

1. Getting Started

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

2. Basic Plot Types

Task	Example	Explanation
Line plot	`plt.plot(, )`	x vs y line (default plot type)
Scatter plot	`plt.scatter(x, y)`	Dots for each value pair
Bar chart	`plt.bar(['A','B'], )`	Categorical x, height y
Histogram	`plt.hist(data, bins=20)`	Count values into bins
Pie chart	`plt.pie()`	Show parts of a whole

3. Displaying and Saving Figures

Task	Example	Explanation
Show plot	`plt.show()`	Opens window or in-notebook display
Save plot	`plt.savefig('fig.png', dpi=300)`	Save to file
Clear figure	`plt.clf()`	Reset figure, if reusing in a loop

4. Figure & Axis Management

Task	Example	Explanation
Create figure	`fig = plt.figure(figsize=(6,3))`	Custom size (inches)
Add subplot(s)	`ax = fig.add_subplot(1,2,1)`	Row=1, Col=2, Index=1
Simple subplots	`fig, axs = plt.subplots(2,2)`	Grid of axes in one call
Plot on ax	`ax.plot(x, y)`	Draw on specific subplot

5. Customizing Appearance

Task	Example	Explanation
Title	`plt.title('My Title')`	Chart title
Ax labels	`plt.xlabel('x label'),plt.ylabel('y label')`	Set x/y axis names
Legend	`plt.legend(['Series1']),ax.legend(loc='upper left')`	Show legend
Axis limits	`plt.xlim(0,10),plt.ylim(-1,1)`	Force axes bounds
Grid	`plt.grid(True, linestyle='--')`	Add grid for readability
Ticks & labels	`plt.xticks(, ['zero','five','ten'])`	Custom ticks and labels
Colors/styles	`plt.plot(x, y, 'r--', linewidth=2)`	Red dashed line, thick
Alpha	`plt.scatter(x, y, alpha=0.6)`	Transparency [0=transparent, 1=opaque]

6. Annotations & Text

Task	Example	Explanation
Add text	`plt.text(2, 0.5, 'hello', fontsize=12, color='blue')`	Add label at (x,y)
Annotate	`plt.annotate('peak', xy=(x0,y0), xytext=(x0+1, y0+1), arrowprops={'arrowstyle':'->'})`	Arrowed annotation

7. Advanced Plotting

Multiple Plots

```
fig, axs = plt.subplots(2, 3, figsize=(12,5))
axs[0,0].plot(x, y)
axs[1,2].hist(data)
plt.tight_layout()
```

Multiple Y-axes

```
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
ax1.plot(t, temp, 'g-')
ax2.plot(t, humidity, 'b-')
```

Log, Polar, 3D Plots

```
plt.semilogy(x, y)                      # Logarithmic y-axis
ax = plt.subplot(projection='polar')     # Polar axes
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(); ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z)
```

8. Customizing Ticks & Scales

```
ax.set_xticks([0,2,4,6])
ax.set_yticklabels(['low','med','high'])
ax.set_xscale('log')
ax.tick_params(axis='x', rotation=45)
```

9. Colormaps & Images

Task	Example	Explanation
Show image	`plt.imshow(np_img, cmap='gray')`	2D array as image; choose color map
Colorbar	`plt.colorbar()`	Add scale legend for values
Contour plot	`plt.contour(X, Y, Z, levels=20)`	Draw contours in 2D/heatmap

10. Styling Themes & Fonts

Task	Example	Explanation
Use style	`plt.style.use('seaborn-v0_8-darkgrid')`	Prebuilt style/themes
Custom font	`plt.rcParams['font.size'] = 14`	Change global font
Change colors globally	`plt.rcParams['axes.prop_cycle'] = plt.cycler(color=['r','g','b'])`	Default color cycle

11. Inset Plots & Zooms

```
from mpl_toolkits.axes_grid1.inset_locator import inset_axes
ax = plt.gca()
ax_inset = inset_axes(ax, width='30%', height='30%', loc='upper right')
ax_inset.plot(zoom_x, zoom_y)
```

12. Interactive Elements (Jupyter/advanced)

- `%matplotlib inline` or `%matplotlib notebook` for interactivity in Jupyter.
- Use widgets for sliders/real-time control: `from ipywidgets import interact`

13. Exporting for Publication

Task	Example	Explanation
Save as PDF/SVG	`plt.savefig('figure.pdf')`	Vector graphic, publication ready
Set DPI	`plt.savefig('fig.png', dpi=300)`	High-res bitmap for print
Transparent background	`plt.savefig('chart.png', transparent=True)`	Overlay charts on backgrounds

14. Professional Tips

- Call `plt.tight_layout()` after complex subplot grids for best spacing.
- Use `ax.set()` to change multiple axis properties in one line:
  `ax.set(title='...', xlabel='...', ylabel='...')`
- Avoid clutter: remove chartjunk with `ax.spines['top'].set_visible(False)` etc.
- For complex plots: prefer the object-oriented API over the `plt` state machine.

15. Example: End-to-End Plot

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

x = np.linspace(0, 10, 200)
y = np.sin(x)
```

```
fig, ax = plt.subplots(figsize=(8, 4))
ax.plot(x, y, label='Sine', color='darkred', linewidth=2)
ax.set(title='Sine Curve', xlabel='X', ylabel='sin(X)')
ax.grid(True, linestyle=":")
ax.legend(loc='upper right')
plt.tight_layout()
plt.savefig('sine-curve.pdf', dpi=200)
plt.show()
```

For more themes, check:

- `plt.style.available`

  To learn interactively:

- Use Jupyter Notebooks and experiment with `%matplotlib inline`.