MATPLOTLIB COMPLETE CHEATSHEET (LINE-BY-LINE)

IMPORT
import matplotlib.pyplot as plt

BASIC PLOT
plt.plot()
plt.scatter()
plt.bar()
plt.barh()
plt.hist()
plt.boxplot()
plt.pie()
plt.stem()
plt.step()
plt.fill()
plt.fill_between()

FIGURE & AXES CREATION
plt.figure()
plt.subplots()
fig.add_subplot()
fig, ax = plt.subplots()

SHOW & SAVE
plt.show()
plt.savefig()
plt.close()

LABELS
plt.xlabel()

plt.ylabel()
plt.title()
plt.suptitle()
plt.legend()
plt.colorbar()

AXIS CONTROL
plt.xlim()
plt.ylim()
plt.xticks()
plt.yticks()
plt.tick_params()
plt.axis()
plt.gca()
plt.gcf()

STYLE
plt.style.use()
plt.grid()
plt.tight_layout()

COLORS
color=
cmap=
plt.colormaps()
plt.set_cmap()

LINE CONTROL
linewidth=
linestyle=
marker=
markersize=
alpha=

SCATTER CONTROL

plt.scatter(..., s=)
plt.scatter(..., c=)
plt.scatter(..., marker=)

BAR CONTROL
plt.bar(..., width=)
plt.barh(..., height=)
plt.bar_label()

HISTOGRAM CONTROL
plt.hist(..., bins=)
plt.hist(..., density=True)
plt.hist(..., range=)

SUBPLOTS
plt.subplot()
plt.subplots_adjust()
fig, axes = plt.subplots()
axes[i].plot()
axes[i,j].scatter()

ANNOTATIONS
plt.annotate()
plt.text()
plt.axvline()
plt.axhline()
plt.axvspan()
plt.axhspan()

IMAGES
plt.imshow()
plt.matshow()
plt.pcolormesh()
plt.contour()
plt.contourf()

3D PLOTTING (mpl_toolkits.mplot3d)

```
ax = fig.add_subplot(projection='3d')
ax.plot3D()
ax.scatter3D()
ax.plot_surface()
ax.plot_wireframe()
ax.contour3D()
```

EVENTS & INTERACTION

```
plt.ginput()
fig.canvas.mpl_connect()
```

TICKS & FORMAT

```
plt.tick_params()
plt.ticklabel_format()
plt.minorticks_on()
plt.minorticks_off()
```

LEGEND CONTROL

```
plt.legend()
plt.legend(loc= )
plt.legend(title= )
plt.legend(frameon=False)
```

AXES MODES

```
plt.semilogx()
plt.semilogy()
plt.loglog()
plt.xscale('log')
plt.yscale('log')
```

FIGURE CONTROL

```
plt.figure(figsize= )
plt.figure(dpi= )
```

fig.set_size_inches()
fig.set_dpi()

LAYOUTS
plt.tight_layout()
plt.subplots_adjust()
fig.set_tight_layout(True)

PATCHES (Shapes)
from matplotlib.patches import Circle, Rectangle, Polygon
ax.add_patch()

DATE PLOTTING
plt.plot_date()
plt.gcf().autofmt_xdate()

STATISTICAL PLOTS
plt.violinplot()
plt.boxplot()
plt.errorbar()
plt.hexbin()

COLOR MAPS
plt.colormaps()
plt.get_cmap()
plt.set_cmap()

ADVANCED CUSTOMIZATION
plt.rcParams
plt.rc()
plt.rcParams.update()

EXPORT OPTIONS
plt.savefig('plot.png')
plt.savefig('plot.pdf')

plt.savefig('plot.svg')
plt.savefig('plot.jpeg')

🔥 Matplotlib + NumPy Combined Cheatsheet (No Repetition, Only Combined Concepts)

🎯 1. Plotting Directly From NumPy Arrays

NumPy arrays plug directly into Matplotlib.
plt.plot(np_array)
plt.scatter(x_np, y_np)
plt.bar(x_np, height_np)
plt.hist(np_data)
plt.imshow(np_image_array)

🎯 2. Creating Data for Plots Using NumPy

NumPy is used to generate mathematical sequences and grids.
x = np.linspace() → for continuous curves
x = np.arange() → for discrete plots
xx, yy = np.meshgrid() → for 2D surfaces
np.random.randn() → scatter noise
np.sin(x), np.cos(x) → function plotting

🎯 3. Mathematical Curves (Common Pattern)

Use NumPy to compute function values, Matplotlib to render.
y = f(x)
plt.plot(x, y)
Example patterns:
x = np.linspace(-5,5,500)
y = np.exp(-x**2)
y = np.tan(x)
y = np.sin(5*x) * np.cos(3*x)

🎯 4. Generating Synthetic Data for Visualizations

ML/DS often needs synthetic NumPy data to feed plots.
np.random.normal() → histograms
np.random.uniform() → scatter distributions
np.random.multivariate_normal() → clustering visuals
np.random.randint() → categorical plots
np.linspace() → time-series axis
Scatter example pattern:
x = np.random.randn(1000)
y = 3*x + noise

🎯 5. Image Data (NumPy ↔ Matplotlib)

Matplotlib treats images as NumPy arrays.
plt.imshow(np_array_2D) → grayscale
plt.imshow(np_array_3D) → RGB
Modifying images:
np_array[::-1] → flip
np.rot90() → rotate
np.clip() → brightness adjust
np.mean() → convert to grayscale manually

🎯 6. Using NumPy Broadcasting for Visual Structures

Patterns drawn using NumPy math + Matplotlib plot.
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)
plt.contourf(X, Y, Z)
Useful for:
- heatmaps
- contour maps
- surface plots
- wave simulations
- Gaussian blobs

🎯 7. Vectorized Line Generation

Make many lines at once using NumPy 2D arrays.

ys = np.sin(x + np.arange(5)[:, None])

plt.plot(x, ys.T)

Used for:

- multiple experiments
- ensemble lines
- neural network weight evolutions

🎯 8. 3D Plotting With NumPy Meshgrids

NumPy generates the 3D numeric domain.

Matplotlib renders it.

X, Y = np.meshgrid(...)

Z = f(X,Y)

Then:

ax.plot_surface(X, Y, Z)

ax.contour3D(X, Y, Z)

ax.plot_wireframe(X, Y, Z)

🎯 9. Time-Series Visualization

NumPy is used to generate numeric time axes.

t = np.arange()

t = np.linspace()

signal = np.sin(2πft)

Then plot:

plt.plot(t, signal)

🎯 10. Statistical Visualization (NumPy provides stats)

Matplotlib displays → NumPy computes.

NumPy side:

np.mean()

np.std()

np.percentile()

np.histogram()

Matplotlib side:
plt.errorbar()
plt.boxplot()
plt.violinplot()
plt.bar()

🎯 11. Shapes / Geometry Using NumPy

Vectorized shapes rendered with Matplotlib.
θ = np.linspace(0, 2π)
x = r*np.cos(ϑ)*
*y = r*np.sin(θ)
plt.plot(x,y)
Used for:
• circles
• ellipses
• spirals
• polygons

🎯 12. Heatmaps & Grids

Generate values with NumPy, visualize with Matplotlib.
Z = np.random.rand(50,50)
plt.imshow(Z)
plt.colorbar()
Or deterministic:
Z = X**2 + Y**2

🎯 13. Histogram Calculation (NumPy) vs Display (Matplotlib)

Hist calculation:
counts, bins = np.histogram(data)
Plot visualization:
plt.bar(bins[:-1], counts)
plt.hist(data) → (Matplotlib does both)

🎯 14. Creating Animations (NumPy generates frames)

Frame generation with NumPy:
y = np.sin(x + phase)
Repeatedly update:
line.set_ydata()
Matplotlib FuncAnimation renders.

🎯 15. Using NumPy for Data Cleaning Before Plotting

Matplotlib cannot handle:
NaN
Inf
NumPy handles them:
np.nan_to_num()
np.isfinite()
np.where()
masking arrays

🎯 16. Machine Learning Visuals Using NumPy → Matplotlib

Decision boundaries:
xx, yy = np.meshgrid()
Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
Plot:
plt.contourf(xx, yy, Z)
Loss curves:
epochs = np.arange()
losses = np.array()
plt.plot(epochs, losses)
Weights histogram:
plt.hist(weights.flatten())

🎯 17. NumPy as a Low-Level Renderer Helper

NumPy computations used for:
• transformations

• scaling
• normalization
• projections
• grid sampling

Matplotlib only draws the result.

Examples:

points @ matrix.T

np.linalg.norm() for distances

np.clip() for color normalization

🎯 18. Custom Colormaps (NumPy arrays)

```
colors = np.linspace(0,1,256)
cmap = ListedColormap(np.c_[colors, colors, colors])
plt.imshow(data, cmap=cmap)
```

🎯 19. Combined Performance Tips

Use NumPy vectorization before plotting

instead of Python loops:

BAD (slow):

for each point: compute…

GOOD (fast):

```
y = np.sin(x)
```