matplotlib Cheat Sheet

This cheat sheet provides a quick reference for essential plotting functions in matplotlib, helping you create and customize various types of visualizations. It covers fundamental plot types—from line and scatter plots to histograms and bar charts—and includes advanced customization options like subplots, color mapping, and annotations.

You'll also find guidance on using pandas Series and DataFrame objects to quickly generate visualizations, perfect for exploring and understanding your data at a glance. These functions are ideal for developing a data cleaning strategy and identifying key trends in your dataset early on.

Designed to help you present data clearly and effectively, this guide ensures you can easily leverage matplotlib's powerful features to gain insights and make data-driven decisions.

Table of Contents

Importing Libraries

Syntax for	How to use	Explained
IMPORT	`import matplotlib.pyplot as plt`	Import the `pyplot` submodule using an alias
	`import seaborn as sns` `sns.set_theme()`	Import `seaborn` and set the default theme
	`import pandas as pd`	Import `pandas` using its common alias
	`import matplotlib.style as style` `style.use('fivethirtyeight')`	Apply the `fivethirtyeight` predefined style

Basic Plotting with matplotlib

Syntax for	How to use	Explained
PLOT	`plt.plot(x_values, y_values)` `plt.show()`	Plot a basic line graph

	`plt.plot(x_values1, y_values1)`
	`plt.plot(x_values2, y_values2)`
	`plt.show()`
		Plot multiple graphs on the same figure

	`plt.plot(x_values1, y_values1)`
	`plt.show()`
	`plt.plot(x_values2, y_values2)`
	`plt.show()`
		Plot graphs in separate figures

Syntax for	How to use	Explained

PLOT	`plt.plot(x_values, y_values,`
	`color='blue', linestyle='--')`
	`plt.show()`
		Customize line color and style

SCATTER	`plt.scatter(x_values, y_values)` `plt.show()`	Create a scatter plot of points

	`plt.scatter(x_values, y_values,`
	`c='red', s=100)`
	`plt.show()`
		Customize point color and size

BAR	`plt.bar(x=x_values, height=heights)` `plt.show()`	Create a vertical bar chart
	`plt.bar(x, height, bottom=previous_heights)` `plt.show()`	Create a stacked bar chart
BARH	`plt.barh(y=y_values, width=widths)` `plt.show()`	Create a horizontal bar chart

	`plt.barh(y, width, color='purple',`
	`edgecolor='black')`
	`plt.show()`
		Customize bar colors and borders

HIST	`plt.hist(data_column)` `plt.show()`	Generate a histogram for a dataset

Basic Plotting with matplotlib

Syntax for	How to use	Explained
HIST	`plt.hist(data_column, bins=30, color='orange')` `plt.show()`	Customize bin count and color
AREA	`plt.fill_between(x, y1=lower, y2=upper)` `plt.show()`	Create an area plot shaded between `y1` and `y2`
PIE	`plt.pie(sizes, labels=labels, autopct='%1.1f%%')` `plt.show()`	Create a pie chart with percentages

Customization

Syntax for	How to use	Explained
TITLE	`plt.title('Title')`	Add a title to the plot
	`plt.title('Custom Title', fontsize=16, color='green')`	Customize title size and color
XLABEL	`plt.xlabel('X-axis Label')`	Add a label to the x-axis
YLABEL	`plt.ylabel('Y-axis Label')`	Add a label to the y-axis

Syntax for	How to use	Explained

LEGEND	`plt.plot(x_values1, y_values1, label='Label 1')`
	`plt.plot(x_values2, y_values2, label='Label 2')`
	`plt.legend()`
	`plt.show()`
		Add a legend to the plot

	`plt.legend(loc='upper right', fontsize=10)` `plt.show()`	Customize legend position and font size
GRID	`plt.grid(True)`	Add gridlines to the plot
SET_XTICKS	`plt.xticks(ticks=x_values, labels=labels)`	Customize the tick labels on the x-axis
	`plt.xticks(rotation=45)`	Rotate the x-axis tick labels
SET_YTICKS	`plt.yticks(ticks=y_values, labels=labels)`	Customize the tick labels on the y-axis
	`plt.yticks(rotation=30)`	Rotate the y-axis tick labels
TICKLABEL_FORMAT	`plt.ticklabel_format(axis='both', style='plain')`	Change scientific notation to plain text

Customization

Syntax for	How to use	Explained
COLORBAR	`plt.scatter(x, y, c=values, cmap='viridis')` `plt.colorbar()`	Use a colormap in the scatter plot
	`plt.scatter(x, y, c=values, cmap='coolwarm')` `plt.colorbar()`	Use a different colormap
ANNOTATE	`plt.annotate('Text', xy=(x, y), xytext=(x_offset, y_offset), arrowprops=dict(facecolor='black', shrink=0.05))` `plt.show()`	Add text and an arrow annotation on the plot

Grid Charts

Syntax for	How to use	Explained

FIGURE	`plt.figure(figsize=(8, 3))`
	`plt.subplot(1, 2, 1)`
	`plt.subplot(1, 2, 2)`
	`plt.show()`
		Create two subplots in a 1-row, 2-column grid

SUBPLOT	`plt.figure(figsize=(10, 12))`
	`for i in range(1, 7):`
	`  plt.subplot(3, 2, i)`
	`  plt.plot(x_values, y_values)`
	`plt.show()`
		Create a 3-row, 2-column grid of subplots

Syntax for	How to use	Explained
SUBPLOTS	`fig, axes = plt.subplots(nrows=4, ncols=1)`	Create a grid of 4 vertically stacked subplots

	`fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10, 8))`
	`axes[0, 0].plot(x_values1, y_values1)`
	`axes[1, 1].plot(x_values2, y_values2)`
	`plt.show()`
		Create a 2x2 grid of subplots and assign plots to specific axes

Advanced Plot Customization

Syntax for	How to use	Explained

SUBPLOTS	`fig, ax = plt.subplots(figsize=(10, 8))`
	`ax.bar(x, height)`
	`plt.show()`
		Create a bar chart using the object-oriented approach

SPINES	`for location in ['left', 'right', 'bottom', 'top']:` `  ax.spines[location].set_visible(False)`	Remove all borders (spines) from the plot
TICK_PARAMS	`ax.tick_params(top=False, left=False)` `ax.tick_params(axis='x', colors='grey')`	Hide specific ticks and change tick colors

Advanced Plot Customization

Syntax for	How to use	Explained
TICK_TOP	`ax.xaxis.tick_top()`	Move x-tick labels to the top of the plot
SET_XTICKS	`ax.set_xticks([0, 150, 300])`	Set custom tick locations on the x-axis
SET_XTICK LABELS	`ax.set_xticklabels(['0', '150', '300'])`	Set custom tick labels on the x-axis
TEXT	`ax.text(x, y, 'Sample Text', fontsize=12, color='blue')`	Add custom text to a specific position on the plot
AXVLINE	`ax.axvline(x=5, color='red', linewidth=2)`	Add a vertical line at a specified x-position with customization
AXHLINE	`ax.axhline(y=3, color='green', linestyle='--')`	Add a horizontal line at a specified y-position with customization

Pandas Visualization

Syntax for	How to use	Explained
LINE (SERIES)	`Series.plot.line()` `plt.show()`	Create a line plot from a Series object
LINE (SERIES)	`Series.plot.line(color='green', linestyle='--')` `plt.show()`	Create a line plot from a Series object
LINE (DATAFRAME)	`DataFrame.plot.line(x='column1', y='column2')` `plt.show()`	Create a line plot from a DataFrame object
SCATTER (DATAFRAME)	`DataFrame.plot.scatter(x='column1', y='column2')` `plt.show()`	Create a scatter plot from a DataFrame object
	`DataFrame.plot.scatter(x='col1', y='col2', color='red', s=50)` `plt.show()`	Customize scatter plot points
HIST (SERIES)	`Series.plot.hist(bins=20)` `plt.show()`	Generate a histogram with custom bin count from a Pandas Series
	`Series.plot.hist(cumulative=True, bins=30)` `plt.show()`	Create a cumulative histogram
BAR (SERIES)	`Series.plot.bar()` `plt.show()`	Create a vertical bar chart from a Pandas Series

Pandas Visualization

Syntax for	How to use	Explained
BARH (SERIES)	`Series.plot.barh()` `plt.show()`	Create a horizontal bar chart from a Series object

	`Series.plot.barh(color='orange',`
	`edgecolor='black')`
	`plt.show()`
		Customize bar colors and borders

BOXPLOT	`DataFrame.plot.box()` `plt.show()`	Create a boxplot to visualize data distributions

Seaborn Visualizations

Syntax for	How to use	Explained

RELPLOT	`sns.relplot(data=data, x='x_var', y='y_var',`
	`hue='hue_var', size='size_var',`
	`style='style_var')`
	`plt.show()`
		Create a relational plot with multiple attributes

	`sns.relplot(data=data, x='x_var', y='y_var',`
	`hue='hue_var', col='col_var')`
	`plt.show()`
		Create subplots for relational plots based on a column

Syntax for	How to use	Explained
HEATMAP	`sns.heatmap(data, annot=True,` `cmap='coolwarm')`	Create a heatmap with annotations
	`sns.heatmap(data, annot=True,` `linewidths=0.5, cmap='Blues')`	Create a heatmap with line spacing and custom colors
PAIRPLOT	`sns.pairplot(data)`	Create pair plots for all combinations of features
VIOLINPLOT	`sns.violinplot(x='x_var', y='y_var',` `data=data)`	Create a violin plot to visualize data distribution
JOINTPLOT	`sns.jointplot(x='x_var', y='y_var',` `data=data, kind='reg')`	Create a joint plot to visualize bivariate data