

All Rights Reserved. Author @ Rajendra Phani		
Basic - Plot - Creation		
Command	What it does	Example
<code>plt.plot(x, y)</code>	2-D line plot (default solid line)	<code>plt.plot(x, y)</code>
<code>plt.scatter(x, y, s=20, c='r')</code>	2-D scatter plot (points)	<code>plt.scatter(x, y, s=50)</code>
<code>plt.bar(x, height)</code>	Vertical bar chart	<code>plt.bar(categories, values)</code>
<code>plt.barh(x, height)</code>	Horizontal bar chart	<code>plt.barh(categories, values)</code>
<code>plt.hist(data, bins=10)</code>	Histogram of 1-D data	<code>plt.hist(values, bins=20)</code>
<code>plt.boxplot(data)</code>	Box-and-whisker plot	<code>plt.boxplot(values)</code>
<code>plt.step(x, y, where='mid')</code>	Step plot (useful for discrete data)	<code>plt.step(x, y)</code>
<code>plt.pie(values, labels=labels)</code>	Pie chart	<code>plt.pie(sizes, labels=labels)</code>
<code>plt.contour(X, Y, Z)</code>	Contour lines (2-D)	<code>plt.contour(X, Y, Z)</code>
<code>plt.contourf(X, Y, Z)</code>	Filled contour plot	<code>plt.contourf(X, Y, Z)</code>
<code>plt.imshow(img, cmap='viridis')</code>	Display 2-D array as image	<code>plt.imshow(img, cmap='gray', origin='lower')</code>
<code>plt.quiver(X, Y, U, V)</code>	Vector field (arrows)	<code>plt.quiver(X, Y, U, V)</code>

Figure - Axes Management		
<code>ax.plot(...)</code>	Plot on a specific Axes object (instead of plt)	<code>ax.plot(x, y)</code>
<code>fig.suptitle('Main Title')</code>	Global title for the whole figure	<code>fig.suptitle('My Figure')</code>
<code>ax.set_title('Axis Title')</code>	Axis-level title	<code>ax.set_title('Time vs. Value')</code>
<code>ax.set_xlabel('X axis'), ax.set_ylabel('Y axis')</code>	Axis labels	<code>ax.set_xlabel('Time (s)')</code>
<code>ax.set_xlim(0, 10), ax.set_ylim(0, 10)</code>	Axis limits	<code>ax.set_xlim([xmin, xmax])</code>
<code>ax.set_aspect('equal')</code>	Equal scaling (1:1)	<code>ax.set_aspect('equal', adjustable='box')</code>
<code>fig.tight_layout()</code>	Auto-adjust subplot params to give sp	<code>fig.tight_layout()</code>
<code>fig.subplots_adjust(hspace=0.3)</code>	Manual spacing	<code>fig.subplots_adjust(hspace=.4)</code>
<code>ax.grid(True, linecolor='gray')</code>	Show gridlines	<code>ax.grid(alpha=0.5)</code>
<code>plt.savefig('plot.png', dpi=300)</code>	Save figure to file	<code>plt.savefig('fig.png', dpi=150)</code>
<code>plt.show()</code>	Display the figure (in scripts)	<code>plt.show()</code>

Styling - Themes		
<code>sns.set_style('whitegrid')</code> (seaborn)	Same as above but via seaborn	<code>sns.set_style('whitegrid')</code>
<code>plt.rcParams.update({'font.size': 12})</code>	Update rcParams globally	<code>plt.rcParams['axes.facecolor'] = '#f0f0f0'</code>
<code>ax.set_prop_cycle(color=['r', 'g', 'b'])</code>	Cycle colors for successive plots on same axes	<code>ax.set_prop_cycle('color', plt.cm.tab10.colors)</code>
<code>plt.colorbar(im, ax=ax)</code>	Add colorbar for image/contour etc.	<code>cbar = plt.colorbar(im, ax=ax)</code>
<code>plt.cm.&lt;colormap&gt;</code>	Access built-in colormaps (e.g., plt.cm.viridis)	<code>cmap = plt.cm.inferno</code>
<code>plt.scatter(..., cmap=cmap)</code>	Colormap for scatter markers (requires mpl)	<code>plt.scatter(x, y, c=z, cmap='plasma')</code>
<code>mpl.rc('axes', labelsiz=14)</code>	Set default font sizes etc.	<code>mpl.rc('font', family='serif')</code>

Annotation - Text		
<code>ax.annotate('Peak', xy=(x0, y0), xytext=(x1, y1), arrowprops=dict(arrowstyle='-&gt;'))</code>	Annotate a point with an arrow	<code>ax.annotate('Max', xy=(xmax, ymax), xytext=(-20, 30), textcoords='offset points')</code>
<code>ax.plot(x0, y0, 'ro')</code>	Marker for annotation point (optional)	<code>ax.plot(xmax, ymax, 'ro')</code>
<code>plt.legend(['Line1', 'Line2'], loc='upper left')</code>	Add legend (auto-generated from labels or explicit)	<code>ax.legend(loc='best')</code>
<code>plt.suptitle('Figure title', fontsize=16)</code>	Big title for whole figure	<code>fig.suptitle('Experiment Results')</code>

Multiple Plots - Subplots		
<code>plt.subplot(2, 3, 4)</code>	Legacy syntax: select subplot #4 in a 2x3 grid	<code>plt.subplot(2, 3, 4); plt.plot(x, y)</code>
<code>ax.set_sharex(other_ax)</code>	Share x-axis with another Axes	<code>ax1.sharex(ax2)</code>
<code>fig.add_axes([left, bottom, width, height])</code>	Add custom Axes in figure coordinates	<code>ax3 = fig.add_axes([0.15, 0.6, 0.25, 0.3])</code>
<code>ax.inset_axes([x0, y0, width, height])</code>	Small inset plot (requires mpl_toolkits)	<code>inset_ax = ax.inset_axes([0.5, 0.5, 0.4, 0.4])</code>

Color - Colormaps		
<code>mpl.colors.Normalize(vmin, vmax)</code>	Normalize data to 0-1 range for colormapping	<code>norm = mpl.colors.Normalize(vmin=data.min(), vmax=data.max())</code>
<code>mpl.cm.ScalarMappable(norm=norm, cmap=cmap)</code>	Create a mappable for colorbars	<code>sm = mpl.cm.ScalarMappable(norm=norm, cmap=cmap)</code>
<code>plt.cm.get_cmap('viridis', N)</code>	Get a colormap with exactly N discrete colors	<code>cmap = plt.cm.get_cmap('tab10', 8)</code>
<code>mpl.colors.ListedColormap(['r', 'g', 'b'])</code>	Custom discrete colormap	<code>cmap = mpl.colors.ListedColormap(['#e41a1c', '#377eb8', '#4daf4a'])</code>

Advanced Plotting (3-D & Project)		
<code>ax.plot_surface(X, Y, Z, cmap='viridis')</code>	3-D surface plot	<code>ax.plot_surface(X, Y, Z)</code>
<code>ax.contour3D(X, Y, Z, 50, cmap='viridis')</code>	3-D contour (wireframe)	<code>ax.contour3D(X, Y, Z, 50)</code>
<code>ax.scatter(x, y, z, c=z, cmap='plasma')</code>	3-D scatter plot	<code>ax.scatter(x, y, z, c=z)</code>
<code>plt.axes(projection='polar')</code>	Polar coordinate plot	<code>ax = plt.subplot(111, projection='polar'); ax.plot(theta, r)</code>