#_ important Matplotlib Operations [+100]

Basic Plotting:

- plt.plot(): Plot y versus x as lines and/or markers.
- plt.scatter(): Make a scatter plot.
- plt.bar(): Make a bar plot.
- plt.barh(): Make a horizontal bar plot.
- plt.hist(): Plot a histogram.
- plt.boxplot(): Make a box and whisker plot.
- plt.pie(): Plot a pie chart.
- plt.fill_between(): Fill area between two horizontal curves.
- plt.errorbar(): Plot error bars.
- plt.stem(): Create a stem plot.

Figure and Axes:

- plt.figure(): Create a new figure.
- plt.subplots(): Create a figure and a set of subplots.
- plt.subplot2grid(): Create an axis at a specific location inside a regular grid.
- plt.axes(): Add axes to the figure.
- fig.add_subplot(): Add a subplot to the current figure.
- ax.plot(): Plot data on a particular axes instance.

Customizing Plots:

- plt.title(): Set the title of the current axes.
- plt.xlabel(): Set the x-axis label.
- plt.ylabel(): Set the y-axis label.
- plt.xlim(): Set the x-axis view limits.
- plt.ylim(): Set the y-axis view limits.
- plt.xticks(): Set the x-axis tick locations and labels.
- plt.yticks(): Set the y-axis tick locations and labels.
- ax.set_xticklabels(): Set the x-axis tick labels on the specific axes.

By: Waleed Mousa

- ax.set_yticklabels(): Set the y-axis tick labels on the specific axes.
- plt.legend(): Place a legend on the axes.
- ax.legend(): Place a legend on the specific axes.
- plt.grid(): Configure the grid lines.
- ax.grid(): Configure the grid lines on a specific axes.
- plt.tight_layout(): Automatically adjust subplot parameters.
- fig.subplots_adjust(): Tune the subplot layout.

Figure Styles and Features:

- plt.style.use(): Use a predefined style.
- mpl.rc(): Set the current rc params.
- plt.rc_context(): Return a context manager for managing rc settings.
- plt.savefig(): Save the current figure.
- plt.show(): Display all open figures.

Colors, Markers, and Line Styles:

- plt.plot(x, y, color='green'): Specify color by name.
- plt.plot(x, y, linestyle='--'): Specify line style.
- plt.plot(x, y, marker='o'): Specify marker style.
- plt.setp(): Set a property on an artist object.

Advanced Plot Types:

- plt.contour(): Contour plot.
- plt.contourf(): Filled contour plot.
- plt.imshow(): Display an image on the axes.
- plt.streamplot(): Draw streamlines of a vector flow.
- plt.quiver(): Plot a 2D field of arrows.
- plt.pcolor(): Create a pseudocolor plot.
- plt.tripcolor(): Create a pseudocolor plot of an unstructured triangular grid.
- plt.tricontour(): Draw contours on an unstructured triangular grid.

By: Waleed Mousa

- plt.hexbin(): Make a hexagonal binning plot.
- plt.stackplot(): Draw a stacked area plot.

Working with Text and Annotations:

- plt.text(): Add text to the axes.
- plt.annotate(): Annotate the point xy with text.
- plt.figtext(): Add text to the figure.

Working with Data:

- plt.fill_betweenx(): Fill area between two vertical curves.
- plt.vlines(): Plot vertical lines.
- plt.hlines(): Plot horizontal lines.
- plt.table(): Add a table to the axes.
- ax.twinx(): Create a second y-axis sharing the same x-axis.
- ax.twiny(): Create a second x-axis sharing the same y-axis.

Customizing Layouts:

- plt.subplot_mosaic(): Create a layout of subplots with a mosaic.
- fig.align_labels(): Align labels for subplots.
- GridSpec(): Specify a geometry for subplots.

Interactive Features:

- plt.connect(): Connect a callback function to an event.
- plt.disconnect(): Disconnect a callback function from an event.
- plt.pause(): Pause the interactive loop.

Customizing Ticks and Spines:

- ax.xaxis.set_major_locator(): Set the locator of the major ticker.
- ax.xaxis.set_minor_locator(): Set the locator of the minor ticker.
- ax.xaxis.set_major_formatter(): Set the formatter of the major ticker.
- ax.xaxis.set_minor_formatter(): Set the formatter of the minor ticker.

By: Waleed Mousa

- `ax.spines['left'].set_position()`: Set the position of the spine.

Legends and Colorbars:

- `ax.legend(loc='upper right')`: Place a legend at a specified location.
- `plt.colorbar()`: Add a colorbar to a plot.
- `ax.legend(handles, labels)`: Create a legend with custom handles and labels.

Animation:

- `animation.FuncAnimation()`: Make an animation by repeatedly calling a function.
- `animation.ArtistAnimation()`: Animation using a fixed set of Artist objects.
- `plt.draw()`: Redraw the current figure.

Working with Paths and Patches:

- `mpl.path.Path()`: Create a new path.
- `mpl.patches.Circle()`: Create a circle patch.
- `mpl.patches.Rectangle()`: Create a rectangle patch.
- `mpl.patches.Polygon()`: Create a polygon patch.

3D Plotting with mplot3d:

- `mpl_toolkits.mplot3d.Axes3D()`: Create a 3D axes.
- `ax.plot_surface()`: Plot a 3D surface.
- `ax.plot_wireframe()`: Plot a 3D wireframe.
- `ax.scatter3D()`: Create a 3D scatter plot.
- `ax.bar3d()`: Create a 3D bar plot.

Working with Images:

- `mpl.image.imread()`: Read an image from a file into an array.
- `mpl.image.imsave()`: Save an array as an image file.

By: Waleed Mousa

Working with Colormaps:

- `mpl.cm.get_cmap()`: Get a colormap instance.
- `mpl.colors.Normalize()`: Normalize a given value to the 0-1 range on a log scale.
- `mpl.colors.LogNorm()`: Normalize a given value to the 0-1 range on a log scale.

Event Handling:

- `mpl.connect('event_name', callback)`: Connect an event with a callback.
- `mpl.disconnect('event_name', callback)`: Disconnect an event from a callback.

Customizing Matplotlib:

- `mpl.use()`: Set the Matplotlib backend.
- `mpl.rcParams.update()`: Update the Matplotlib rcParams.
- `mpl.style.available`: List available styles.

Matplotlib Configuration and Helpers:

- `mpl.get_configdir()`: Get the directory of Matplotlib configuration.
- `mpl.get_cachedir()`: Get the directory of Matplotlib cache.
- `mpl.font_manager.findfont()`: Find a font.

Advanced Features:

- `mpl.colors.Colormap()`: Base class for all scalar to RGBA mappings.
- `mpl.ticker.ScalarFormatter()`: Format tick values as a number.
- `mpl.ticker.FuncFormatter()`: User-defined function for formatting.
- `mpl.dates.DateFormatter()`: Format dates in plots.
- `mpl.dates.AutoDateLocator()`: Auto-select the date ticks.

Matplotlib with Pandas:

- `DataFrame.plot()`: Make plots of Series or DataFrame.

By: Waleed Mousa

- `Series.plot()`: Make plots of Series.

Saving Figures in Different Formats:

- `fig.savefig('filename.png')`: Save the figure as a .png file.
- `fig.savefig('filename.svg')`: Save the figure as a .svg file.
- `fig.savefig('filename.pdf')`: Save the figure as a .pdf file.

Interactive Backends (e.g., with Jupyter Notebook):

- `%matplotlib inline`: Set up Matplotlib for use in Jupyter Notebook.
- `%matplotlib notebook`: Enable interactive figures in a Jupyter notebook.

Customizing Figure Size and DPI:

- `plt.figure(figsize=(8, 6), dpi=100)`: Create a figure with specific size and DPI.

Subplot Spacing and Margins:

- `plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=None)`: Adjust the subplot layout parameters.