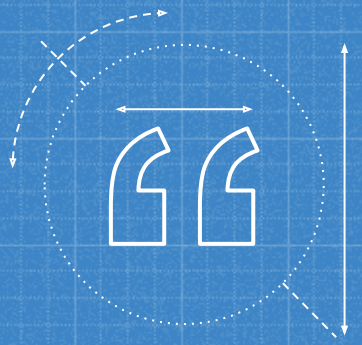
Matplotlib Cheat-Sheets

- Ankita S.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.



Common Methods and Attributes

Labeling functions	
plt.title("title")	Add a title
plt.xlabel("x axis label")	Add a label to the x axis
plt.ylabel("y axis label")	Add a label to the y axis
plt.legend(loc = int)	Add a legend
plt.xticks(range(min, max, interval)	Modify the x axis tick marks

Multiple plots

```
plt.plot(x_data1, y_data1)
plt.plot(x_data2, y_data2)
plt.plot(x_data3, y_data3)
plt.show()
```

You can put multiple plots in one figure by defining each one before plt.show() or plt.savefig()

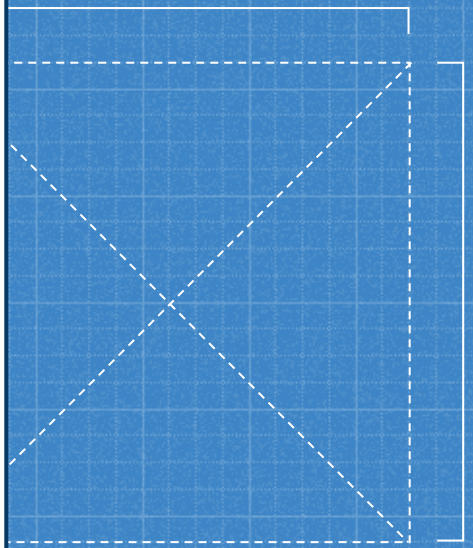
Using colormaps

```
# Choose a colormap and assign to
a variable
cm = plt.cm.get_cmap("RdYlBu")
# Set the color map in a plot
plt.scatter(x_data, y_data,
cmap=cm)
```

Optional arguments	
color ="color"	Change plot color
marker = "symbol"	Change marker for line or scatter plot (".", "x", "|", "o")
markersize = int	Change marker size
linewidth = int	Change line width for line graph
cmap = colormap	Color plot according to a colormap

Key functions	
plt.clf()	Clear figure
plt.savefig("filename")	Save figure (call before plt.show())
plt.show()	Show figure

Axis functions	
plt.xlim(xmin, xmax)	Set the limits for the x axis
plt.ylim(ymin, ymax)	Set the limits for the y axis
plt.xscale("scale type")	Set scale for the x axis (ex. "log")
plt.yscale("scale type")	Set scale for the y axis (ex. "log")
plt.twinx()	Add a second y axis
plt.axis("off")	Do not show the axes
plt.gca().invert_xaxis()	Invert the x axis
plt.gca().invert_yaxis()	Invert the y axis

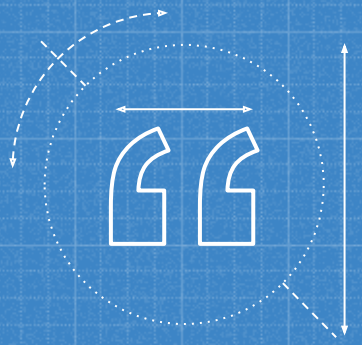


Importing the library

```
import matplotlib.pyplot as plt
```

Plots and key arguments

Line graph	plt.plot()	(x_data, y_data)
Scatter plot	plt.scatter()	(x_data, y_data)
Bar chart	plt.bar()	(x_locs, bar_heights, width = int)
Histogram	plt.hist()	(data, bins = int)
Pie chart	plt.pie()	(data, labels = list)



More Methods and attributes

1 Prepare The Data

Also see Lists & NumPy

1D Data

```
>>> import numpy as np
>>> x = np.linspace(0, 10, 100)
>>> y = np.cos(x)
>>> z = np.sin(x)
```

2D Data or Images

```
>>> data = 2 * np.random.random((10, 10))
>>> data2 = 3 * np.random.random((10, 10))
>>> Y, X = np.mgrid[-3:3:100j, -3:3:100j]
>>> U = -1 - X**2 + Y
>>> V = 1 + X - Y**2
>>> from matplotlib.cbook import get_sample_data
>>> img = np.load(get_sample_data('axes_grid/bivariate_normal.npy'))
```

2 Create Plot

```
>>> import matplotlib.pyplot as plt
```

Figure

```
>>> fig = plt.figure()
>>> fig2 = plt.figure(figsize=plt.figaspect(2.0))
```

Axes

All plotting is done with respect to an `Axes`. In most cases, a subplot will fit your needs. A subplot is an axes on a grid system.

```
>>> fig.add_axes()
>>> ax1 = fig.add_subplot(221) # row-col-num
>>> ax3 = fig.add_subplot(212)
>>> fig3, axes = plt.subplots(nrows=2,ncols=2)
>>> fig4, axes2 = plt.subplots(ncols=3)
```

3 Plotting Routines

1D Data

```
>>> lines = ax.plot(x,y)
>>> ax.scatter(x,y)
>>> axes[0,0].bar([1,2,3],[3,4,5])
>>> axes[1,0].barh([0.5,1,2.5],[0,1,2])
>>> axes[1,1].axhline(0.45)
>>> axes[0,1].axvline(0.65)
>>> ax.fill(x,y,color='blue')
>>> ax.fill_between(x,y,color='yellow')
```

Draw points with lines or markers connecting them
Draw unconnected points, scaled or colored
Plot vertical rectangles (constant width)
Plot horiontal rectangles (constant height)
Draw a horizontal line across axes
Draw a vertical line across axes
Draw filled polygons
Fill between y-values and 0

Vector Fields

```
>>> axes[0,1].arrow(0,0,0.5,0.5)
>>> axes[1,1].quiver(y,z)
>>> axes[0,1].streamplot(X,Y,U,V)
```

Add an arrow to the axes
Plot a 2D field of arrows
Plot 2D vector fields

Data Distributions

```
>>> ax1.hist(y)
>>> ax3.boxplot(y)
>>> ax3.violinplot(z)
```

Plot a histogram
Make a box and whisker plot
Make a violin plot

2D Data or Images

```
>>> fig, ax = plt.subplots()
>>> im = ax.imshow(img,
                   cmap='gist_earth',
                   interpolation='nearest',
                   vmin=-2,
                   vmax=2)
```

Colormapped or RGB arrays

```
>>> axes2[0].pcolor(data2)
>>> axes2[0].pcolormesh(data)
>>> CS = plt.contour(Y,X,U)
>>> axes2[2].contourf(data1)
>>> axes2[2]= ax.clabel(CS)
```

Pseudocolor plot of 2D array
Pseudocolor plot of 2D array
Plot contours
Plot filled contours
Label a contour plot

4 Customize Plot

Colors, Color Bars & Color Maps

```
>>> plt.plot(x, x, x, x**2, x, x**3)
>>> ax.plot(x, y, alpha = 0.4)
>>> ax.plot(x, y, c='k')
>>> fig.colorbar(im, orientation='horizontal')
>>> im = ax.imshow(img,
                   cmap='seismic')
```

Markers

```
>>> fig, ax = plt.subplots()
>>> ax.scatter(x,y,marker=".")
>>> ax.plot(x,y,marker="o")
```

Linestyles

```
>>> plt.plot(x,y,linewidth=4.0)
>>> plt.plot(x,y,ls='solid')
>>> plt.plot(x,y,ls='--')
>>> plt.plot(x,y,'--',x**2,y**2,'-.')
>>> plt.setp(lines,color='r',linewidth=4.0)
```

Text & Annotations

```
>>> ax.text(1,
            -2.1,
            'Example Graph',
            style='italic')
>>> ax.annotate("Sine",
                xy=(8, 0),
                xycoords='data',
                xytext=(10.5, 0),
                textcoords='data',
                arrowprops=dict(arrowstyle="->",
                                connectionstyle="arc3"),)
```

Mathtext

```
>>> plt.title(r'$sigma_i=15$', fontsize=20)
```

Limits, Legends & Layouts

Limits & Autoscaling

```
>>> ax.margins(x=0.0,y=0.1)
```
Add padding to a plot

```
>>> ax.axis('equal')
```
Set the aspect ratio of the plot to 1

```
>>> ax.set(xlim=[0,10.5],ylim=[-1.5,1.5])
```
Set limits for x-and y-axis

```
>>> ax.set_xlim(0,10.5)
```
Set limits for x-axis

Legends

```
>>> ax.set(title='An Example Axes',
           ylabel='Y-Axis',
           xlabel='X-Axis')
```
Set a title and x-and y-axis labels

```
>>> ax.legend(loc='best')
```
No overlapping plot elements

Ticks

```
>>> ax.xaxis.set(ticks=range(1,5),
                 ticklabels=[3,100,-12,"foo"])
```
Manually set x-ticks

```
>>> ax.tick_params(axis='y',
                   direction='inout',
                   length=10)
```
Make y-ticks longer and go in and out

Subplot Spacing

```
>>> fig3.subplots_adjust(wspace=0.5,
                         hspace=0.3,
                         left=0.125,
                         right=0.9,
                         top=0.9,
                         bottom=0.1)
```
Adjust the spacing between subplots

```
>>> fig.tight_layout()
```
Fit subplot(s) in to the figure area

Axis Spines

```
>>> ax1.spines['top'].set_visible(False)
```
Make the top axis line for a plot invisible

```
>>> ax1.spines['bottom'].set_position(('outward',10))
```
Move the bottom axis line outward

5 Save Plot

Save figures

```
>>> plt.savefig('foo.png')
```

Save transparent figures

```
>>> plt.savefig('foo.png', transparent=True)
```

6 Show Plot

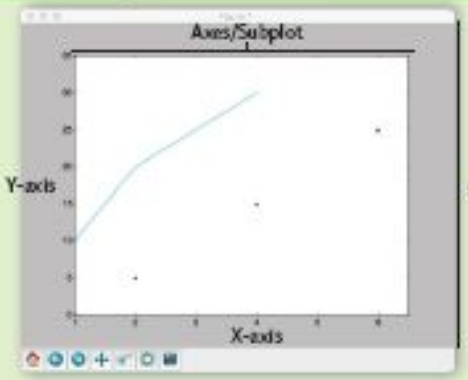
```
>>> plt.show()
```

Close & Clear

`>>> plt.cla()`	Clear an axis
`>>> plt.clf()`	Clear the entire figure
`>>> plt.close()`	Close a window

Plot Anatomy & Workflow

Plot Anatomy



Workflow

The basic steps to creating plots with matplotlib are:

1 Prepare data 2 Create plot 3 Plot 4 Customize plot 5 Save plot 6 Show plot

```
>>> import matplotlib.pyplot as plt
>>> x = [1,2,3,4]                    Step 1
>>> y = [10,20,25,30]
>>> fig = plt.figure()               Step 2
>>> ax = fig.add_subplot(111)        Step 3
>>> ax.plot(x, y, color='lightblue', linewidth=3)   Step 3, 4
>>> ax.scatter([2,4,6],
               [5,15,25],
               color='darkgreen',
               marker='^')
>>> ax.set_xlim(1, 6.5)
>>> plt.savefig('foo.png')
>>> plt.show()                       Step 6
```