

Matplotlib Cheat Sheet

f616 adapted from datacamp.com

Introductory Note

This document is an adaption of the original datacamp.org cheat sheet.

- <https://www.datacamp.com/resources/cheat-sheets/matplotlib-cheat-sheet-plotting-in-python>
- <https://github.com/f616/Python-Matplotlib-Cheat-Sheet>

Matplotlib

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

1 Prepare The Data

1D Data

```
1 import numpy as np
2 x = np.linspace(0, 10, 100)
3 y = np.cos(x)
4 z = np.sin(x)
```

2D Data or Images

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

2 Create Plot

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

Figure

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

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.

Axes

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

3 Save Plot

```
1 plt.savefig('foo.png') #Save figures
2 plt.savefig('foo.png', transparent=True) #Save transparent figures
```

4 Show Plot

```
1 plt.show()
```

5 Plotting Routines

1D Data

```
1 fig, ax = plt.subplots()
2 lines = ax.plot(x,y) #Draw points with lines or markers
  connecting them
3 ax.scatter(x,y) #Draw unconnected points, scaled or colored
4 axes[0,0].bar([1,2,3],[3,4,5]) #Plot vertical rectangles
  (constant width)
5 axes[1,0].barh([0.5,1,2.5],[0,1,2]) #Plot horizontal
  rectangles (constant height)
6 axes[1,1].axhline(0.45) #Draw a horizontal line across axes
7 axes[0,1].axvline(0.65) #Draw a vertical line across axes
8 ax.fill(x,y,color='blue') #Draw filled polygons
9 ax.fill_between(x,y,color='yellow') #Fill between y-values
  and 0
```

2D Data

```
1 fig, ax = plt.subplots()
2 im = ax.imshow(img, cmap='gist_earth',
  interpolation='nearest', vmin=-2, vmax=2) #Colormapped or
  RGB arrays
3 axes2[0].pcolor(data2) #Pseudocolor plot of 2D array
4 axes2[0].pcolormesh(data) #Pseudocolor plot of 2D array
5 CS = plt.contour(Y,X,U) #Plot contours
6 axes2[2].contourf(data1) #Plot filled contours
7 axes2[2] = ax.clabel(CS) #Label a contour plot
```

Vector Fields

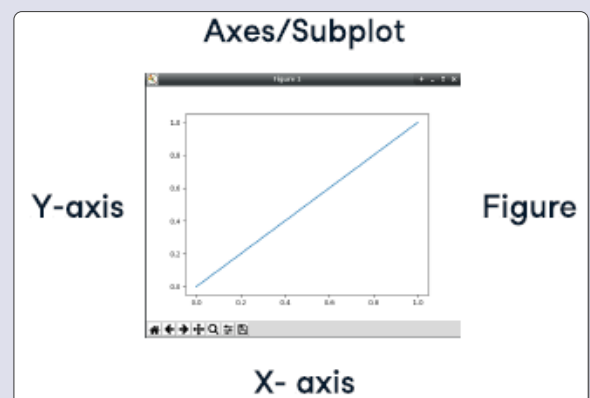
```
1 axes[0,1].arrow(0,0,0.5,0.5) #Add an arrow to the axes
2 axes[1,1].quiver(y,z) #Plot a 2D field of arrows
3 axes[0,1].streamplot(X,Y,U,V) #Plot a 2D field of arrows
```

Data Distributions

```
1 ax1.hist(y) #Plot a histogram
2 ax3.boxplot(y) #Make a box and whisker plot
3 ax3.violinplot(z) #Make a violin plot
```

6 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

```

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

```

7 Close and Clear

```

1 plt.cla() #Clear an axis
2 plt.clf() #Clear the entire figure
3 plt.close() #Close a window

```

8 Plotting Customize Plot

Colors, Color Bars & Color Maps

```

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

```

Makers

```

1 fig, ax = plt.subplots()
2 ax.scatter(x, y, marker='.')
3 ax.plot(x, y, marker='o')

```

Linestyles

```

1 plt.plot(x, y, linewidth=4.0)
2 plt.plot(x, y, ls='solid')
3 plt.plot(x, y, ls='--')
4 plt.plot(x, y, '--', x**2, y**2, '-.')
5 plt.setp(lines, color='r', linewidth=4.0)

```

Text & Annotations

```

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

```

Mathtext

```

1 plt.title(r'$\sigma_i=15$', fontsize=20)

```

Limits & Autoscaling

```

1 ax.margins(x=0.0,y=0.1) #Add padding to a plot
2 ax.axis('equal') #Set the aspect ratio of the plot to 1
3 ax.set(xlim=[0,10.5],ylim=[-1.5,1.5]) #Set limits for x-and y-axis
4 ax.set_xlim(0,10.5) #Set limits for x-axis

```

Legends

```

1 ax.set(title='An Example Axes', ylabel='Y-Axis', xlabel='X-Axis') #Set a title and x-and y-axis labels
2 ax.legend(loc='best') #No overlapping plot elements

```

Ticks

```

1 ax.xaxis.set(ticks=range(1,5), ticklabels=[3,100,-12,'foo']) #Manually set x-ticks
2 ax.tick_params(axis='y', direction='inout', length=10) #Make y-ticks longer and go in and out

```

Subplot Spacing

```

1 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
3 fig.tight_layout() #Fit subplot(s) in to the figure area

```

Axis Spines

```

1 ax1.spines['top'].set_visible(False) #Make the top axis line for a plot invisible
2 ax1.spines['bottom'].set_position(('outward',10)) #Move the bottom axis line outward

```