matplotlib

Cheat sheet Version 3.7.4

Quick start API

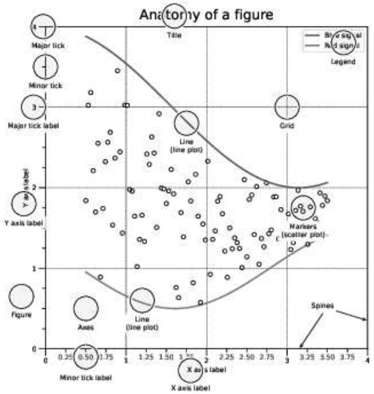
```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt


X = np.linspace(0, 2*np.pi, 100)
Y = np.cos(X)

fig, ax = plt.subplots()
ax.plot(X, Y, color='green')

fig.savefig("figure.pdf")
plt.show()
```

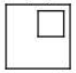
Anatomy of a figure



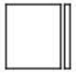
Subplots layout API

`subplot[s](rows, cols, …)` API
`fig, axs = plt.subplots(3, 3)`

`G = gridspec(rows,cols, …)` API
`ax = G[0, :]`



`ax.inset_axes(extent)` API

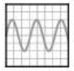


`d=make_axes_locatable(ax)` API
`ax = d.new_horizontal('10%')`

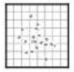
Getting help

- matplotlib.org
- github.com/matplotlib/matplotlib/issues
- discourse.matplotlib.org
- stackoverflow.com/questions/tagged/matplotlib
- https://gitter.im/matplotlib/matplotlib
- twitter.com/matplotlib
- ✉ Matplotlib users mailing list

Basic plots

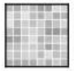


`plot([X], Y, [fmt], …)` API
X, Y, fmt, color, marker, linestyle



`scatter(X, Y, …)` API
X, Y, [s]izes, [c]olors, marker, cmap

`bar[h](x, height, …)` API
x, height, width, bottom, align, color



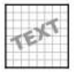
`imshow(Z, …)` API
Z, cmap, interpolation, extent, origin

`contour[f]([X], [Y], Z, …)` API
X, Y, Z, levels, colors, extent, origin

`pcolormesh([X], [Y], Z, …)` API
X, Y, Z, vmin, vmax, cmap

`quiver([X], [Y], U, V, …)` API
X, Y, U, V, C, units, angles

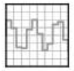
`pie(X, …)` API
Z, explode, labels, colors, radius



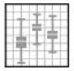
`text(x, y, text, …)` API
x, y, text, va, ha, size, weight, transform

`fill[_between][x](…)` API
X, Y1, Y2, color, where

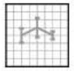
Advanced plots



`step(X, Y, [fmt], …)` API
X, Y, fmt, color, marker, where

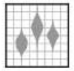


`boxplot(X, …)` API
X, notch, sym, bootstrap, widths

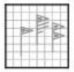


`errorbar(X,Y,xerr,yerr, …)` API
X, Y, xerr, yerr, fmt

`hist(X, bins, …)` API
X, bins, range, density, weights



`violinplot(D, …)` API
D, positions, widths, vert



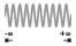
`barbs([X], [Y], U, V, …)` API
X, Y, U, V, C, length, pivot, sizes

`eventplot(positions, …)` API
positions, orientation, lineoffsets

`hexbin(X, Y, C, …)` API
X, Y, C, gridsize, bins

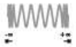
Scales API

`ax.set_[xy]scale(scale, …)`



`linear`
any values

`log`
values > 0



`symlog`
any values

`logit`
0 < values < 1

Projections API

`subplot(…, projection=p)`

p='polar'

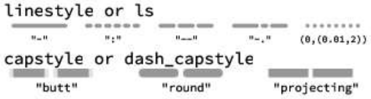
p='3d' API

`p=ccrs.Orthographic()` API
import cartopy.crs as ccrs

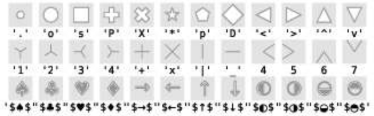
Lines API

`linestyle or ls`

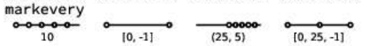
`capstyle or dash_capstyle`



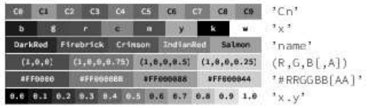
Markers API



`markevery`

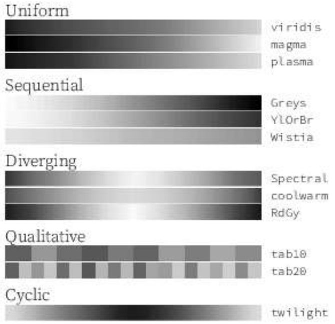


Colors API



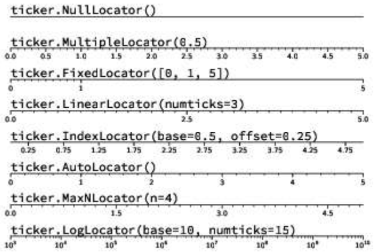
Colormaps API

`plt.get_cmap(name)`



Tick locators API

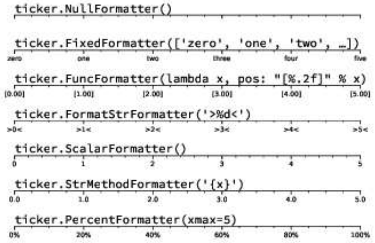
```
from matplotlib import ticker
ax.[xy]axis.set_[minor|major]_locator(locator)
```



```
ticker.NullLocator()
ticker.MultipleLocator(0.5)
ticker.FixedLocator([0, 1, 5])
ticker.LinearLocator(numticks=3)
ticker.IndexLocator(base=0.5, offset=0.25)
ticker.AutoLocator()
ticker.MaxNLocator(n=4)
ticker.LogLocator(base=10, numticks=15)
```

Tick formatters API

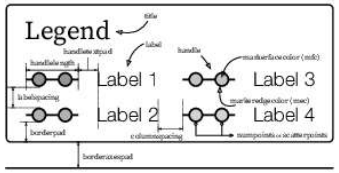
```
from matplotlib import ticker
ax.[xy]axis.set_[minor|major]_formatter(formatter)
```



```
ticker.NullFormatter()
ticker.FixedFormatter(['zero', 'one', 'two', …])
ticker.FuncFormatter(lambda x, pos: "[%.2f]" % x)
ticker.FormatStrFormatter('>%d<')
ticker.ScalarFormatter()
ticker.StrMethodFormatter('{x}')
ticker.PercentFormatter(xmax=5)
```

Ornaments

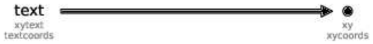
`ax.legend(…)` API
handles, labels, loc, title, frameon



`ax.colorbar(…)` API
mappable, ax, cax, orientation

0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

`ax.annotate(…)` API
text, xy, xytext, xycoords, textcoords, arrowprops



Event handling API

```
fig, ax = plt.subplots()
def on_click(event):
  print(event)
fig.canvas.mpl_connect(
  'button_press_event', on_click)
```

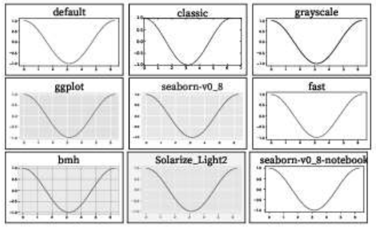
Animation API

```
import matplotlib.animation as mpla

T = np.linspace(0, 2*np.pi, 100)
S = np.sin(T)
line, = plt.plot(T, S)
def animate(i):
    line.set_ydata(np.sin(T+i/50))
anim = mpla.FuncAnimation(
    plt.gcf(), animate, interval=5)
plt.show()
```

Styles API

`plt.style.use(style)`



Quick reminder

```
ax.grid()
ax.set_[xy]lim(vmin, vmax)
ax.set_[xy]label(label)
ax.set_[xy]ticks(ticks, [labels])
ax.set_[xy]ticklabels(labels)
ax.set_title(title)
ax.tick_params(width=10, …)
ax.set_axis_[on|off]()

fig.suptitle(title)
fig.tight_layout()
plt.gcf(), plt.gca()
mpl.rc('axes', linewidth=1, …)
[fig|ax].patch.set_alpha(0)
text=r'$\frac{-e^{i\pi}}{2^n}$'
```

Keyboard shortcuts API

ctrl + s	Save	ctrl + w	Close plot
r	Reset view	f	Fullscreen 0/1
f	View forward	b	View back
p	Pan view	o	Zoom to rect
x	X pan/zoom	y	Y pan/zoom
g	Minor grid 0/1	G	Major grid 0/1
l	X axis log/linear	L	Y axis log/linear

Ten simple rules READ

1. Know your audience
2. Identify your message
3. Adapt the figure
4. Captions are not optional
5. Do not trust the defaults
6. Use color effectively
7. Do not mislead the reader
8. Avoid "chartjunk"
9. Message trumps beauty
10. Get the right tool