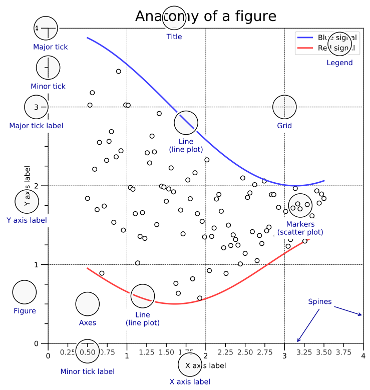
matplotlib

Cheat sheet Version 3.2

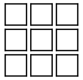
Quick start

```
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='C1')

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

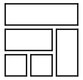
Anatomy of a figure



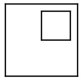
Subplots layout



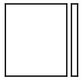
subplot[s](cols,rows,…)
fig, axs = plt.subplots(3,3)



G = gridspec(cols,rows,…)
ax = G[0,:]



ax.inset_axes(extent)

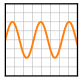


d=make_axes_locatable(ax)
ax=d.new_horizontal('10%')

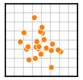
Getting help

matplotlib.org
github.com/matplotlib/matplotlib/issues
discourse.matplotlib.org
stackoverflow.com/matplotlib
gitter.im/matplotlib
twitter.com/matplotlib
Matplotlib users mailing list

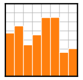
Basic plots



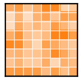
plot([X],Y,[fmt],…)
X, Y, fmt, color, marker, linestyle



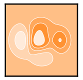
scatter(X,Y,…)
X, Y, [s]izes, [c]olors, markers, cmap



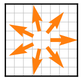
bar[h](x,height,…)
x, height, width, bottom, align, color



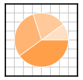
imshow(Z,[cmap],…)
Z, cmap, interpolation, extent, origin



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

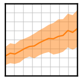


quiver([X],[Y],U,V,…)
X, Y, U, V, C, units, angles



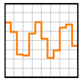
pie(X,[explode],…)
Z, explode, labels, colors, radius

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

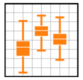


fill[_between][x](…)
X, Y1, Y2, color, where

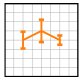
Advanced plots



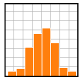
step(X,Y,[fmt],…)
X, Y, fmt, color, marker, where



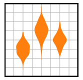
boxplot(X,…)
X, notch, sym, bootstrap, widths



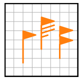
errorbar(X,Y,xerr,yerr,…)
X, Y, xerr, yerr, fmt



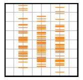
hist(X, bins, …)
X, bins, range, density, weights



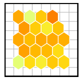
violinplot(D,…)
D, positions, widths, vert



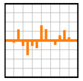
barbs([X],[Y], U, V, …)
X, Y, U, V, C, length, pivot, sizes



eventplot(positions,…)
positions, orientation, lineoffsets



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



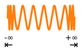
xcorr(X,Y,…)
X, Y, normed, detrend

Scales

ax.set_[xy]scale(scale,…)

linear
any values

log
values > 0

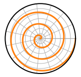


symlog
any values

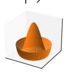
logit
0 < values < 1

Projections

subplot(…,projection=p)



p='polar'

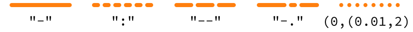


p='3d'

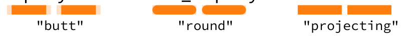
p=Orthographic()
from cartopy.crs import Cartographic

Lines

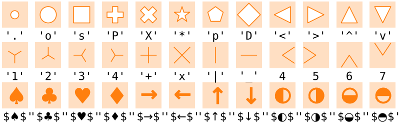
linestyle or ls



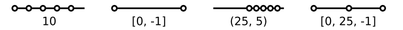
capstyle or dash_capstyle



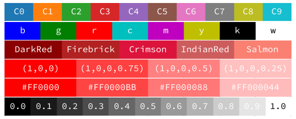
Markers



markevery



Colors

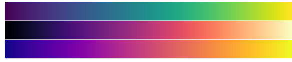


'Cn'
'x'
'name'
(R,G,B[,A])
'#RRGGBB[AA]'
'x.y'

Colormaps

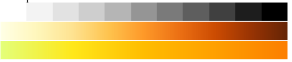
plt.get_cmap(name)

Uniform



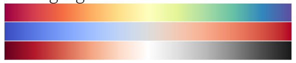
viridis
magma
plasma

Sequential



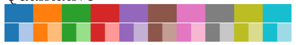
Greys
YlOrBr
Wistia

Diverging



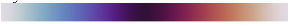
Spectral
coolwarm
RdGy

Qualitative



tab10
tab20

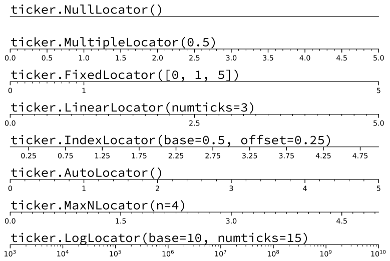
Cyclic



twilight

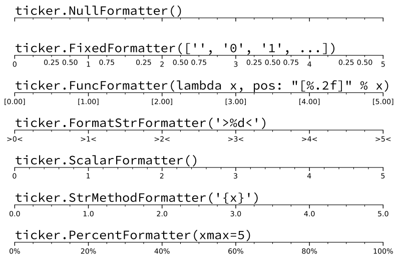
Tick locators

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



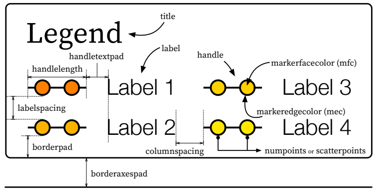
Tick formatters

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

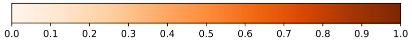


Ornaments

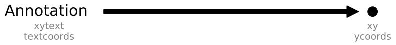
ax.legend(…)
handles, labels, loc, title, frameon



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



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



Event handling

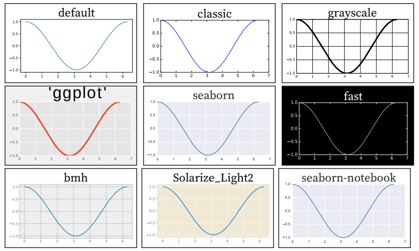
```
fig, ax = plt.subplots()
def on_click(event):
    print(event)
fig.canvas.mpl_connect(
    'button_press_event', on_click)
```

Animation

```
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

plt.style.use(style)



Quick reminder

```
ax.grid()
ax.patch.set_alpha(0)
ax.set_[xy]lim(vmin, vmax)
ax.set_[xy]label(label)
ax.set_[xy]ticks(list)
ax.set_[xy]ticklabels(list)
ax.set_[sup]title(title)
ax.tick_params(width=10, …)
ax.set_axis_[on|off]()

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

Keyboard shortcuts

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

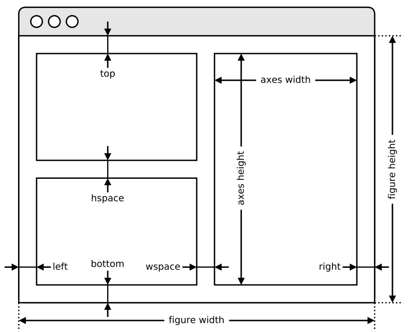
Ten Simple Rules

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

Axes adjustements

API

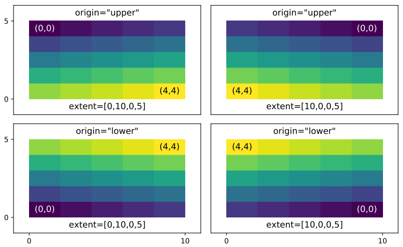
plt.subplot_adjust(…)



Extent & origin

API

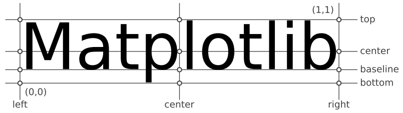
ax.imshow(extent=…, origin=…)



Text alignments

API

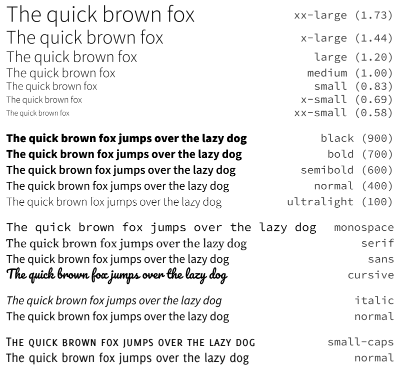
ax.text(…, ha=… , va=…, …)



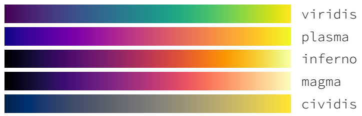
Text parameters

API

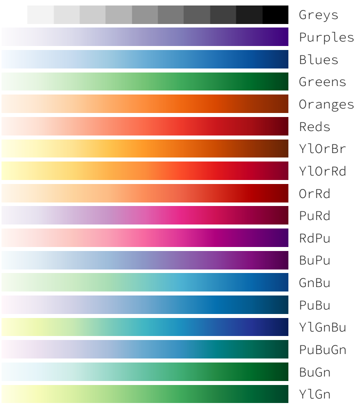
ax.text(…, family=… , size=…, weight = …)
ax.text(…, fontproperties = …)


The quick brown fox xx-large (1.73)
The quick brown fox x-large (1.44)
The quick brown fox large (1.20)
The quick brown fox medium (1.00)
The quick brown fox small (0.83)
The quick brown fox x-small (0.69)
The quick brown fox xx-small (0.58)
The quick brown fox jumps over the lazy dog black (900)
The quick brown fox jumps over the lazy dog bold (700)
The quick brown fox jumps over the lazy dog semibold (600)
The quick brown fox jumps over the lazy dog normal (400)
The quick brown fox jumps over the lazy dog ultralight (100)
The quick brown fox jumps over the lazy dog monospace
The quick brown fox jumps over the lazy dog serif
The quick brown fox jumps over the lazy dog sans
The quick brown fox jumps over the lazy dog cursive
The quick brown fox jumps over the lazy dog italic
The quick brown fox jumps over the lazy dog normal
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG small-caps
The quick brown fox jumps over the lazy dog normal

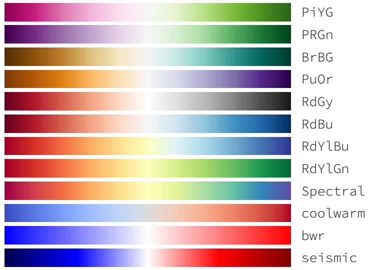
Uniform colormaps



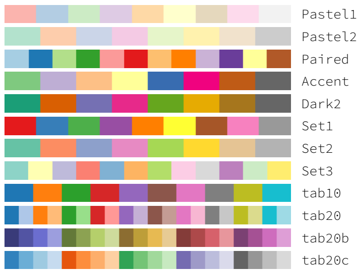
Sequential colormaps



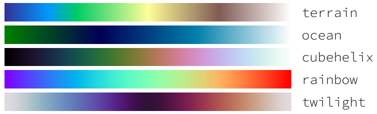
Diverging colormaps



Qualitative colormaps



Miscellaneous colormaps



Color names

API

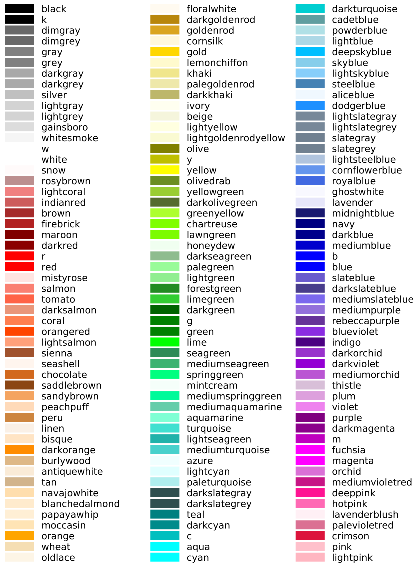
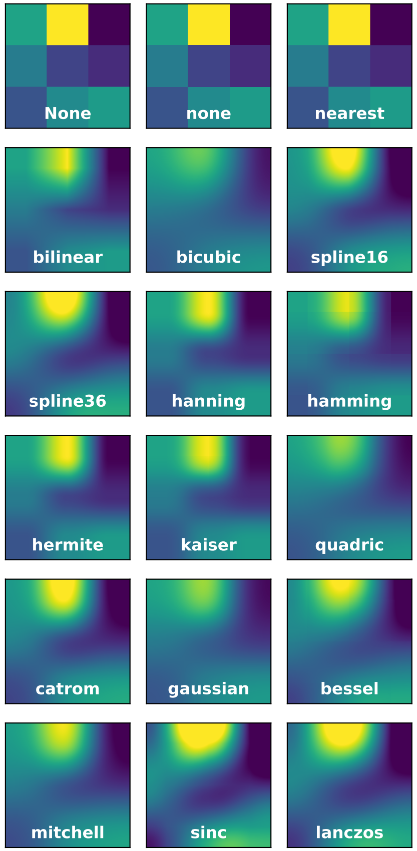
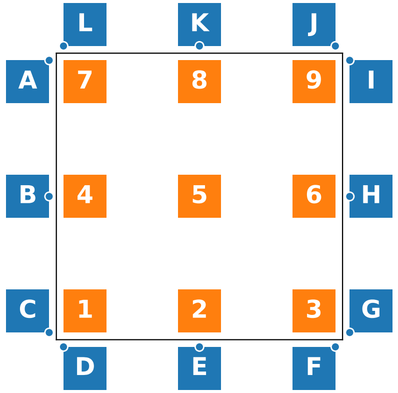


Image interpolation

API



Legend placement



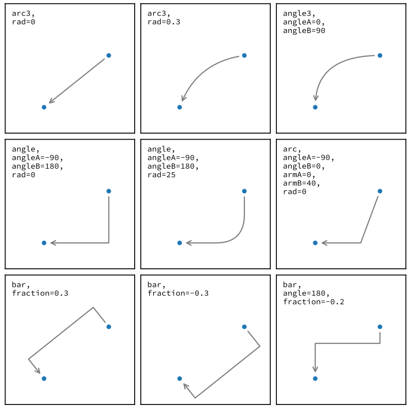
ax.legend(loc="string", bbox_to_anchor=(x,y))

1: lower left	2: lower center	3: lower right
4: left	5: center	6: right
7: upper left	8: upper center	9: upper right

A: upper right / (-.1,.9)	B: right / (-.1,.5)
C: lower right / (-.1,.1)	D: upper left / (-.1,-.1)
E: upper center / (.5,-.1)	F: upper right / (.9,-.1)
G: lower left / (1.1,.1)	H: left / (1.1,.5)
I: upper left / (1.1,.9)	J: lower right / (.9,1.1)
K: lower center / (.5,1.1)	L: lower left / (.1,1.1)

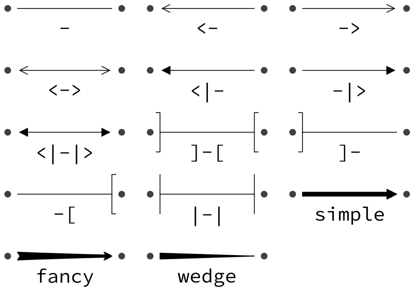
Annotation connection styles

API



Annotation arrow styles

API



How do I …

… resize a figure?
→ fig.set_size_inches(w,h)

… save a figure?
→ fig.savefig("figure.pdf")

… save a transparent figure?
→ fig.savefig("figure.pdf", transparent=True)

… clear a figure?
→ ax.clear()

… close all figures?
→ plt.close("all")

… remove ticks?
→ ax.set_xticks([])

… remove tick labels ?
→ ax.set_[xy]ticklabels([])

… rotate tick labels ?
→ ax.set_[xy]ticks(rotation=90)

… hide top spine?
→ ax.spines['top'].set_visible(False)

… hide legend border?
→ ax.legend(frameon=False)

… show error as shaded region?
→ ax.fill_between(X, Y+error, Y-error)

… draw a rectangle?
→ ax.add_patch(plt.Rectangle((0, 0),1,1)

… draw a vertical line?
→ ax.axvline(x=0.5)

… draw outside frame?
→ ax.plot(…, clip_on=False)

… use transparency?
→ ax.plot(…, alpha=0.25)

… convert an RGB image into a gray image?
→ gray = 0.2989*R+0.5870*G+0.1140*B

… set figure background color?
→ fig.patch.set_facecolor("grey")

… get a reversed colormap?
→ plt.get_cmap("viridis_r")

… get a discrete colormap?
→ plt.get_cmap("viridis", 10)

… show a figure for one second?
→ fig.show(block=False), time.sleep(1)

Performance tips

```
scatter(X, Y)                                slow
plot(X, Y, marker="o", ls="")                fast

for i in range(n): plot(X[i])                slow
plot(sum([x+[None] for x in X],[]))          fast

cla(), imshow(…), canvas.draw()              slow
im.set_data(…), canvas.draw()                fast
```

Beyond Matplotlib

Seaborn: Statistical Data Visualization
Cartopy: Geospatial Data Processing
yt: Volumetric data Visualization
mpld3: Bringing Matplotlib to the browser
Datashader: Large data processing pipeline
plotnine: A Grammar of Graphics for Python

NUMFOCUS
OPEN CODE = BETTER SCIENCE