matplotlib

Cheat sheet

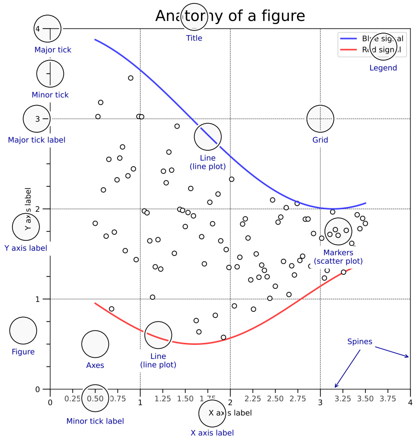
Version 3.5.0

Quick start

API

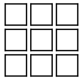
```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt


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



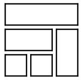
Subplots layout

API



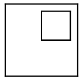
`subplot[s](rows,cols,…)` API

`fig, axs = plt.subplots(3, 3)`

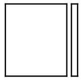


`G = gridspec(rows,cols,…)` API

`ax = G[0,:]`



`ax.inset_axes(extent)` API



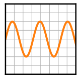
`d=make_axes_locatable(ax)` API

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

Getting help

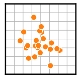
- matplotlib.org
- github.com/matplotlib/matplotlib/issues
- discourse.matplotlib.org
- stackoverflow.com/questions/tagged/matplotlib
- https://gitter.im/matplotlib/matplotlib
- twitter.com/matplotlib
- Matplotlib users mailing list

Basic plots



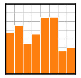
`plot([X],Y,[fmt],…)` API

X, Y, fmt, color, marker, linestyle



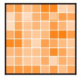
`scatter(X,Y,…)` API

X, Y, [s]izes, [c]olors, marker, cmap



`bar[h](x,height,…)` API

x, height, width, bottom, align, color



`imshow(Z,…)` API

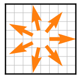
Z, cmap, interpolation, extent, origin

`contour[f]([X],[Y],Z,…)` API

X, Y, Z, levels, colors, extent, origin

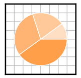
`pcolormesh([X],[Y],Z,…)` API

X, Y, Z, vmin, vmax, cmap



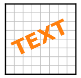
`quiver([X],[Y],U,V,…)` API

X, Y, U, V, C, units, angles



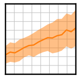
`pie(X,…)` API

Z, explode, labels, colors, radius



`text(x,y,text,…)` API

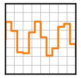
x, y, text, va, ha, size, weight, transform



`fill[_between][x](…)` API

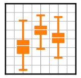
X, Y1, Y2, color, where

Advanced plots



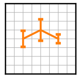
`step(X,Y,[fmt],…)` API

X, Y, fmt, color, marker, where



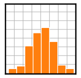
`boxplot(X,…)` API

X, notch, sym, bootstrap, widths



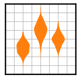
`errorbar(X,Y,xerr,yerr,…)` API

X, Y, xerr, yerr, fmt



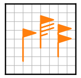
`hist(X, bins, …)` API

X, bins, range, density, weights



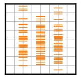
`violinplot(D,…)` API

D, positions, widths, vert



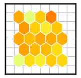
`barbs([X],[Y], U, V, …)` API

X, Y, U, V, C, length, pivot, sizes



`eventplot(positions,…)` API

positions, orientation, lineoffsets



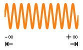
`hexbin(X,Y,C,…)` API

X, Y, C, gridsize, bins

Scales

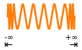
API

`ax.set_[xy]scale(scale,…)`



`linear`
any values

`log`
values > 0



`symlog`
any values

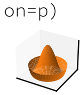
`logit`
0 < values < 1

Projections

API

`subplot(…,projection=p)`

`p='polar'`



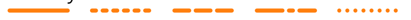
`p='3d'` API

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

Lines

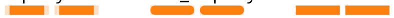
API

`linestyle or ls`



"-" ":" "--" "-." (0,(0.01,2))

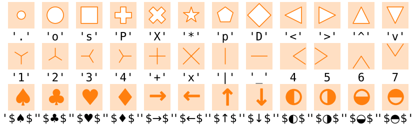
`capstyle or dash_capstyle`



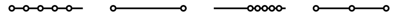
"butt" "round" "projecting"

Markers

API



`markevery`



10 [0, -1] (25, 5) [0, 25, -1]

Colors

API

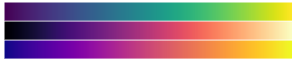
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	'Cn'
b	g	r	c	m	y	k	w			'x'
DarkRed	Firebrick	Crimson	IndianRed	Salmon						'name'
(1,0,0)	(1,0,0,0.75)	(1,0,0,0.5)	(1,0,0,0.25)							(R,G,B[,A])
#FF0000	#FF0000BB	#FF000088	#FF000044							'#RRGGBB[AA]'
0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0 'x.y'

Colormaps

API

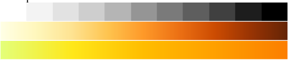
`plt.get_cmap(name)`

Uniform



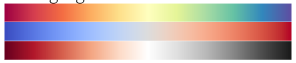
viridis
magma
plasma

Sequential



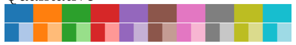
Greys
YlOrBr
Wistia

Diverging



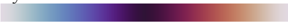
Spectral
coolwarm
RdGy

Qualitative



tab10
tab20

Cyclic

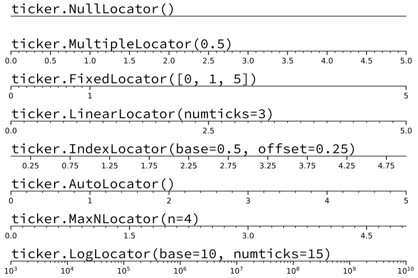


twilight

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()`

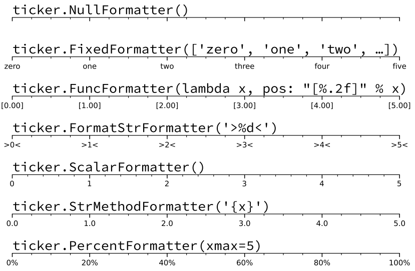
`ticker.MaxNLocator(n=4)`

`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()`

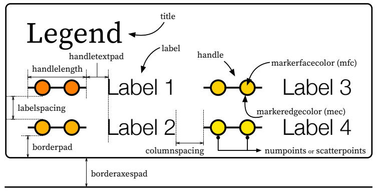
`ticker.StrMethodFormatter('{x}')`

`ticker.PercentFormatter(xmax=5)`

Ornaments

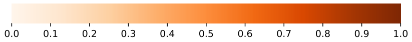
`ax.legend(…)` API

handles, labels, loc, title, frameon



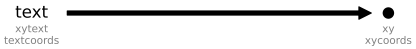
`ax.colorbar(…)` API

mappable, ax, cax, orientation



`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

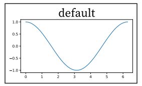
API

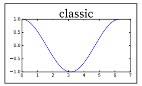
```
import matplotlib.animation as mpla

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()
```

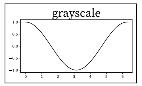
Styles

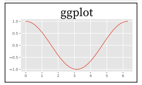
API

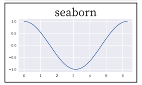
`plt.style.use(style)`

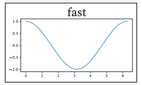


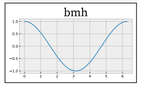


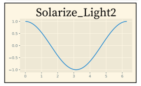


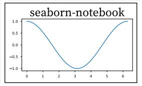












Quick reminder

```
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

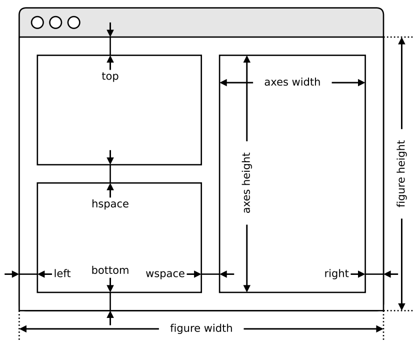
READ

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

Axes adjustments

API

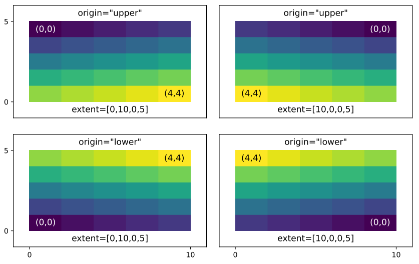
plt.**subplots_adjust**(...)



Extent & origin

API

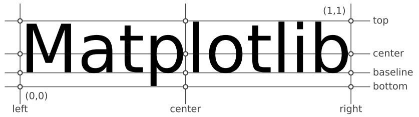
ax.**imshow**(extent=..., origin=...)



Text alignments

API

ax.**text**(..., ha=... , va=..., ...)



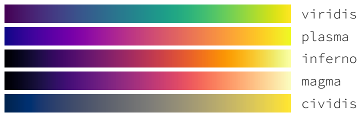
Text parameters

API

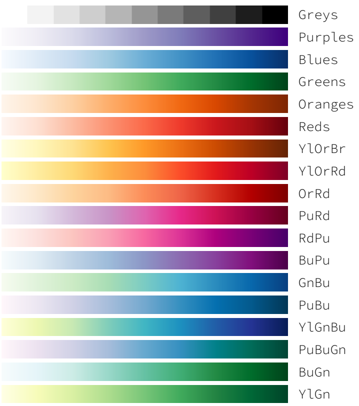
ax.**text**(..., family=..., size=..., weight=...)
ax.**text**(..., fontproperties=...)

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

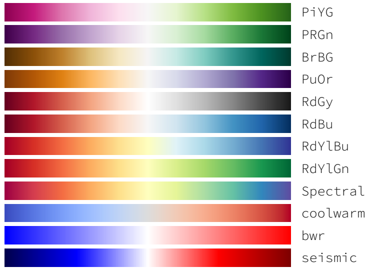
Uniform colormaps



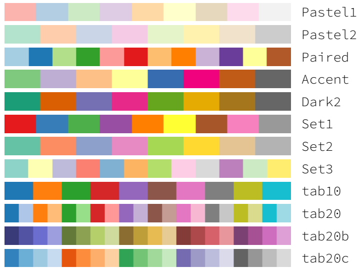
Sequential colormaps



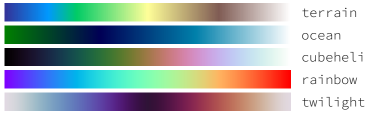
Diverging colormaps



Qualitative colormaps



Miscellaneous colormaps



Color names

API

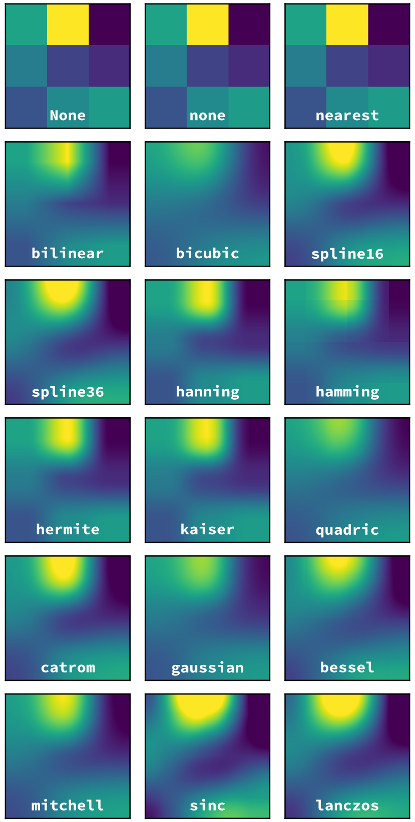
black
k
dimgray
dimgrey
gray
grey
darkgray
darkgrey
silver
lightgray
lightgrey
gainsboro
whitesmoke
w
white
snow
rosybrown
lightcoral
indianred
brown
firebrick
maroon
darkred
r
red
mistyrose
salmon
tomato
darksalmon
coral
orangered
lightsalmon
sienna
seashell
chocolate
saddlebrown
sandybrown
peachpuff
peru
linen
bisque
darkorange
burlywood
antiquewhite
tan
navajowhite
blanchedalmond
papayawhip
moccasin
orange
wheat
oldlace

floralwhite
darkgoldenrod
goldenrod
cornsilk
gold
lemonchiffon
khaki
palegoldenrod
darkkhaki
ivory
beige
lightyellow
lightgoldenrodyellow
olive
y
yellow
olivedrab
yellowgreen
darkolivegreen
greenyellow
chartreuse
lawngreen
honeydew
darkseagreen
palegreen
lightgreen
forestgreen
limegreen
darkgreen
g
green
lime
seagreen
mediumseagreen
springgreen
mintcream
mediumspringgreen
mediumaquamarine
aquamarine
turquoise
lightseagreen
mediumturquoise
azure
lightcyan
paleturquoise
darkslategray
darkslategrey
teal
darkcyan
c
aqua
cyan

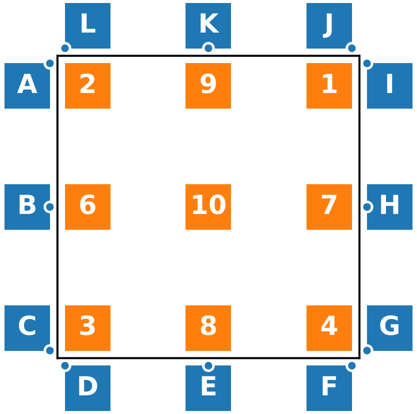
darkturquoise
cadetblue
powderblue
lightblue
deepskyblue
skyblue
lightskyblue
steelblue
aliceblue
dodgerblue
lightslategray
lightslategrey
slategray
slategrey
lightsteelblue
cornflowerblue
royalblue
ghostwhite
lavender
midnightblue
navy
darkblue
mediumblue
b
blue
slateblue
darkslateblue
mediumslateblue
mediumpurple
rebeccapurple
blueviolet
indigo
darkorchid
darkviolet
mediumorchid
thistle
plum
violet
purple
darkmagenta
m
fuchsia
magenta
orchid
mediumvioletred
deeppink
hotpink
lavenderblush
palevioletred
crimson
pink
lightpink

Image interpolation

API



Legend placement



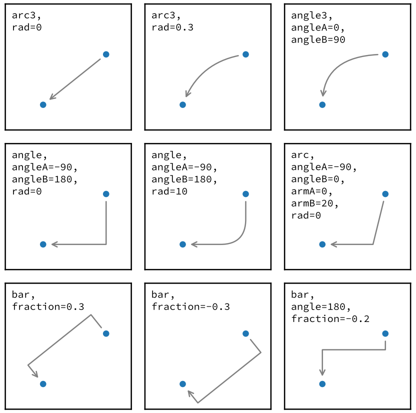
ax.**legend**(loc="string", bbox_to_anchor=(x,y))

2: upper left	9: upper center	1: upper right
6: center left	10: center	7: center right
3: lower left	8: lower center	4: lower right

A: upper right / `(-0.1,0.9)`	B: center right / `(-0.1,0.5)`
C: lower right / `(-0.1,0.1)`	D: upper left / `(0.1,-0.1)`
E: upper center / `(0.5,-0.1)`	F: upper right / `(0.9,-0.1)`
G: lower left / `(1.1,0.1)`	H: center left / `(1.1,0.5)`
I: upper left / `(1.1,0.9)`	J: lower right / `(0.9,1.1)`
K: lower center / `(0.5,1.1)`	L: lower left / `(0.1,1.1)`

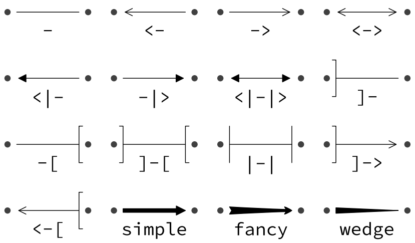
Annotation connection styles

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/an axes?
→ fig.clear() → ax.clear()
... close all figures?
→ plt.close("all")
... remove ticks?
→ ax.set_[xy]ticks([])
... remove tick labels ?
→ ax.set_[xy]ticklabels([])
... rotate tick labels ?
→ ax.tick_params(axis="x", 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

Matplotlib Cheatsheets

NUMFOCUS
OPEN CODE = BETTER SCIENCE