

Ouick start

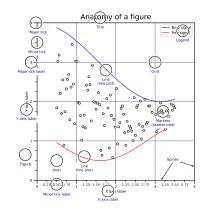
import numpy as np import matplotlib as mpl import matplotlib.pyplot as plt

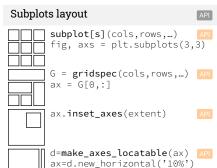
X = np.linspace(0, 2*np.pi, 100) Y = np.cos(X)

fig, ax = plt.subplots() ax.plot(X,Y,color='C1')

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

Anatomy of a figure





Getting help

matplotlib.org

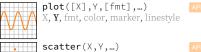
O discourse.matplotlib.org

₩ gitter.im/matplotlib

Matplotlib users mailing list

Basic plots

API

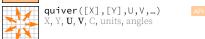


X, Y, [s]izes, [c]olors, markers, cmap

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

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











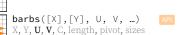
Advanced plots



<pre>errorbar(X,Y,xerr,yerr,)</pre>	API	
X, Y, xerr, yerr, fmt		













Tick locators

from matplotlib import ticker

ticker.MultipleLocator(0.5)

ticker.FixedLocator([0, 1, 5])

ticker.LinearLocator(numticks=3)

ticker.IndexLocator(base=0.5, offset=0.25)

ticker.LogLocator(base=10, numticks=15)

0.25 0.75 1.25 1.75 2.25 2.75 3.25 3.75 4.25

ax.[xy]axis.set_[minor|major]_formatter(formatter)

ticker.FuncFormatter(lambda x, pos: "[%.2f]" % x)

ticker.FixedFormatter(['', '0', '1', ...])

ticker.NullLocator()

ticker.AutoLocator()

Tick formatters

ticker.NullFormatter()

ticker.ScalarFormatter()

Ornaments

ax.legend(...)

Legend ←

ax.colorbar(...)

Event handling

from matplotlib import ticker

ticker.FormatStrFormatter('>%d<'

ticker.StrMethodFormatter('{x}')

ticker.PercentFormatter(xmax=5)

handles, labels, loc, title, frameon

_abel 1

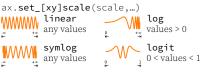
Label 2

mappable, ax, cax, orientation

0.3 0.4 0.5 0.6 0.7 0.8

ticker.MaxNLocator(n=4)

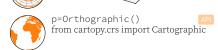
ax.[xy]axis.set [minor|major] locator(locator)



Projections subplot(...,projection=p) p='3d' p='polar'

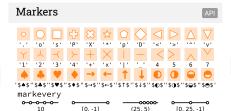
Scales

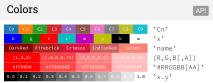
"butt"





"projecting"





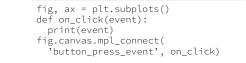
Colormaps

plt.get_cmap(name)

Cyclic







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()
```

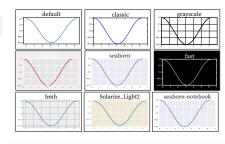
Styles

API

Label 3

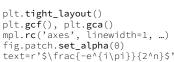
Label 4

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]()
```



X axis log/linear L Y axis log/linear

Keyboard shortcuts



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

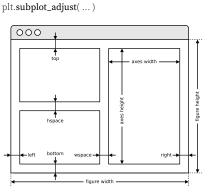
6. Use Color Effectively

7. Do Not Mislead the Reader

8. Avoid "Chartiunk"

9. Message Trumps Beauty 10. Get the Right Tool

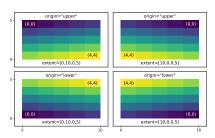




plt.imshow(extent=..., origin=...)

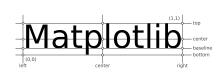
Extent & origin

Axes adjustements



Text alignments

plt.text(..., ha=... , va=..., ...)



Text parameters

The quick brown fox jumps over the lazy dog

plt.text(..., family=..., size=..., weight = ...) plt.text(..., fontproperties = ...)

The quick brown fox	xx-large	(1.73
The quick brown fox	x-large	(1.44
The guick brown fox	large	(1.20
The guick 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		
The quick brown fox jumps over the lazy dog bold		(700
The quick brown fox jumps over the lazy dog	semibolo	(600

The quick brown fox jumps over the lazy dog	ultralight (100)	
The quick brown fox jumps over the l The quick brown fox jumps over the lazy dog The quick brown fox jumps over the lazy dog The quick brown fox jumps over the lazy dog	azy dog monospace serif sans cursive	
The quick brown fox jumps over the lazy dog	italic	

The quick brown fox jumps over the lazy doe normal small-caps THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG The quick brown fox jumps over the lazy dog

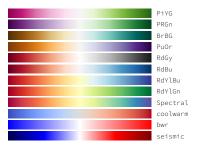
Uniform colormaps



Sequential colormaps



Diverging colormaps



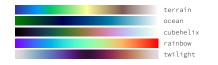
Qualitative colormaps

API

normal (400)

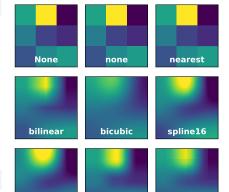


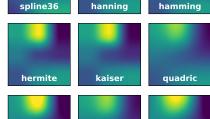
Miscellaneous colormaps

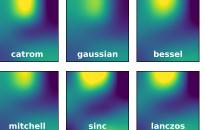


Color names cadetblue powderblue lightblue darkgoldenrod goldenrod dimgray dimgrey gray cornsilk gold deepskyblue lemonchiffon khaki grey darkgray skyblue lightskyblue darkgrey palegoldenrod pa. darkkn. ivory beige lightyellow lightgoldenrodyellor oflive y yellow 'wedrab 'reer aliceblue lightgray lightgrey gainsboro whitesmoke dodgerblue lightslategray slategray slategray slategrey lightsteelblue cornflowerblue yellow olivedrab yellowgreen darkolivegree, greenyellow chartreuse lawngreen honeydew darkser white white snow rosybrowr lightcoral indianred brown firebrick maroon darkred cornflowerblu royalblue ghostwhite lavender midnightblue honeydew darkseagreer palegreen lightgreen forestgreen blue slateblue darkslateblue mistyrose salmon tomato darksalmon limegreen darkgreen mediumpurple rebeccapurple orangered blueviolet lightsalmon indigo lime seagreen mediumseagreen sienna chocolate springgreen mintcream mediumorchid saddlebrown sandybrown mediumspringgreen plum peachpuff mediumaquama purple darkmagenta aguamarine turquoise lightseagreen mediumturquoise fuchsia darkorange burlywood antiquewhite tan navajowhite blanchedalmond azure lightcyan paleturquoise darkslategray darkslategrey magenta orchid orchid mediumvioletred deeppink hotpink lavenderblush palevioletred crimson teal darkcyan aqua cyan

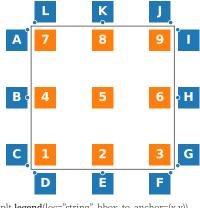
Image interpolation







Legend placement



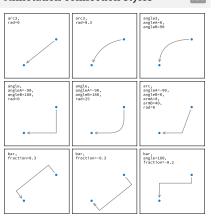
plt.legend(loc="string", bbox_to_anchor=(x,y))

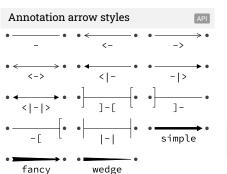
1: lower left	2: lower center	3: lower right
4: left	5: center	6: right
7: upper left	8: upper center	9: upper right

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,.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





How do I ...

- ... resize a figure? \rightarrow fig.set_size_inches(w,h)
- ... save a figure?
- → plt.savefig("figure.pdf")
- ... save a transparent figure? → plt.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_[xv]ticklabels([])
 - ... rotate tick labels?
 - \rightarrow plt.[xv]ticks(rotation=90)
 - ... hide top spine?
 - \rightarrow ax.spines['top'].set_visible(False)
 - ... hide legend border?
 - → plt.legend(frameon=False)
 - ... show error as shaded region?
 - \rightarrow ax.fill_between(X, Y+error, Y-error)
 - ... draw a rectangle?
 - \rightarrow ax.add_patch(plt.Rectangle((0, 0),1,1)
 - ... draw a vertical line?
 - \rightarrow ax.axvline(x=0.5)
 - ... draw outside frame?
 - \rightarrow ax.plot(..., clip_on=False)
 - ... use transparency?
 - \rightarrow ax.plot(..., alpha=0.25)
 - ... convert an RGB image into a gray image?
 - \rightarrow 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?
 - \rightarrow plt.get_cmap("viridis", 10)
 - ... show a figure for one second?
 - \rightarrow plt.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],[]))cla(), imshow(...), canvas.draw() slow im.set_data(...), canvas.draw() fast

Beyond Matplotlib

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

Matplotlib Cheatsheets (c) 2020 Nicolas P. Rougier Released under a CC-BY 4.0 International License

