Matplotlib 初级

Matplotlib是一个用于在Python中制作二维绘图的库。它的设计理念是：你应该只用几个命令就能创建简单的图形：

1 初始化

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

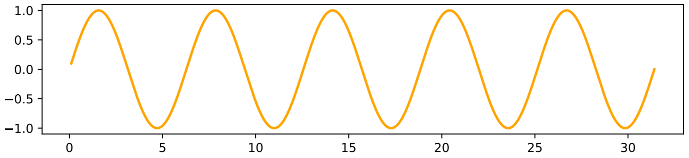
2 准备数据

```
X = np.linspace(0, 4*np.pi, 1000)
Y = np.sin(X)
```

3 渲染

```
fig, ax = plt.subplots()
ax.plot(X, Y)
fig.show()
```

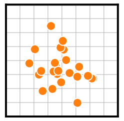
4 观察



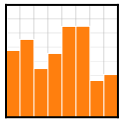
选择图形

Matplotlib提供了几种类型的图形（见图库）：

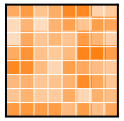
```
X = np.random.uniform(0, 1, 100)
Y = np.random.uniform(0, 1, 100)
ax.scatter(X, Y)
```



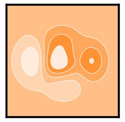
```
X = np.arange(10)
Y = np.random.uniform(1, 10, 10)
ax.bar(X, Y)
```



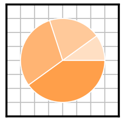
```
Z = np.random.uniform(0, 1, (8,8))

ax.imshow(Z)
```



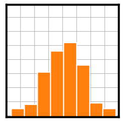
```
Z = np.random.uniform(0, 1, (8,8))

ax.contourf(Z)
```



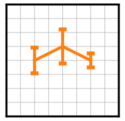
```
Z = np.random.uniform(0, 1, 4)

ax.pie(Z)
```



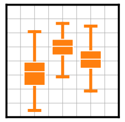
```
Z = np.random.normal(0, 1, 100)

ax.hist(Z)
```



```
X = np.arange(5)
Y = np.random.uniform(0, 1, 5)
ax.errorbar(X, Y, Y/4)
```



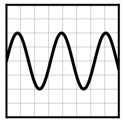
```
Z = np.random.normal(0, 1, (100,3))

ax.boxplot(Z)
```



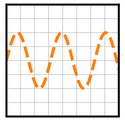
调整

你可以修改绘图中的几乎任何东西，包括限制、颜色、标记、线宽和样式、刻度线和刻度线标签、标题等等。

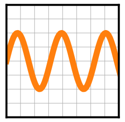
```
X = np.linspace(0, 10, 100)
Y = np.sin(X)
ax.plot(X, Y, color="black")
```



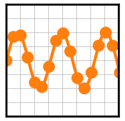
```
X = np.linspace(0, 10, 100)
Y = np.sin(X)
ax.plot(X, Y, linestyle="--")
```



```
X = np.linspace(0, 10, 100)
Y = np.sin(X)
ax.plot(X, Y, linewidth=5)
```



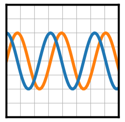
```
X = np.linspace(0, 10, 100)
Y = np.sin(X)
ax.plot(X, Y, marker="o")
```



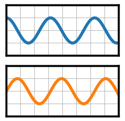
组织

你可以在同一个图上绘制多个数据，但你也可以将一个图分成几个子图（名为Axes）：

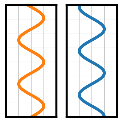
```
X = np.linspace(0, 10, 100)
Y1, Y2 = np.sin(X), np.cos(X)
ax.plot(X, Y1, X, Y2)
```



```
fig, (ax1, ax2) = plt.subplots((2,1))
ax1.plot(X, Y1, color="C1")
ax2.plot(X, Y2, color="C0")
```

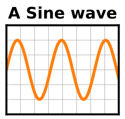


```
fig, (ax1, ax2) = plt.subplots((1,2))
ax1.plot(Y1, X, color="C1")
ax2.plot(Y2, X, color="C0")
```

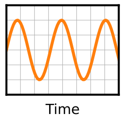


标记 (所有东西)

```
ax.plot(X, Y)
fig.suptitle(None)
ax.set_title("A Sine wave")
```



```
ax.plot(X, Y)
ax.set_ylabel(None)
ax.set_xlabel("Time")
```



探索

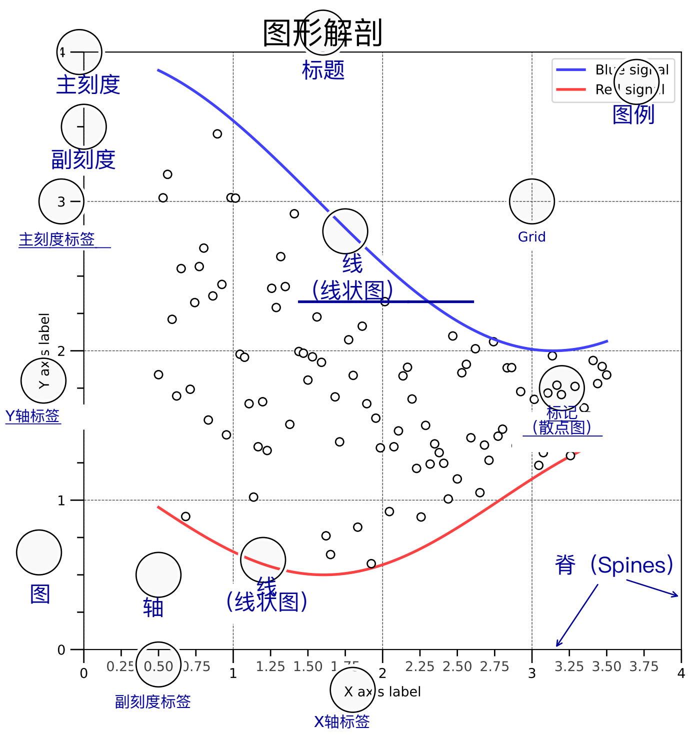
图形以图形用户界面显示，可以放大和平移，在不同的视图之间导航，并在鼠标下显示数值。

保存 (位图或矢量图)

```
fig.savefig("my-first-figure.png", dpi=300)
fig.savefig("my-first-figure.pdf")
```

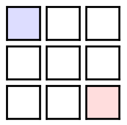
Matplotlib 中级

一个matplotlib图形是由构成实际图形的元素的层次结构组成的。每个元素都可以被修改。

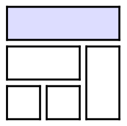


Figure, axes & spines

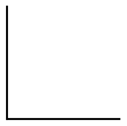
```
fig, axs = plt.subplots((3,3))
axs[0,0].set_facecolor("#ddddff")
axs[2,2].set_facecolor("#ffffdd")
```



```
gs = fig.add_gridspec(3, 3)
ax = fig.add_subplot(gs[0, :])
ax.set_facecolor("#ddddff")
```

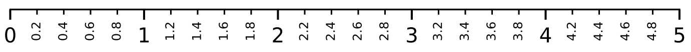


```
fig, ax = plt.subplots()
ax.spines["top"].set_color("None")
ax.spines["right"].set_color("None")
```



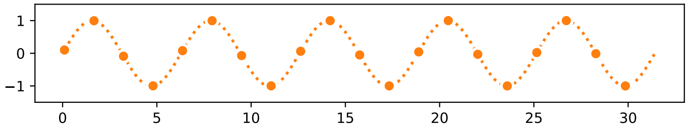
刻度与标签

```
from mpl.ticker import MultipleLocator as ML
from mpl.ticker import ScalarFormatter as SF
ax.xaxis.set_minor_locator(ML(0.2))
ax.xaxis.set_minor_formatter(SF())
ax.tick_params(axis='x',which='minor',rotation=90)
```



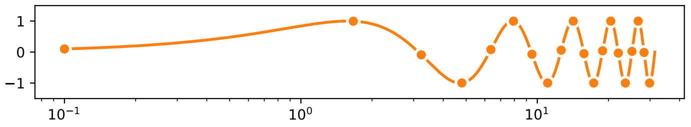
线条与标记

```
X = np.linspace(0.1, 10*np.pi, 1000)
Y = np.sin(X)
ax.plot(X, Y, "C1o:", markevery=25, mec="1.0")
```



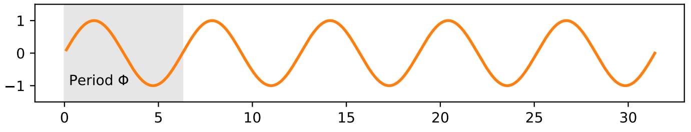
缩放与投影

```
fig, ax = plt.subplots()
ax.set_xscale("log")
ax.plot(X, Y, "C1o-", markevery=25, mec="1.0")
```



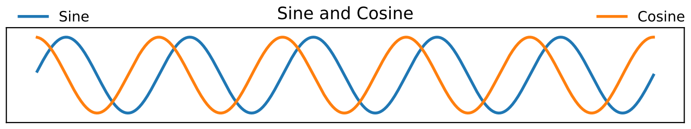
文字与装饰

```
ax.fill_betweenx([-1,1],[0],[2*np.pi])
ax.text(0, -1, r" Period $\Phi$")
```



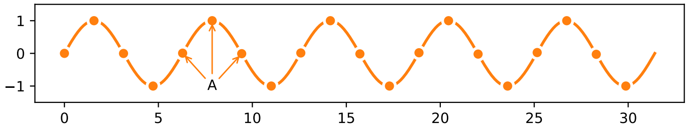
图例

```
ax.plot(X, np.sin(X), "C0", label="Sine")
ax.plot(X, np.cos(X), "C1", label="Cosine")
ax.legend(bbox_to_anchor=(0,1,1,.1),ncol=2,
          mode="expand",  loc="lower left")
```



标注

```
ax.annotate("A", (X[250],Y[250]),(X[250],-1),
  ha="center", va="center",arrowprops =
  {"arrowstyle" : "->", "color": "C1"})
```



颜色

任何颜色都可以使用，但Matplotlib提供了一些颜色集：

C0	C1	C2	C3	C4	C5	C6	C7	C8	C9

0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0

尺寸与DPI

考虑将一个正方形的图形放在一张两列的A4纸上，每边的页边距为2厘米，列距为1厘米。图的宽度是（21-2*2-1）/ 2=8厘米。一英寸为2.54厘米，图的大小应该是3.15×3.15英寸。

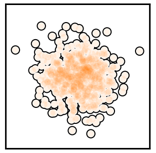
```
fig = plt.figure(figsize=(3.15,3.15), dpi=50)
plt.savefig("figure.pdf", dpi=600)
```

Matplotlib 提示与技巧

透明度

Scatter plots can be enhanced by using transparency (alpha) in order to show area with higher density. Multiple scatter plots can be used to delineate a frontier.

```
X = np.random.normal(-1, 1, 500)
Y = np.random.normal(-1, 1, 500)
ax.scatter(X, Y, 50, "0.0", lw=2) # optional
ax.scatter(X, Y, 50, "1.0", lw=0) # optional
ax.scatter(X, Y, 40, "C1",  lw=0, alpha=0.1)
```



光栅化

If your figure has many graphical elements, such as a huge scatter, you can rasterize them to save memory and keep other elements in vector format.

```
X = np.random.normal(-1, 1, 10_000)
Y = np.random.normal(-1, 1, 10_000)
ax.scatter(X, Y, rasterized=True)
fig.savefig("rasterized-figure.pdf", dpi=600)
```

离线渲染

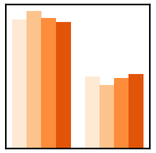
使用Agg后端直接在数组中渲染一个图形。

```
from matplotlib.backends.backend_agg import FigureCanvas
canvas = FigureCanvas(Figure()))
... # draw som stuff
canvas.draw()
Z = np.array(canvas.renderer.buffer_rgba())
```

连续颜色范围

你可以使用colormap从一系列的连续颜色中选择。

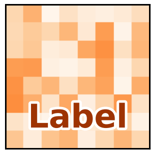
```
X = np.random.randn(1000, 4)
cmap = plt.get_cmap("Oranges")
colors = cmap([0.2, 0.4, 0.6, 0.8])

ax.hist(X, 2, histtype='bar', color=colors)
```



文本大纲

使用文本大纲来使文字更显眼

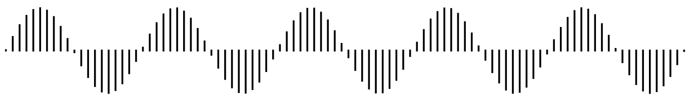
```
import matplotlib.patheffects as fx
text = ax.text(0.5, 0.1, "Label")
text.set_path_effects([
  fx.Stroke(linewidth=3, foreground='1.0'),
  fx.Normal()])
```



多线图

你可以用None作为分隔符，一次绘制多条线。

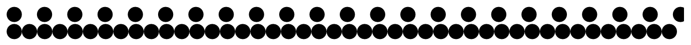
```
X,Y = [], []
for x in np.linspace(0, 10*np.pi, 100):
  X.extend([x, x, None]), Y.extend([0, sin(x), None])
ax.plot(X, Y, "black")
```



点状线

若要使用圆形点状线，使用自定义的linestyle，更改dash_capstyle

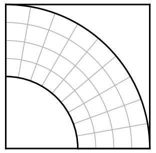
```
ax.plot([0,1], [0,0], "C1",
        linestyle = (0, (0.01, 1)), dash_capstyle="round")
ax.plot([0,1], [1,1], "C1",
        linestyle = (0, (0.01, 2)), dash_capstyle="round")
```



组合轴

你可以使用不同投影的叠加轴

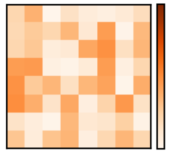
```
ax1 = fig.add_axes([0,0,1,1],
                   label="cartesian")
ax2 = fig.add_axes([0,0,1,1],
                   label="polar",
                   projection="polar")
```



色条调整

你可以在添加色条时调整它的大小。

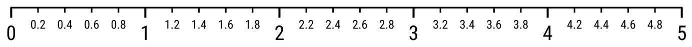
```
im = ax.imshow(Z)

cb = plt.colorbar(im,
        fraction=0.046, pad=0.04)
cb.set_ticks([])
```



利用排版的优势

你可以使用一种紧密的字体，如Roboto Condensed，以节省刻度标签的空间。

```
for tick in ax.get_xticklabels(which='both'):
    tick.set_fontname("Roboto Condensed")
```



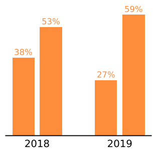
去除边界

一旦你的图完成了，你可以调用tight_layout()来删除白边。如果有剩余的边距， 你可以使用pdfcrop工具（TeX live自带）。

Hatching

You can achieve a nice visual effect with thick hatch patterns.

```
cmap = plt.get_cmap("Oranges")
plt.rcParams['hatch.color'] = cmap(0.2)
plt.rcParams['hatch.linewidth'] = 8
ax.bar(X, Y, color=cmap(0.6), hatch="/" )
```



阅读文档

Matplotlib提供了大量的文档，解释了每个命令的细节，并且一般都附有例子。再加上庞大的在线图库，这些文档就是一座金矿。

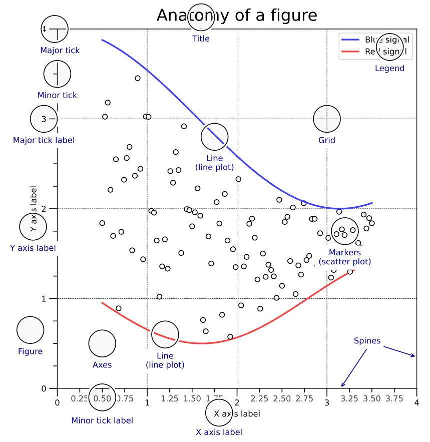
matplotlib 小抄

版本 3.4.2

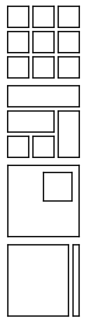
快速开始 API

```
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='C1')

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

图形解剖



子图布局 API



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

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

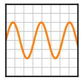
`ax.inset_axes(extent)` API

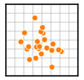
`d=make_axes_locatable(ax)` API
`ax=d.new_horizontal('10%')`

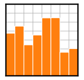
获得帮助

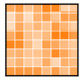
- matplotlib.org
- github.com/matplotlib/matplotlib/issues
- discourse.matplotlib.org
- stackoverflow.com/questions/tagged/matplotlib
- gitter.im/matplotlib
- twitter.com/matplotlib
- Matplotlib users mailing list

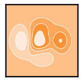
基本图形

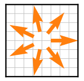

`plot([X],Y,[fmt],…)` API
X, Y, fmt, color, marker, linestyle

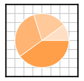

`scatter(X,Y,…)` API
X, Y, [s]izes, [c]olors, marker, cmap

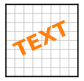

`bar[h](x,height,…)` API
x, height, width, bottom, align, color


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

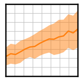

`contour[f]([X],[Y],Z,,…)` API
X, Y, Z, levels, colors, extent, origin


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

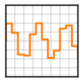

`pie(X,[explode],…)` API
Z, explode, labels, colors, radius

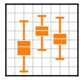


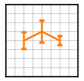
`text(x,y,text,…)` API
x, y, text, va, ha, size, weight, transform

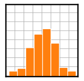

`fill[_between][x]( … )` API
X, Y1, Y2, color, where

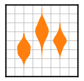
Advanced plots

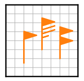

`step(X,Y,[fmt],…)` API
X, Y, fmt, color, marker, where

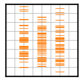

`boxplot(X,…)` API
X, notch, sym, bootstrap, widths

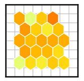

`errorbar(X,Y,xerr,yerr,…)` API
X, Y, xerr, yerr, fmt

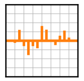

`hist(X, bins, …)` API
X, bins, range, density, weights


`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

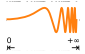

`hexbin(X,Y,C,…)` API
X, Y, C, gridsize, bins

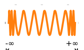

`xcorr(X,Y,…)` API
X, Y, normed, detrend

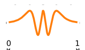
Scales API

`ax.set_[xy]scale(scale,…)`

`linear`
any values

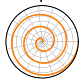

`log`
values > 0

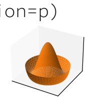

`symlog`
any values


`logit`
0 < values < 1

Projections API

`subplot(…,projection=p)`


`p='polar'`

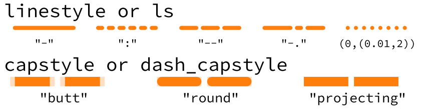

`p='3d'`

`p=Orthographic()` API
`from cartopy.crs import Cartographic`

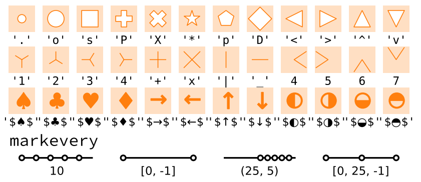
Lines API

`linestyle or ls`

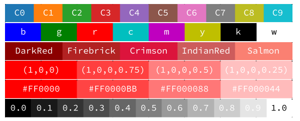
`capstyle or dash_capstyle`



Markers API



Colors API

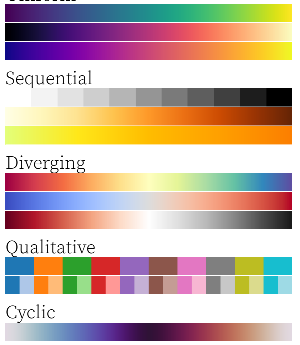


C0 C1 C2 C3 C4 C5 C6 C7 C8 C9
b g r c m y k w
DarkRed Firebrick Crimson IndianRed Salmon
(1,0,0) (1,0,0,0.75) (1,0,0,0.5) (1,0,0,0.25)
#FF0000 #FF0000BB #FF000088 #FF000044
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

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

Colormaps API

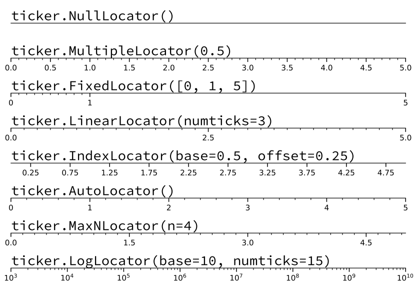
`plt.get_cmap(name)`



viridis
magma
plasma
Greys
YlOrBr
Wistia
Spectral
coolwarm
RdGy
tab10
tab20
twilight

Tick locators API

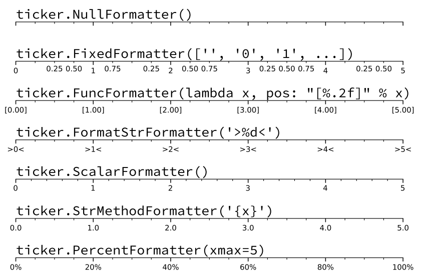
```
from matplotlib import ticker
ax.[xy]axis.set_[minor|major]_locator(locator)
```



```
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

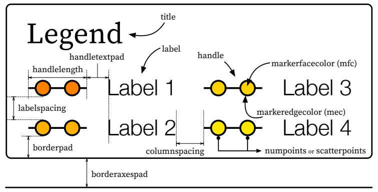
```
from matplotlib import ticker
ax.[xy]axis.set_[minor|major]_formatter(formatter)
```



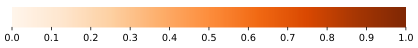
```
ticker.NullFormatter()
ticker.FixedFormatter(['', '0', '1', …])
ticker.FuncFormatter(lambda x, pos: "[%.2f]" % x)
ticker.FormatStrFormatter('>%d<')
ticker.ScalarFormatter()
ticker.StrMethodFormatter('{x}')
ticker.PercentFormatter(xmax=5)
```

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

Annotation
xytext
textcoords
xy
ycoords

Event handling API

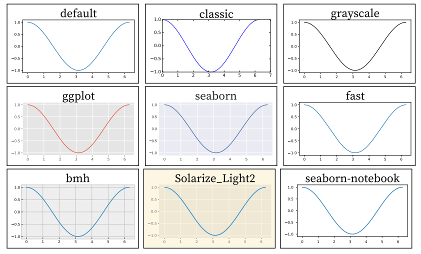
```
fig, ax = plt.subplots()
def on_click(event):
  print(event)
fig.canvas.mpl_connect(
  'button_press_event', on_click)
```

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.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]()

fig.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 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

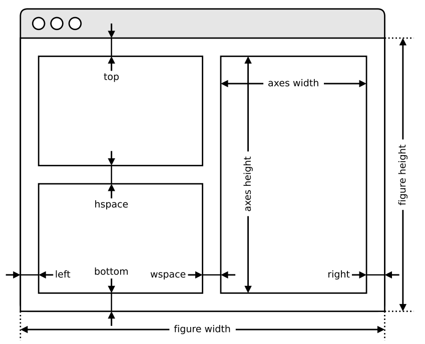
Ten simple rules READ

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

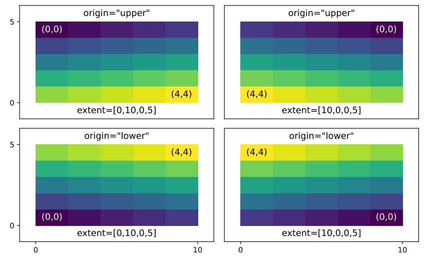
plt.subplots_adjust(…)



Extent & origin

API

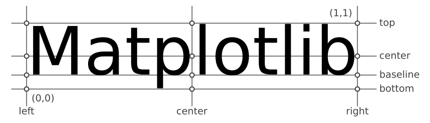
ax.imshow(extent=…, origin=…)



Text alignments

API

ax.text(…, ha=… , va=…, …)



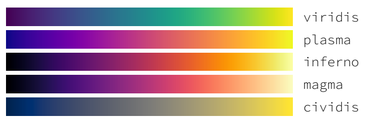
Text parameters

API

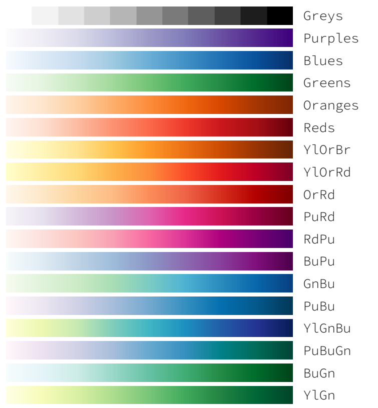
ax.text(…, family=… , size=…, weight = …)
ax.text(…, fontproperties = …)

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

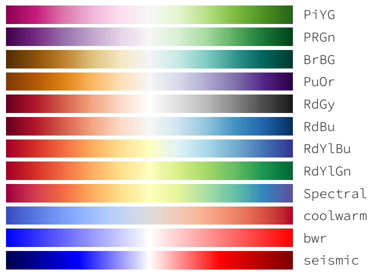
Uniform colormaps



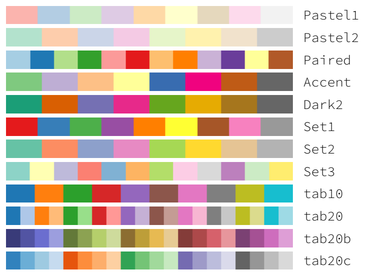
Sequential colormaps



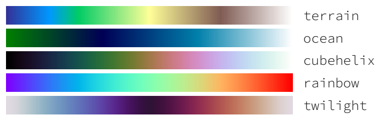
Diverging colormaps



Qualitative colormaps



Miscellaneous colormaps



Color names

API

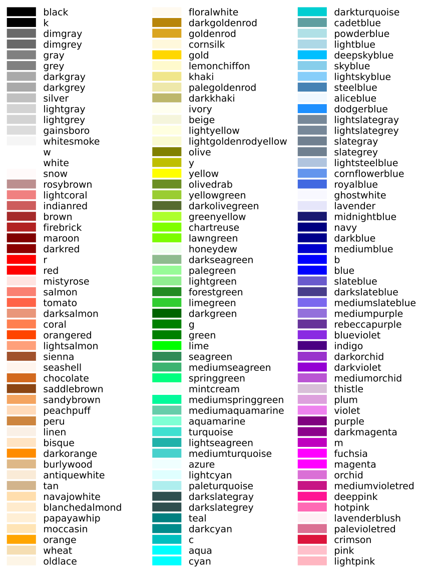
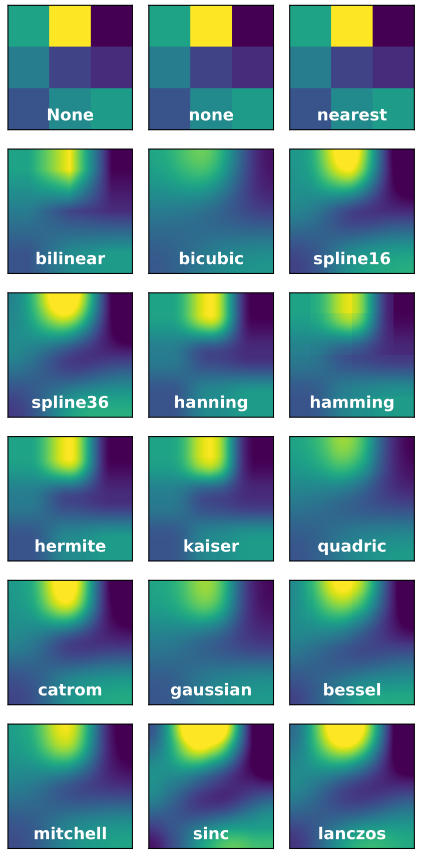
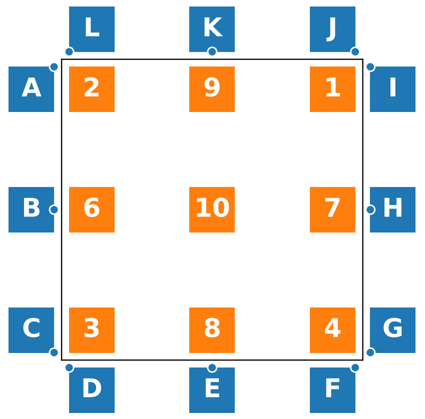


Image interpolation

API



Legend placement



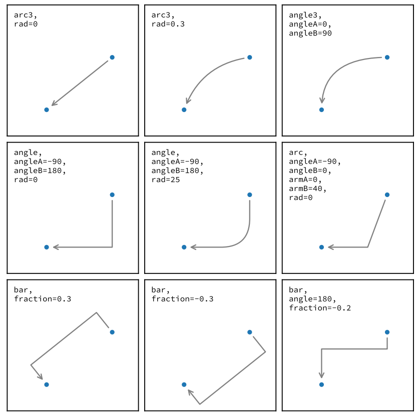
ax.legend(loc="string", bbox_to_anchor=(x,y))

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)`

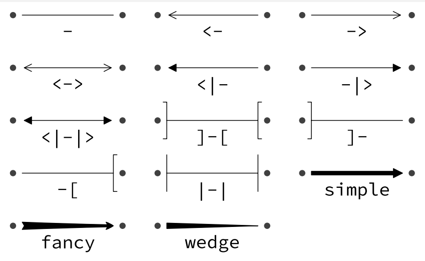
Annotation connection styles

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?
→ 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