

## Quick start

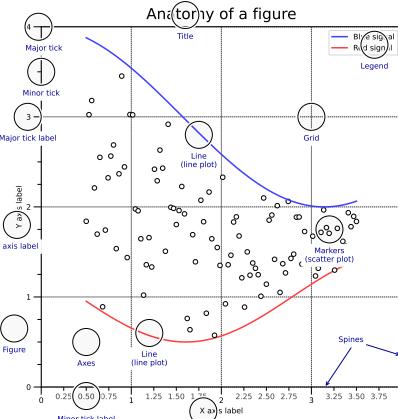
```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt

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

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

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

## Anatomy of a figure



## Subplots layout

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

G = gridspec(rows, cols, ...)
ax = G[0, :]

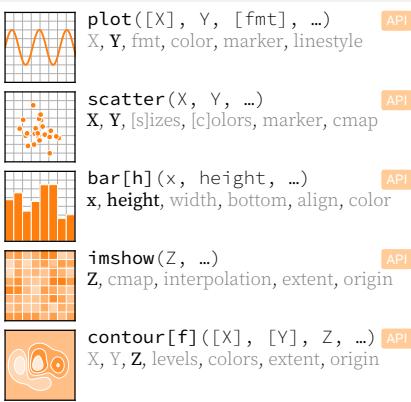
ax.inset_axes(extent)

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

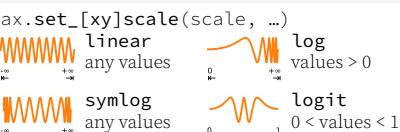
## Getting help

- [matplotlib.org](https://matplotlib.org)
- [github.com/matplotlib/matplotlib/issues](https://github.com/matplotlib/matplotlib/issues)
- [discourse.matplotlib.org](https://discourse.matplotlib.org)
- [stackoverflow.com/questions/tagged/matplotlib](https://stackoverflow.com/questions/tagged/matplotlib)
- <https://gitter.im/matplotlib/matplotlib>
- [twitter.com/matplotlib](https://twitter.com/matplotlib)
- [Matplotlib users mailing list](mailto:Matplotlib users mailing list)

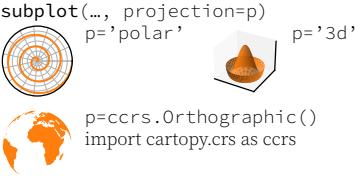
## Basic plots



## Scales



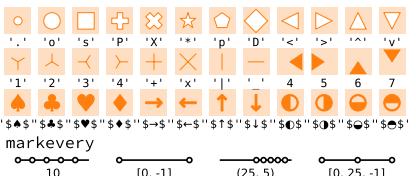
## Projections



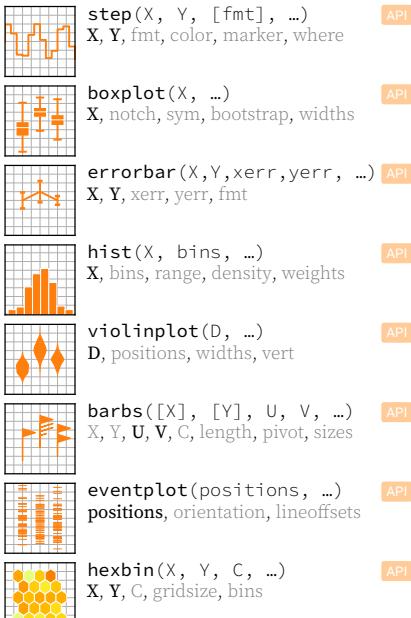
## Lines



## Markers



## Advanced plots



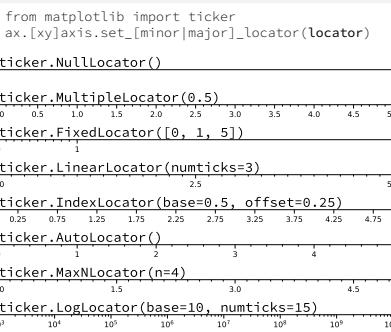
## Colors



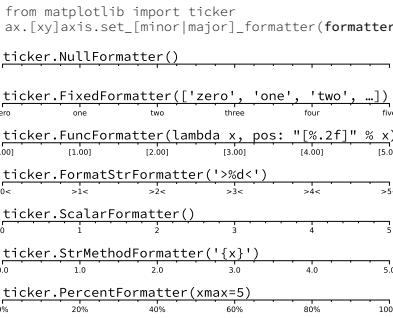
## Colormaps



## Tick locators

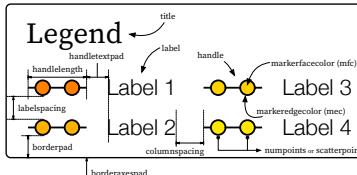


## Tick formatters



## Ornaments

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



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

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

```
text
xytext
textcoords
xy
xycoords
```

## Event handling

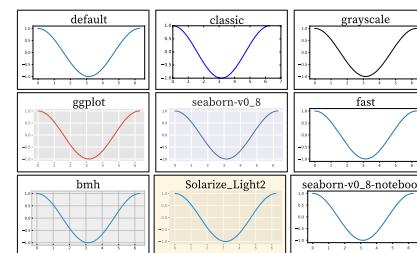
```
fig, ax = plt.subplots()
def on_click(event):
    print(event)
fig.canvas.mpl_connect('button_press_event', on_click)
```

## Animation

```
import matplotlib.animation as mpl
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 = mpl.FuncAnimation(
    plt.gcf(), animate, interval=5)
plt.show()
```

## Styles

```
plt.style.use(style)
```



## Quick reminder

```
ax.grid()
ax.set_xyllim(vmin, vmax)
ax.set_xylabel(label)
ax.set_xyticks(ticks, [labels])
ax.set_xyticklabels(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'$\frac{-1}{pi}2^n$'
```

## Keyboard shortcuts

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

## Ten simple rules

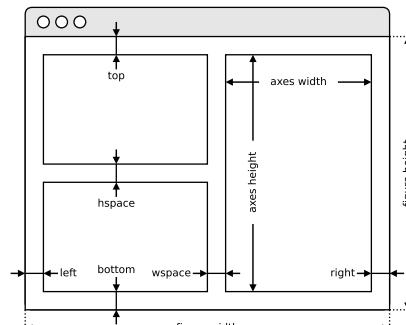
1. Know your audience
2. Identify your message
3. Adapt the figure
4. Captions are not optional
5. Do not trust the defaults
6. Use color effectively
7. Do not mislead the reader
8. Avoid "chartjunk"
9. Message trumps beauty
10. Get the right tool

READ

## Axes adjustments

API

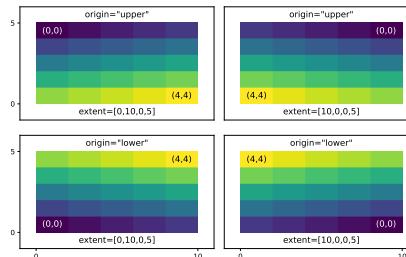
```
plt.subplots_adjust(...)
```



## Extent & origin

API

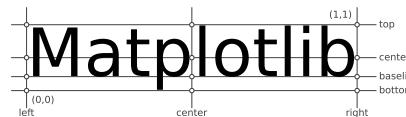
```
ax.imshow(extent=..., origin=...)
```



## 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

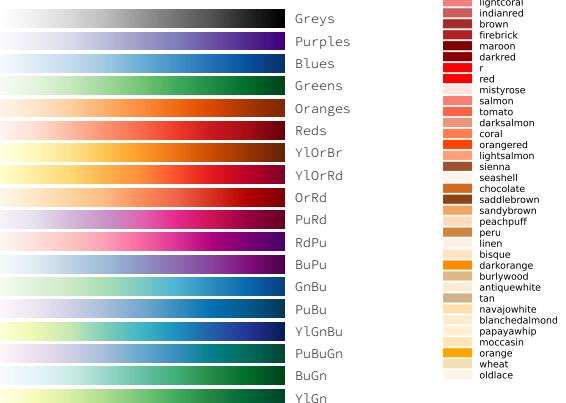
The quick brown fox jumps over the lazy dog

normal

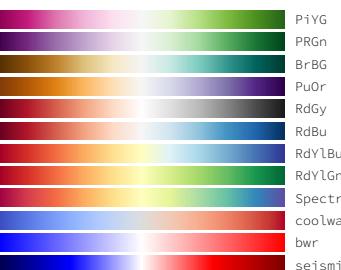
## Uniform colormaps



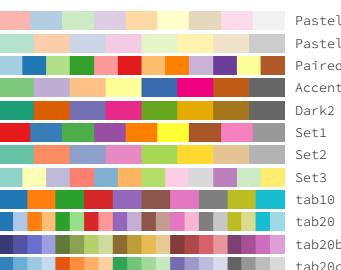
## Sequential colormaps



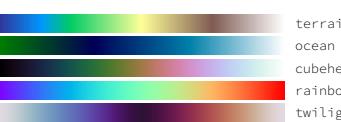
## Diverging colormaps



## Qualitative colormaps



## Miscellaneous colormaps



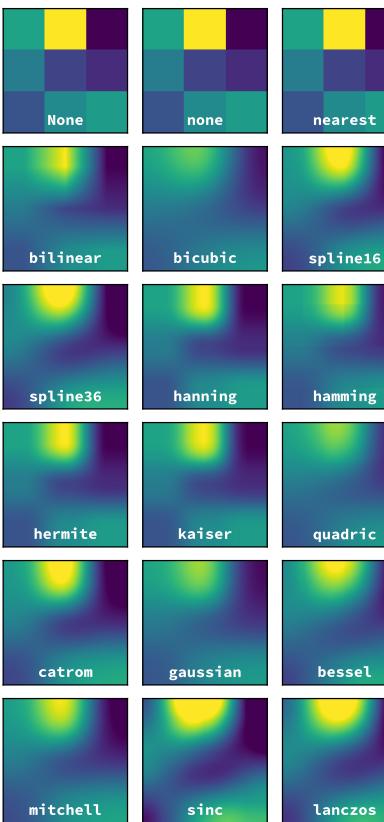
## Color names

API

black	floralwhite	darkturquoise	lightgray
black	darkgoldenrod	cadetblue	powderblue
dimgray	cornsilk	steelblue	lightblue
dimgray	gold	steelblue	steelblue
gray	lemonchiffon	steelblue	steelblue
gray	khaki	steelblue	steelblue
darkgray	palegoldenrod	steelblue	steelblue
darkgray	darkskhaki	steelblue	steelblue
silver	lightgoldenrodyellow	steelblue	steelblue
lightgray	lightyellow	steelblue	steelblue
lightgray	lightgoldenrodyellow	steelblue	steelblue
lightgray	olivedrab	steelblue	steelblue
lightgray	yellowgreen	steelblue	steelblue
lightgray	greenyellow	steelblue	steelblue
lightgray	chartreuse	steelblue	steelblue
lightgray	lawngreen	steelblue	steelblue
lightgray	honeydew	steelblue	steelblue
w	darkseagreen	steelblue	steelblue
w	lightgreen	steelblue	steelblue
white	palegreen	steelblue	steelblue
white	lightgreen	steelblue	steelblue
white	yellowgreen	steelblue	steelblue
white	limegreen	steelblue	steelblue
white	darkgreen	steelblue	steelblue
r	darkgreen	steelblue	steelblue
red	red	steelblue	steelblue
red	firebrick	steelblue	steelblue
red	darkred	steelblue	steelblue
red	darkred	steelblue	steelblue
red	red	steelblue	steelblue
red	firebrick	steelblue	steelblue
red	darkred	steelblue	steelblue
YLorBr	YLorBr	steelblue	steelblue
YLorRd	YLorRd	steelblue	steelblue
OrRd	OrRd	steelblue	steelblue
PuRd	PuRd	steelblue	steelblue
RdPu	RdPu	steelblue	steelblue
BuPu	BuPu	steelblue	steelblue
GrBu	GrBu	steelblue	steelblue
PuBu	PuBu	steelblue	steelblue
YlGnBu	YlGnBu	steelblue	steelblue
PuBuGn	PuBuGn	steelblue	steelblue
BuGn	BuGn	steelblue	steelblue
YlGn	YlGn	steelblue	steelblue

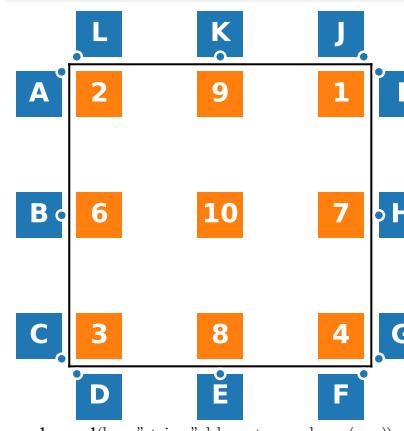
## Image interpolation

API



## Legend placement

API



```
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)

## 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(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)

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

slow

```
for i in range(n): plot(i, X[i], "o", ls="")
```

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

Copyright (c) 2021 Matplotlib Development Team

Released under a CC-BY 4.0 International License

**NUMFOCUS**  
OPEN CODE = BETTER SCIENCE