

#### Quick start

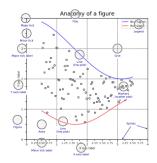
import numpy as np import matplotlib as mpl import matplotlib.pyplot as plt

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,...) [AP]

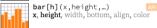


#### Getting help

- matplotlib.org
- □ discourse.matplotlib.org
- gitter.im/matplotlib
- witter.com/matplotlib
   twitter.com/matplotlib
- Matplotlib users mailing list

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

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



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



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,...) [API] 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

symlog any values



log

# Projections

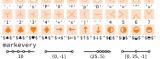
subplot(...,projection=p) p='polar'



# Lines



# Markers



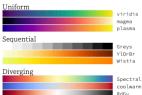
# Colors

API



#### Colormaps

plt.get\_cmap(name)



tab20

# Tick locators

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

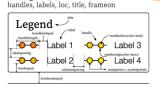
from matplotlib import ticker ax.[xy]axis.set\_[minor|major]\_formatter(formatter) ticker.NullFormatter()

ticker.FixedFormatter(['', '0', '1', ...]) ticker.FuncFormatter(lambda x, pos: "[%.2f]" % x) ticker.FormatStrFormatter('>%d<')

ticker.StrMethodFormatter('{x}') ticker.PercentFormatter(xmax=5)

# Ornaments

ax.legend(...)



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

API

0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9

text, xy, xytext, xycoords, textcoords, arrowprops

Annotation

# Event handling

fig, ax = plt.subplots()
def on\_click(event): print(event) fig.canvas.mpl\_connect( button\_press\_event', on\_click) Animation

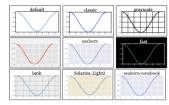
import matplotlib.animation as mpla

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)

#### Styles

plt.show()

plt.style.use(style)



API

#### 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 X axis log/linear L Y axis log/linear

#### Ten Simple Rules

- 1. Know Your Audience
- Identify Your Message Adapt the Figure
- 4. Captions Are Not Optional
- 5. Do Not Trust the Defaults
- 6. Use Color Effectively
- Do Not Mislead the Reader
- Avoid "Chartjunk"
- Message Trumps Beauty
- 10. Get the Right Tool