Matplotlib for beginners

Matplotlib is a library for making 2D plots in Python. It is designed with the philosophy that you should be able to create simple plots with just a few commands:

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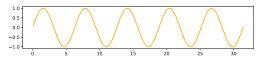
```
import numpy as np
import matplotlib.pyplot as plt
```

2 Prepare

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

3 Render

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

4 Observe

Choose

Matplotlib offers several kind of plots (see Gallery):

```
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)
```

Tweak

You can modify pretty much anything in a plot, including limits, colors, markers, line width and styles, ticks and ticks labels, titles, etc.

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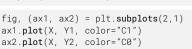
```
X = np.linspace(0, 10, 100)
Y = np.sin(X)
ax.plot(X, Y, linewidth=5)
```

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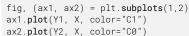
Organize

You can plot several data on the the same figure, but you can also split a figure in several subplots (named *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")
```
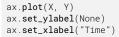
Label (everything)

```
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")
```

Explore

Figures are shown with a graphical user interface that allows to zoom and pan the figure, to navigate between the different views and to show the value under the mouse.

Save (bitmap or vector format)

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

matplotlib
Cheat sheet
Version 3.5.0

Quick start `API`

```python
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")
fig.show()
```

Anatomy of a figure

Anatomy of a figure

Subplots layout `API`

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

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

```python
ax.inset_axes(extent)
```

```python
d=make_axes_locatable(ax)
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`

```python
ax.set_[xy]scale(scale,…)
```

linear any values	**log** values > 0
symlog any values	**logit** 0 < values < 1

Projections `API`

```python
subplot(…,projection=p)
```
p='polar' p='3d'

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

Lines `API`

linestyle or **ls**

capstyle or **dash_capstyle**
"butt" "round" "projecting"

Markers `API`

'.' '0' 's' 'P' 'X' '*' 'p' 'D' '<' '>' '^' 'v'
'1' '2' '3' '4' '+' 'x' '|' '_' 4 5 6 7
'$♣$''$♦$''$♥$''$♠$''$♦$''$←$''$→$''$↑$''$↓$''O''O''O''O'

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

Colors `API`

C0 C1 C2 C3 C4 C5 C6 C7 C8 C9   'Cn'
                                  'x'
DarkRed Firebrick Crimson IndianRed Salmon   'name'
                                  (R,G,B[,A])
#FF0000 #FF0000B #FF0000BB #FF0000B   '#RRGGBB[AA]'
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 1.0   'x.y'

Colormaps `API`

```python
plt.get_cmap(name)
```

Uniform
viridis
magma
plasma

Sequential
Greys
YlOrBr
Wistia

Diverging
Spectral
coolwarm
RdGy

Qualitative
tab10
tab20

Cyclic
twilight

Tick locators `API`

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

```python
ticker.NullLocator()
```

```python
ticker.MultipleLocator(0.5)
```

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

```python
ticker.LinearLocator(numticks=3)
```

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

```python
ticker.AutoLocator()
```

```python
ticker.MaxNLocator(n=4)
```

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

Tick formatters `API`

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

```python
ticker.NullFormatter()
```

```python
ticker.FixedFormatter(['zero', 'one', 'two', …])
```

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

```python
ticker.FormatStrFormatter('>%d<')
```

```python
ticker.ScalarFormatter()
```

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

```python
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`

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

Animation `API`

```python
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`

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

default	classic	grayscale
ggplot	seaborn	fast
bmh	Solarize_Light2	seaborn-notebook

Quick reminder

```python
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

plt.**subplots_adjust**(…)

Extent & origin

ax.**imshow**(extent=…, origin=…)

Text alignments

ax.**text**(…, ha=… , va=…, …)

Matplotlib

Text parameters

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

viridis
plasma
inferno
magma
cividis

Sequential colormaps

Greys
Purples
Blues
Greens
Oranges
Reds
YlOrBr
YlOrRd
OrRd
PuRd
RdPu
BuPu
GnBu
PuBu
YlGnBu
PuBuGn
BuGn
YlGn

Diverging colormaps

PiYG
PRGn
BrBG
PuOr
RdGy
RdBu
RdYlBu
RdYlGn
Spectral
coolwarm
bwr
seismic

Qualitative colormaps

Pastel1
Pastel2
Paired
Accent
Dark2
Set1
Set2
Set3
tab10
tab20
tab20b
tab20c

Miscellaneous colormaps

terrain
ocean
cubehelix
rainbow
twilight

Color names

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

Legend placement

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

2: upper left	9: upper center
6: center left	10: center
3: lower left	8: lower center

1: upper right	
7: center right	
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

Annotation arrow styles

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

NUMFOCUS
OPEN CODE = BETTER SCIENCE