Search the docs ...

matplotlib
matplotlib.afm
matplotlib.animation
matplotlib.artist
matplotlib.axes
matplotlib.axis
matplotlib.backend_bases
matplotlib.backend_managers
matplotlib.backend_tools
matplotlib.backends
matplotlib.bezier
matplotlib.blocking_input
matplotlib.category
matplotlib.cbook
matplotlib.cm
matplotlib.collections
matplotlib.colorbar
matplotlib.colors
matplotlib.container
matplotlib.contour
matplotlib.dates
matplotlib.docstring
matplotlib.dviread
matplotlib.figure
matplotlib.font_manager
matplotlib.fontconfig_pattern
matplotlib.gridspec
matplotlib.image
matplotlib.legend

API Reference

When using the library you will typically create Figure and Axes objects and call their methods to add content and modify the appearance.

- matplotlib.figure: axes creation, figure-level content
- matplotlib.axes: most plotting methods, Axes labels, access to axis styling, etc.

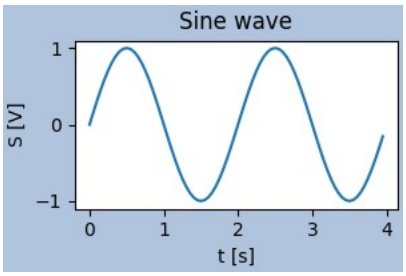
Example: We create a Figure `fig` and Axes `ax`. Then we call methods on them to plot data, add axis labels and a figure title.

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

x = np.arange(0, 4, 0.05)
y = np.sin(x*np.pi)

fig, ax = plt.subplots(figsize=(3,2), constrained_layout=True)
ax.plot(x, y)
ax.set_xlabel('t [s]')
ax.set_ylabel('S [V]')
ax.set_title('Sine wave')
fig.set_facecolor('lightsteelblue')
```

(Source code, png, pdf)



Modules

Alphabetical list of modules:

- matplotlib
- matplotlib.afm
- matplotlib.animation
- matplotlib.artist
- matplotlib.axes
- matplotlib.axis
- matplotlib.backend_bases
- matplotlib.backend_managers
- matplotlib.backend_tools
- matplotlib.backends
- matplotlib.bezier
- matplotlib.blocking_input
- matplotlib.category
- matplotlib.cbook
- matplotlib.cm
- matplotlib.collections
- matplotlib.colorbar
- matplotlib.colors
- matplotlib.container

- matplotlib.contour
- matplotlib.dates
- matplotlib.docstring
- matplotlib.dviread
- matplotlib.figure
- matplotlib.font_manager
- matplotlib.fontconfig_pattern
- matplotlib.gridspec
- matplotlib.image
- matplotlib.legend
- matplotlib.legend_handler
- matplotlib.lines
- matplotlib.markers
- matplotlib.mathtext
- matplotlib.mlab
- matplotlib.offsetbox
- matplotlib.patches
- matplotlib.path
- matplotlib.patheffects
- matplotlib.pyplot
- matplotlib.projections
- matplotlib.quiver
- matplotlib.rcsetup
- matplotlib.sankey
- matplotlib.scale
- matplotlib.sphinxext.mathmpl
- matplotlib.sphinxext.plot_directive
- matplotlib.spines
- matplotlib.style
- matplotlib.table
- matplotlib.testing
- matplotlib.text
- matplotlib.texmanager
- matplotlib.textpath
- matplotlib.ticker
- matplotlib.tight_bbox
- matplotlib.tight_layout
- matplotlib.transforms
- matplotlib.tri
- matplotlib.type1font
- matplotlib.units
- matplotlib.widgets
- matplotlib._api
- matplotlib._enums
- mpl_toolkits.mplot3d
- mpl_toolkits.axes_grid1
- mpl_toolkits.axisartist
- mpl_toolkits.axes_grid

Usage patterns

Below we describe several common approaches to plotting with Matplotlib.

The pyplot API

matplotlib.pyplot is a collection of functions that make Matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

`pyplot` is mainly intended for interactive plots and simple cases of programmatic plot generation.

Further reading:

- The `matplotlib.pyplot` function reference
- Pyplot tutorial
- Pyplot examples

The object-oriented API

At its core, Matplotlib is object-oriented. We recommend directly working with the objects, if you need more control and customization of your plots.

In many cases you will create a `Figure` and one or more `Axes` using `pyplot.subplots` and from then on only work on these objects. However, it's also possible to create `Figure`s explicitly (e.g. when including them in GUI applications).

Further reading:

- **`matplotlib.axes.Axes` and `matplotlib.figure.Figure` for an overview of**
  plotting functions.
- **Most of the examples use the object-oriented approach**
  (except for the pyplot section)

The pylab API (discouraged)

> ⚠ **Warning**
>
> Since heavily importing into the global namespace may result in unexpected behavior, the use of pylab is strongly discouraged. Use `matplotlib.pyplot` instead.

`pylab` is a module that includes `matplotlib.pyplot`, `numpy`, `numpy.fft`, `numpy.linalg`, `numpy.random`, and some additional functions, all within a single namespace. Its original purpose was to mimic a MATLAB-like way of working by importing all functions into the global namespace. This is considered bad style nowadays.