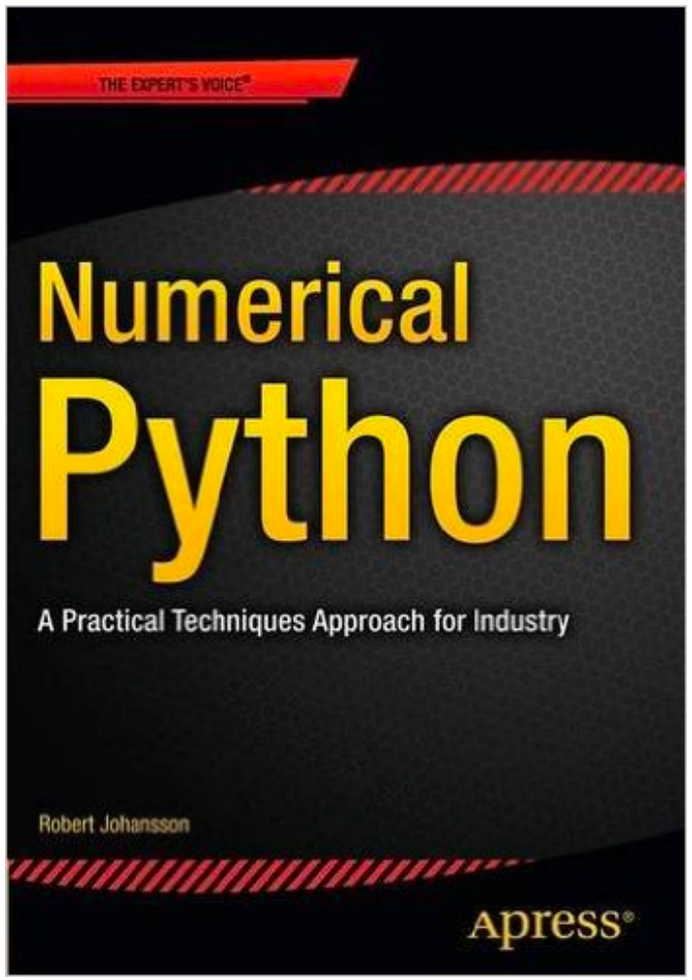
Python 科学计算第二讲

杨宏亮

2019/05/10

Matplotlib

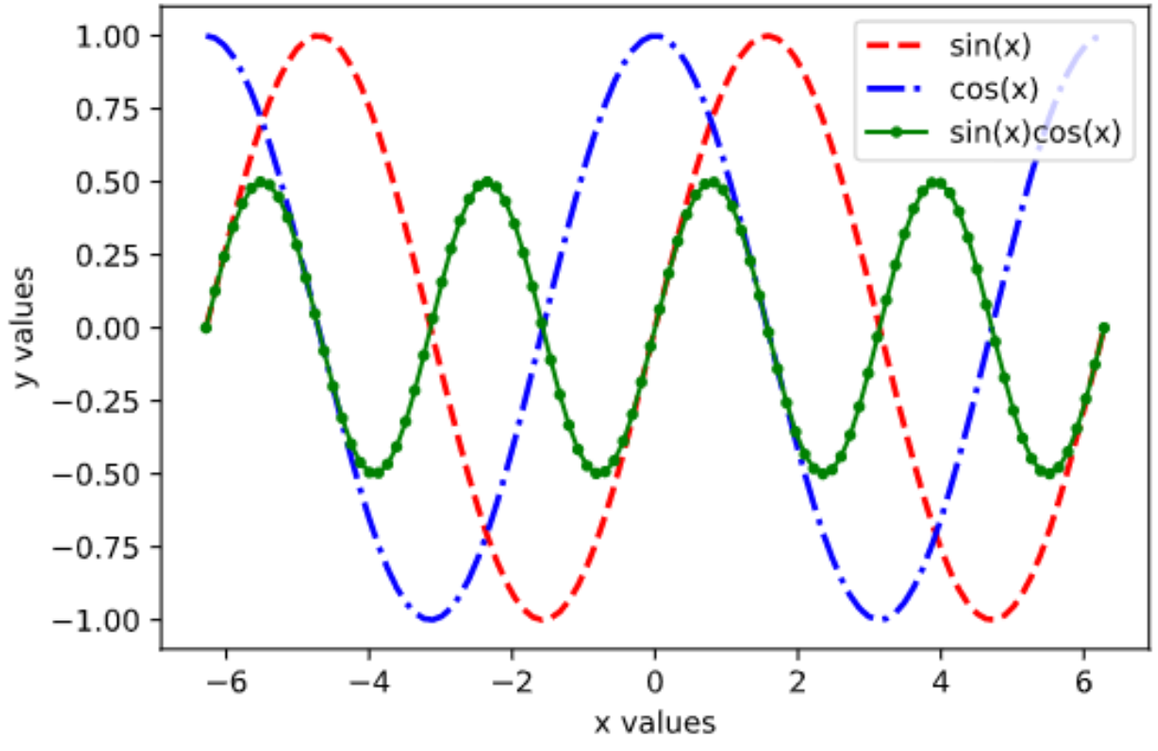
参考书



http://jrjohansson.github.io/numericalpython.html

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
x = np.linspace(-2*np.pi, 2*np.pi, 100)
fig, ax = plt.subplots()
ax.plot(x, np.sin(x), "r--", lw=2, label="sin(x)")
ax.plot(x, np.cos(x), color="b", linestyle="-.",
        linewidth=2, label="cos(x)")
ax.plot(x, np.cos(x)*np.sin(x), "go-", ms=3, label="sin(x)cos(x)")

ax.set_xlabel("x values")
ax.set_ylabel("y values")
ax.legend()
```

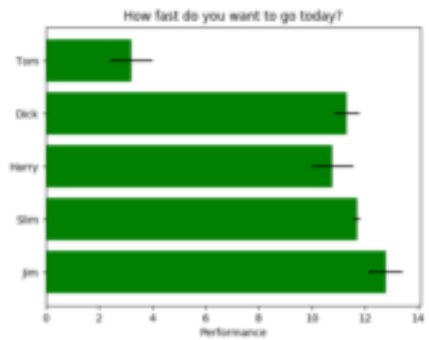


Matplotlib gallery

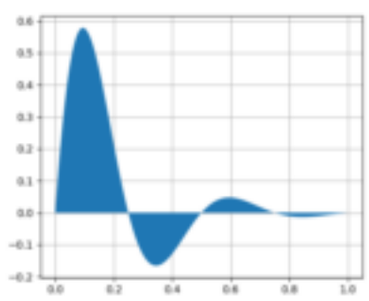
https://matplotlib.org/gallery.html

- Gallery
  - Lines, bars, and markers
  - Shapes and collections
  - Statistical plots
  - Images, contours, and fields
  - Pie and polar charts
  - Color
  - Text, labels, and annotations
  - Ticks and spines
  - Axis scales
  - Subplots, axes, and figures
  - Style sheets
  - Specialty plots
  - Showcase
  - API
  - pylab examples
  - mplot3d toolkit
  - axes_grid toolkit
  - widgets
  - Miscellaneous examples

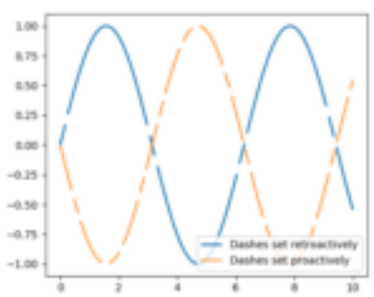
Lines, bars, and markers



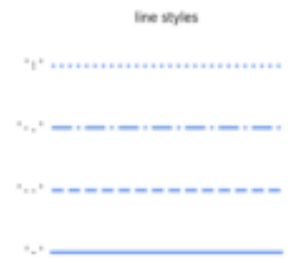
barh_demo



fill_demo



line_demo_dash_control



line_styles_reference

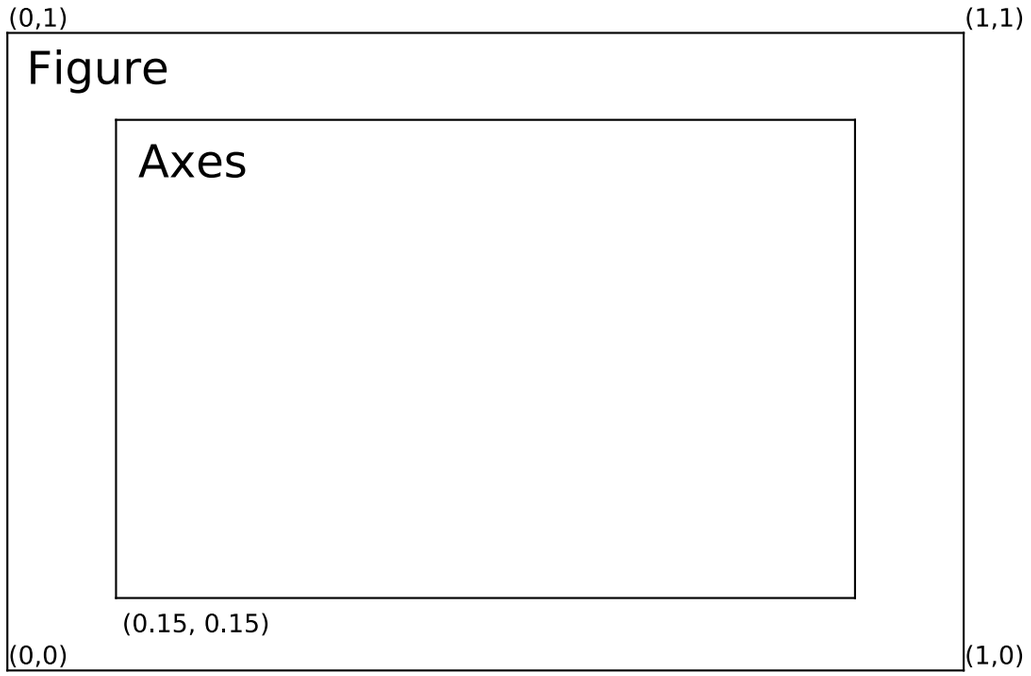


Figure 可以理解为画布, 所有的东西都在画布上;
Axis 为实际的坐标系统

Backend	Description
GTKAgg	Agg rendering to a GTK 2.x canvas (requires PyGTK and pycairo or cairocffi; Python2 only)
GTK3Agg	Agg rendering to a GTK 3.x canvas (requires PyGObject and pycairo or cairocffi)
GTK	GDK rendering to a GTK 2.x canvas (not recommended and d eprecated in 2.0) (requires PyGTK and pycairo or cairocffi; Python2 only)
GTKCairo	Cairo rendering to a GTK 2.x canvas (requires PyGTK and pycairo or cairocffi; Python2 only)
GTK3Cairo	Cairo rendering to a GTK 3.x canvas (requires PyGObject and pycairo or cairocffi)
WXAgg	Agg rendering to to a wxWidgets canvas (requires wxPython)
WX	Native wxWidgets drawing to a wxWidgets Canvas (not recommended and deprecated in 2.0) (requires wxPython)
TkAgg	Agg rendering to a Tk canvas (requires TkInter)
Qt4Agg	Agg rendering to a Qt4 canvas (requires PyQt4 or `pyside`)
Qt5Agg	Agg rendering in a Qt5 canvas (requires PyQt5)
macosx	Cocoa rendering in OSX windows (presently lacks blocking show() behavior when matplotlib is in non-interactive mode)

```
import matplotlib as mpl
mpl.use("qt5agg")
import matplotlib.pyplot as plt
```

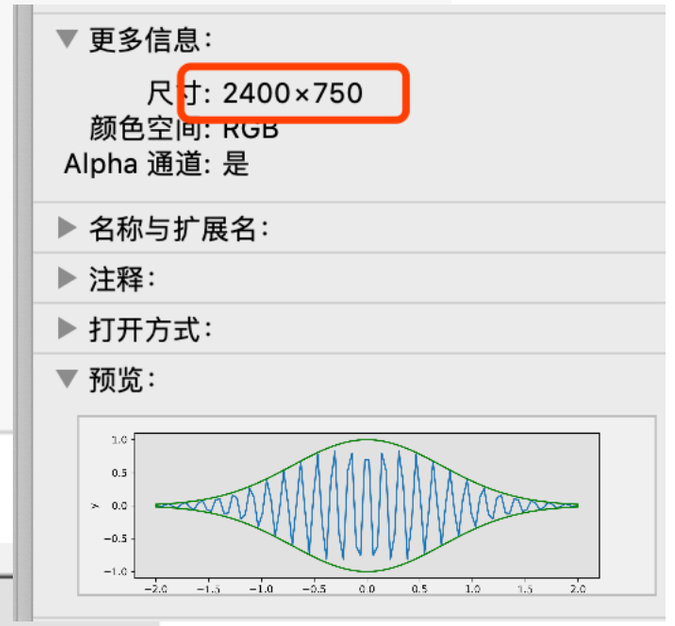
```
%matplotlib inline
# %config InlineBackend.figure_format="svg"
%config InlineBackend.figure_format='retina'
```

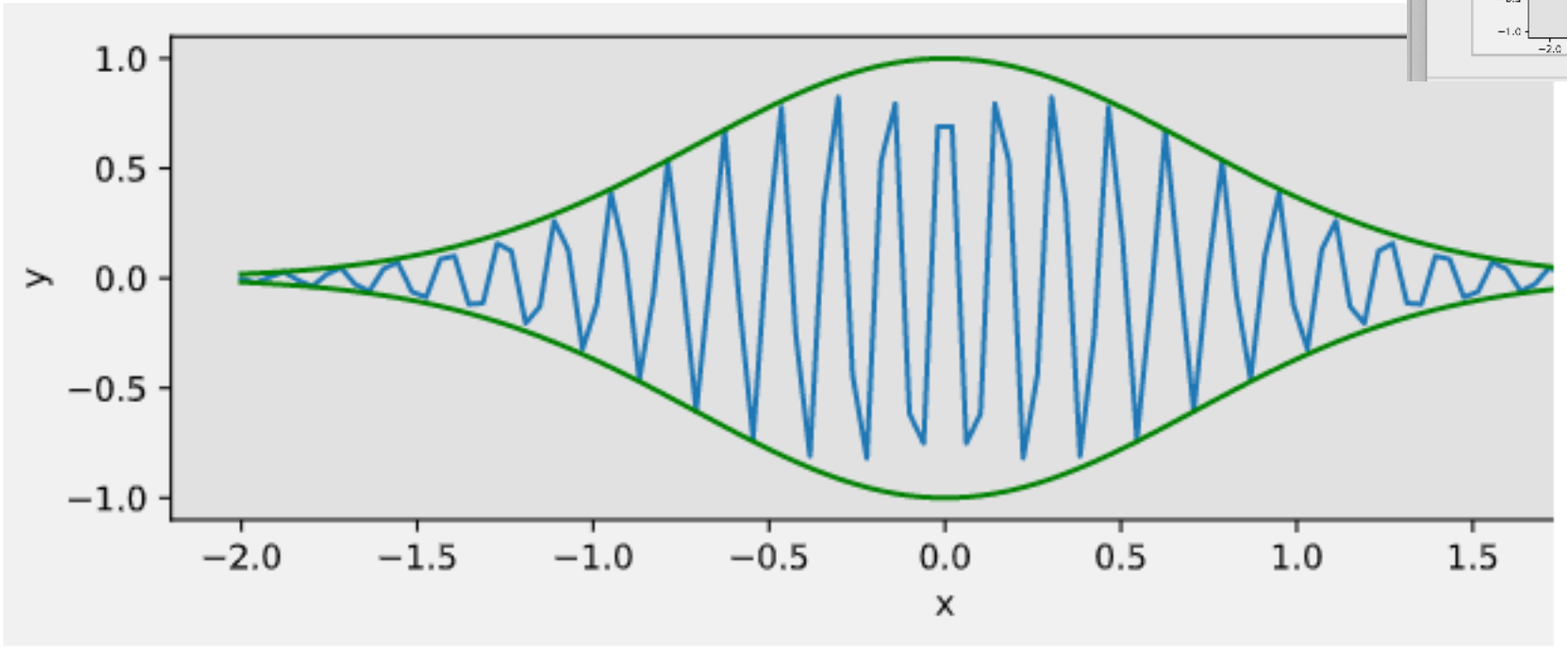
```python
fig = plt.figure(figsize=(8, 2.5), facecolor="#f1f1f1")
left, bottom, width, height = 0.1, 0.1, 0.8, 0.8
ax = fig.add_axes((left, bottom, width, height), facecolor="#e1e1e2")

x = np.linspace(-2, 2, 100)
y1 = np.cos(40*x)
y2= np.exp(-x**2)

ax.plot(x, y1*y2)
ax.plot(x, y2, "g")
ax.plot(x, -y2, "g")

ax.set_xlabel("x")
ax.set_ylabel("y")

fig.savefig("graph.png", dpi=300, facecolor="#f1f1f1")
```

dpi: dots per inch

像素=dpi * fig size

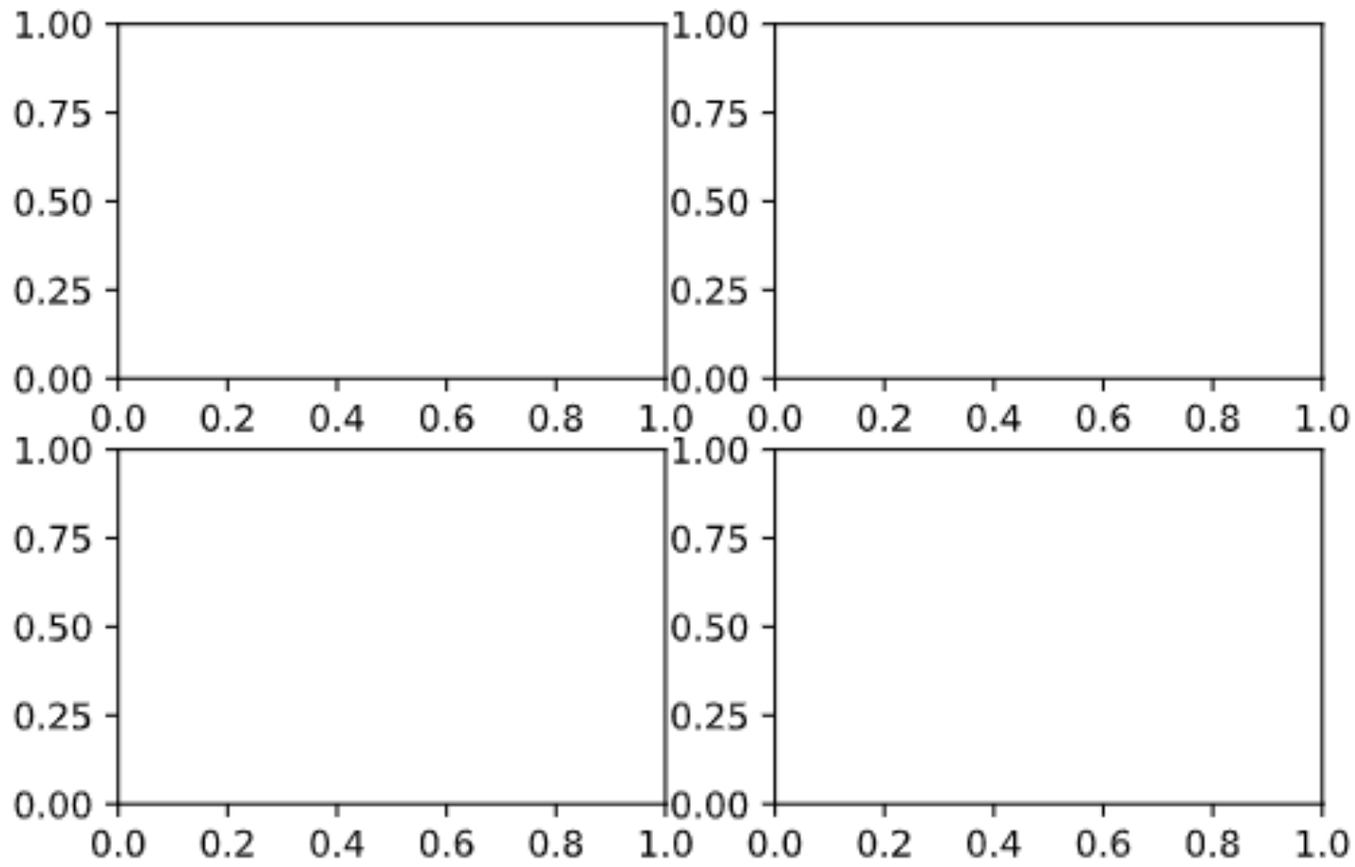
常用图片格式: PNG, PDF, EPS, SVG





Axes

```
fig, axes = plt.subplots(nrows=2, ncols=2)
```




```
axes
```

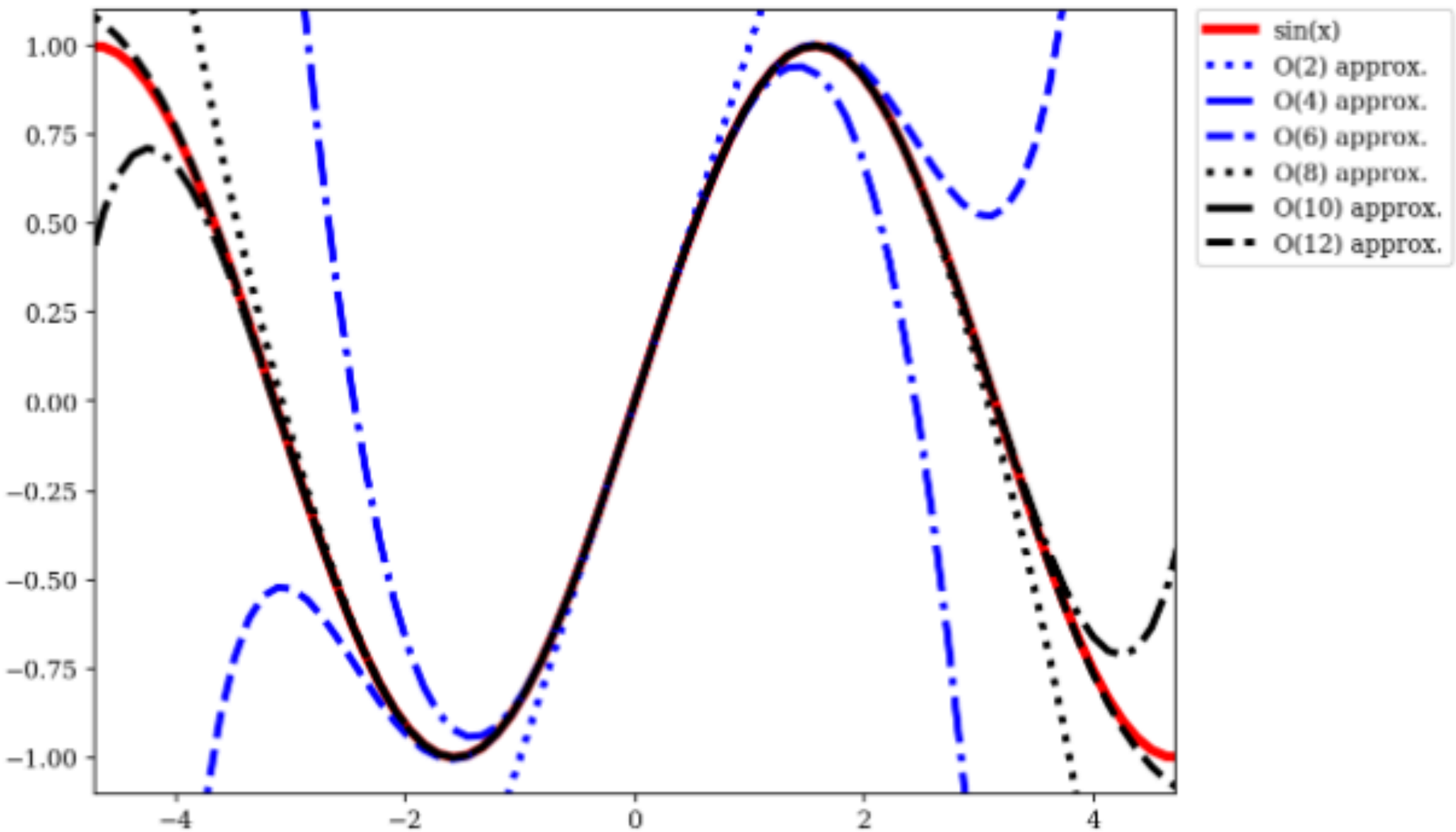
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11659e128>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1158a1668>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x11574c160>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11587f3c8>]],
      dtype=object)
```

Line properties

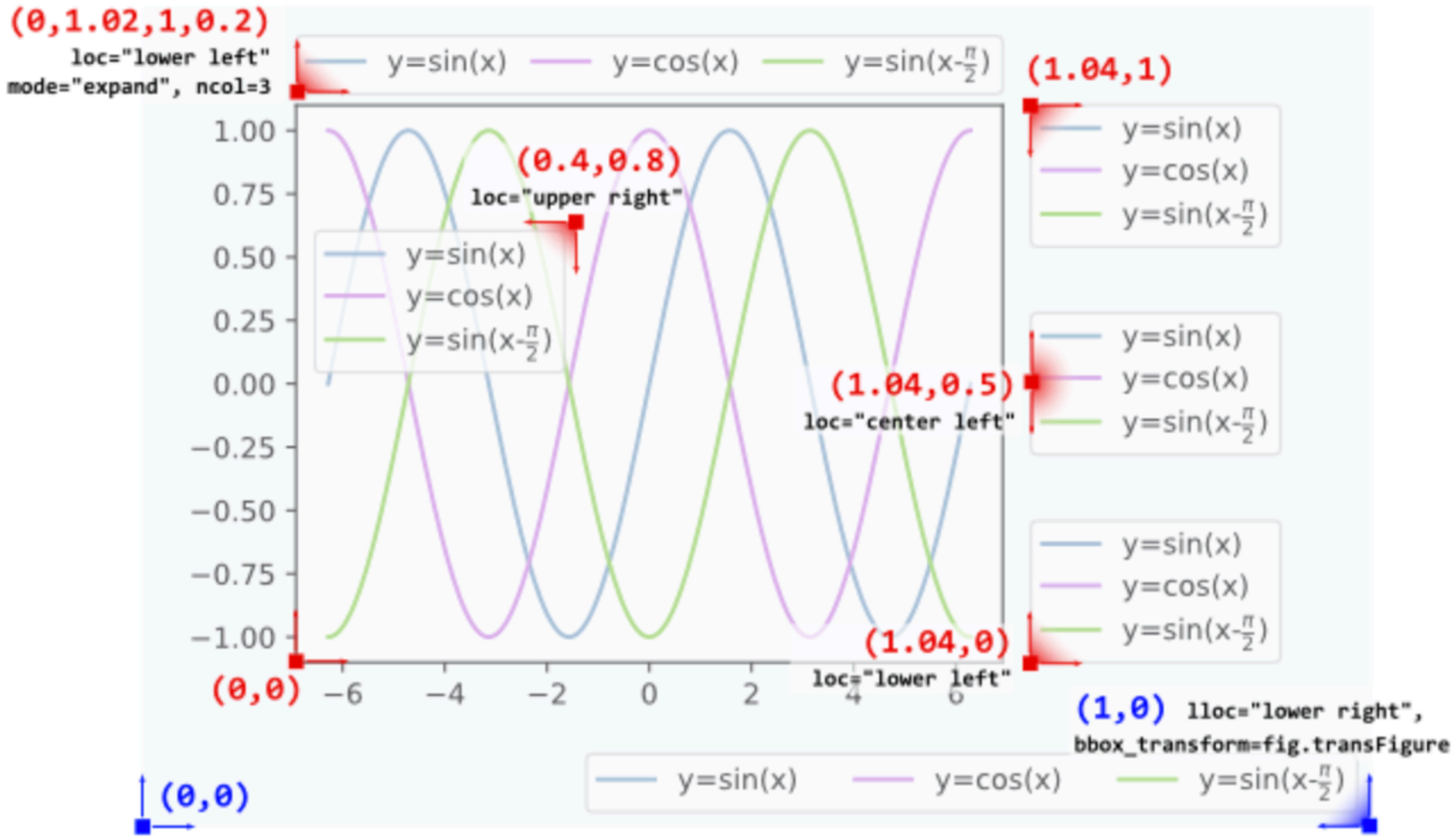
Argument	Example values	Description
`color`	A color specification can be a string with a color name, such as “red,” “blue,” etc., or a RGB color code on the form “#aabbcc.”	A color specification.
`alpha`	Float number between 0.0 (completely transparent) to 1.0 (completely opaque).	The amount of transparency.
`linewidth, lw`	Float number.	The width of a line.
`linestyle, ls`	‘-‘ – solid ‘--’ – dashed ‘:’ – dotted ‘.-’ – dash-dotted	The style of the line, i.e., whether the line is to be draw as a solid line, or if it should be, for example, dotted or dashed.

Argument	Example values	Description
`marker`	+, o, * = cross, circle, star s = square . = small dot 1, 2, 3, 4, `...` = triangle-shaped symbols with different angles.	Each data point, whether or not it is connected with adjacent data points, can be represented with a marker symbol as specified with this argument.
`markersize`	Float number.	The marker size.
`makerfacecolor`	Color specification (see above).	The fill color for the marker.
`markeredgewidth`	Float number.	The line width of the marker edge.
`markeredgecolor`	Color specification (see above).	The marker edge color.

```
fig, ax = plt.subplots(figsize=(10, 6))

ax.plot(x, np.sin(x), linewidth=4, color="red", label='sin(x)')

colors = ["blue", "black"]
linestyles = [':', '-.', '--']
for idx, n in enumerate(range(1, 12, 2)):
    ax.plot(x, sin_expansion(x, n), color=colors[idx // 3],
            linestyle=linestyles[idx % 3], linewidth=3,
            label="O(%d) approx." % (n+1))

ax.set_ylim(-1.1, 1.1)
ax.set_xlim(-1.5*np.pi, 1.5*np.pi)
ax.legend(bbox_to_anchor=(1.02, 1), loc=2, borderaxespad=0.0)
fig.subplots_adjust(right=.75);
```



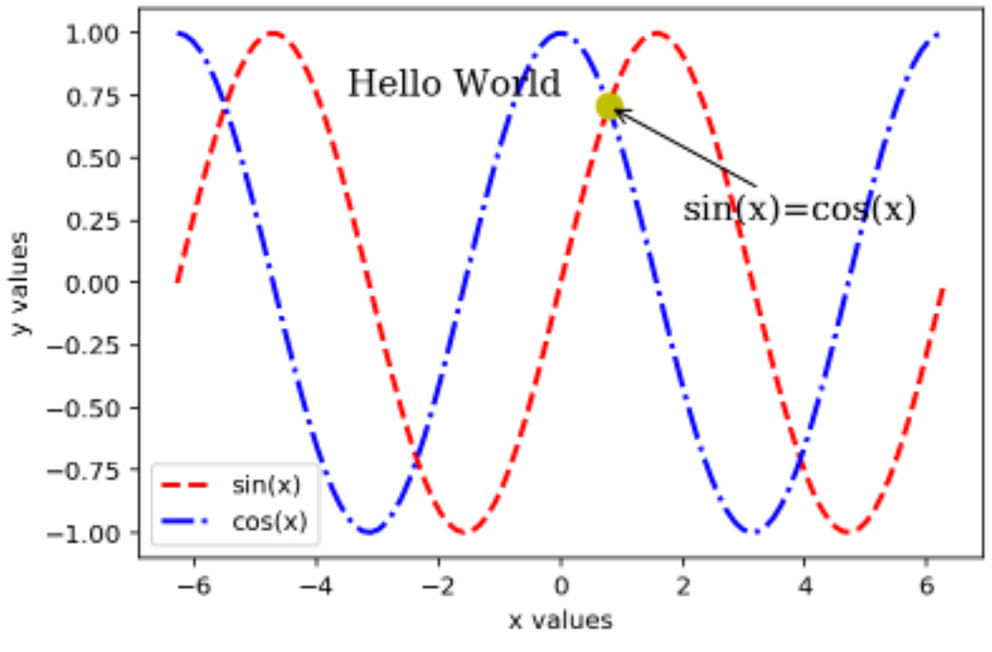
Legend




```
l1 = ax.legend(bbox_to_anchor=(1.04,1), borderaxespad=0)
l2 = ax.legend(bbox_to_anchor=(1.04,0), loc="lower left", borderaxespad=0)
l3 = ax.legend(bbox_to_anchor=(1.04,0.5), loc="center left", borderaxespad
l4 = ax.legend(bbox_to_anchor=(0,1.02,1,0.2), loc="lower left",
                mode="expand", borderaxespad=0, ncol=3)
l5 = ax.legend(bbox_to_anchor=(1,0), loc="lower right",
                bbox_transform=fig.transFigure, ncol=3)
l6 = ax.legend(bbox_to_anchor=(0.4,0.8), loc="upper right")
```

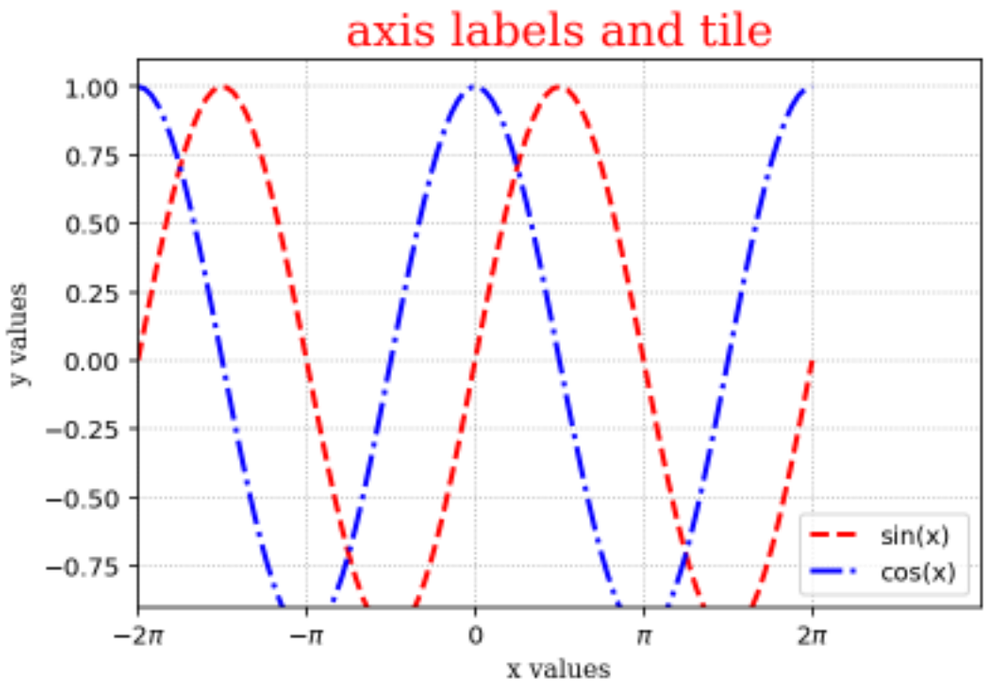
Argument	Description
`fontsize`	The size of the font, in points.
`family`	The font type.
`backgroundcolor`	Color specification for the background of the text label.
`color`	Color specification for the font color.
`alpha`	Transparency of the font color.
`rotation`	Rotation angle of the text label.

```
ax.text(-3.5, 0.75, "Hello World", fontsize=14, family="serif")
ax.annotate("sin(x)=cos(x)",
            fontsize=14, family="serif",
            xy=(0.78539816, 0.70710678),
            xytext=(2.0, 0.25),
            arrowprops=dict(arrowstyle="->"))
```

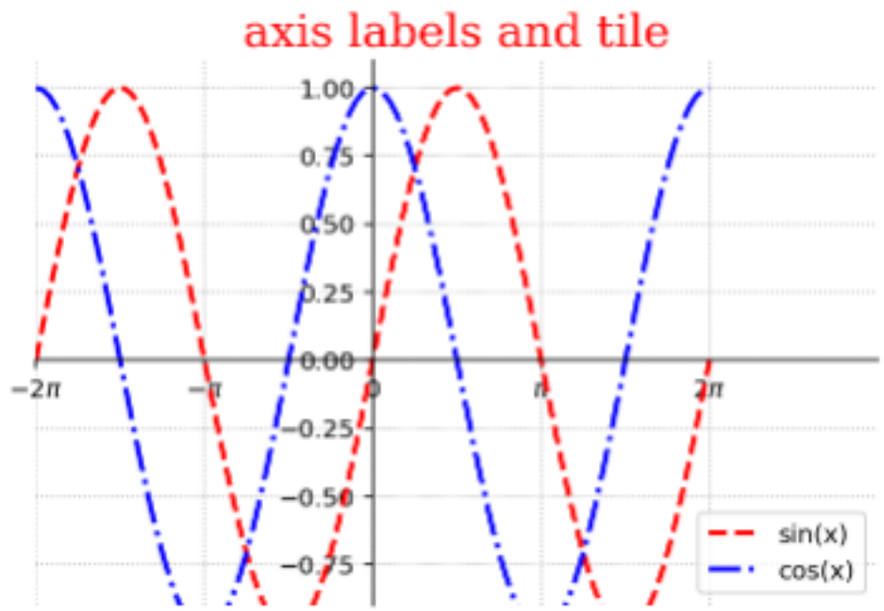


Axis

```
ax.set_xlabel("x values",  fontname="serif")
ax.set_ylabel("y values",  fontname="serif")
ax.set_title("axis labels and tile", fontsize=18,
             fontname="serif", color="red")

ax.set_xlim([-2*np.pi, 2*np.pi*1.5])
ax.set_ylim([-0.9, 1.1])

# set major tick
ax.set_xticks([-2 * np.pi, -np.pi, 0, np.pi, 2 * np.pi])
ax.set_xticklabels(['$-2\pi$', '$-\pi$', 0, r'$\pi$', r'$2\pi$'])

# set minor tick
mpl.ticker.FixedLocator([-3 * np.pi / 2, -np.pi/2, 0,
                         np.pi/2, 3 * np.pi/2])

ax.grid(ls=":")
```

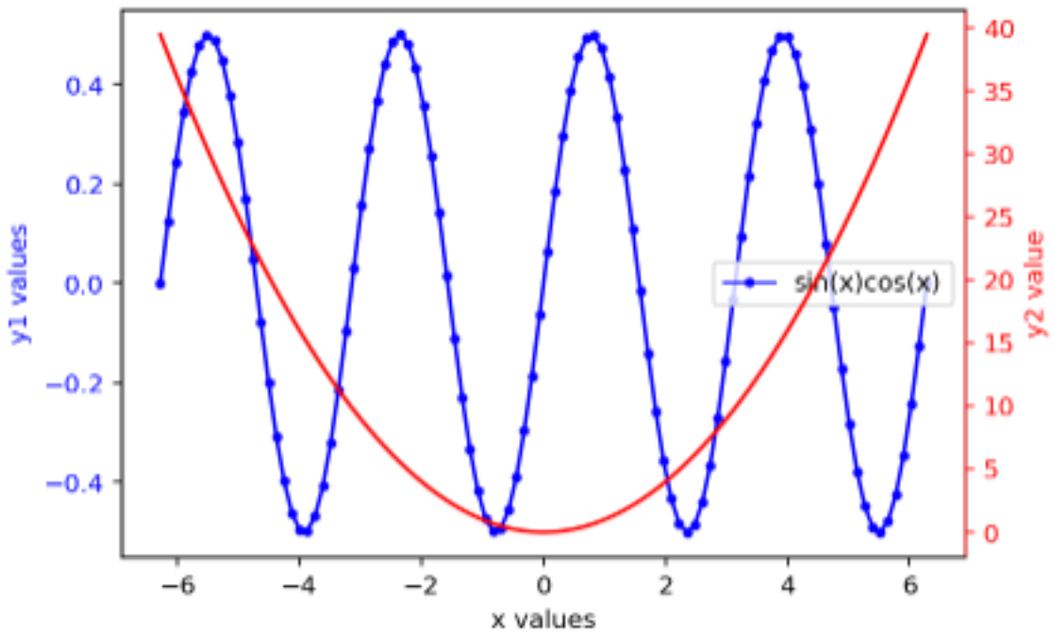


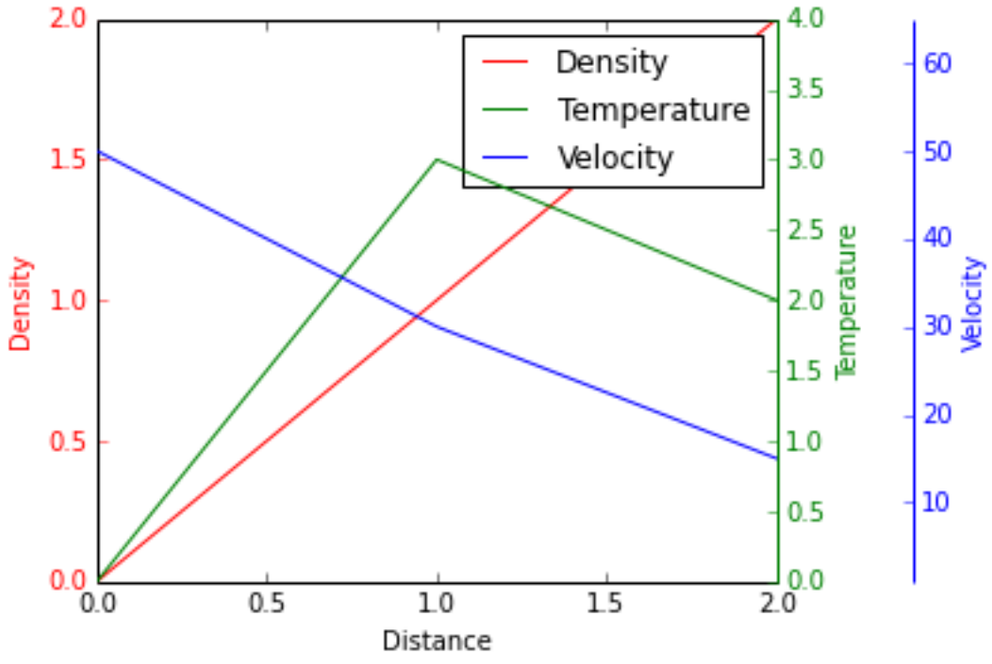

```
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')

ax.spines["bottom"].set_position(("data", 0))
ax.spines["left"].set_position(("data", 0))
```



Twin axes

```
ax2 = ax.twinx()
p2, = ax2.plot(x, x**2, "r")
for label in ax2.get_yticklabels():
    label.set_color("red")
ax2.set_ylabel("y2 value", color=p2.get_color())

ax2.spines["right"].set_color(p2.get_color())

ax2.tick_params(axis='y', colors=p2.get_color())
ax.legend()
```

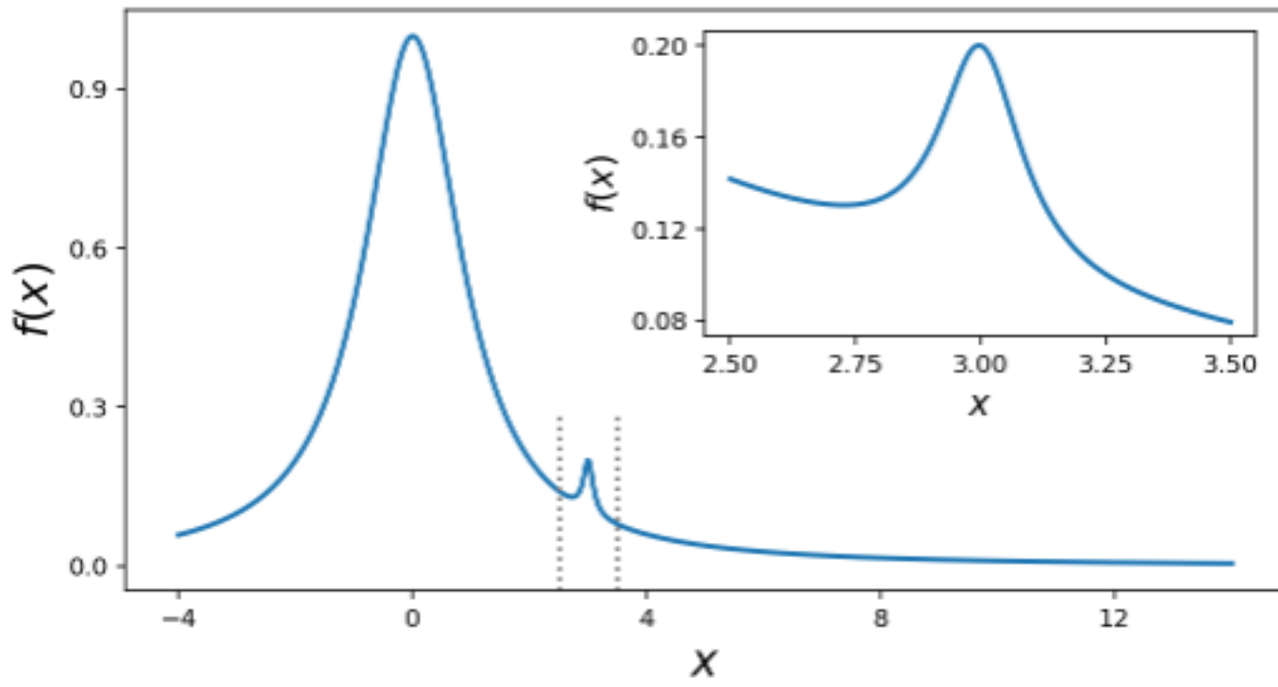




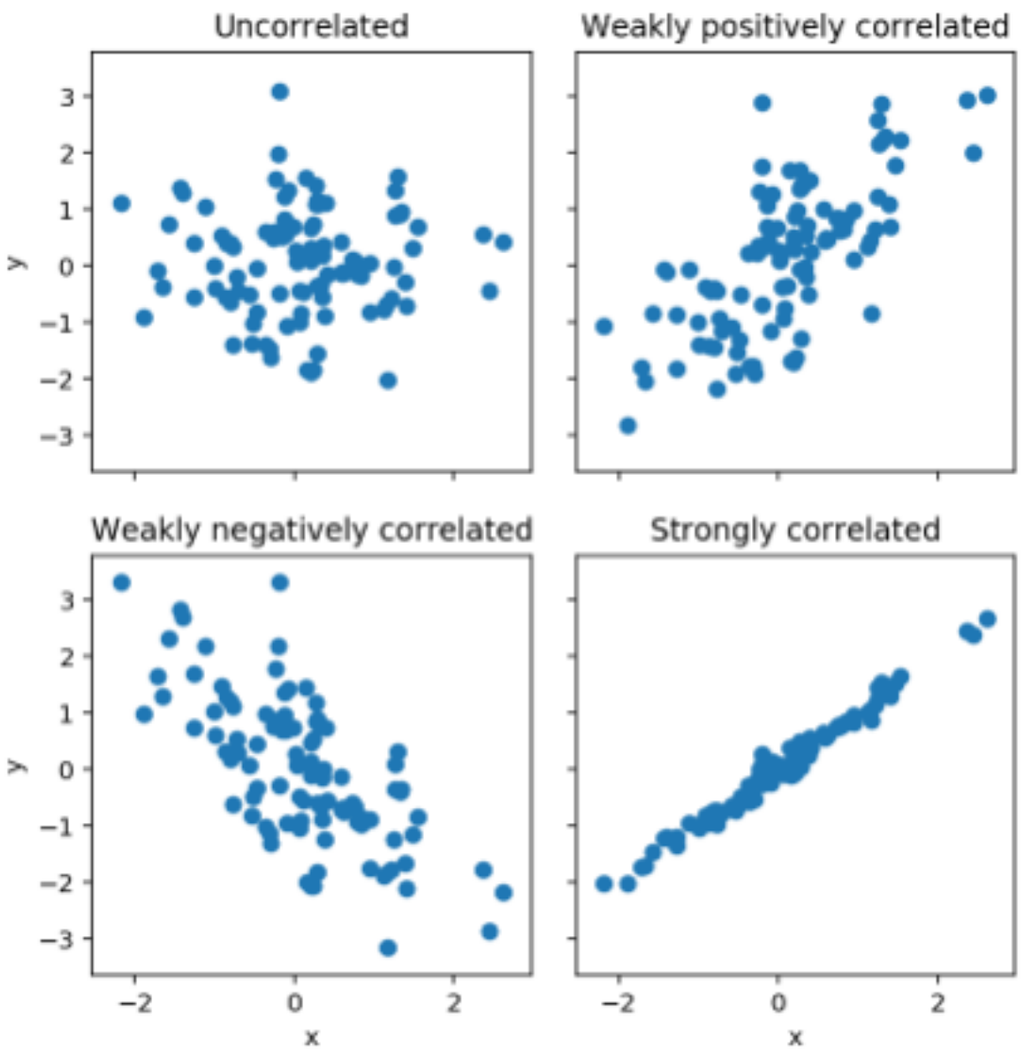
https://stackoverflow.com/questions/20356982/matplotliåb-colour-multiple-twinx-axes

Insert plot

```
fig = plt.figure(figsize=(8, 4))

def f(x):
    return 1/(1 + x**2) + 0.1/(1 + ((3 - x)/0.1)**2)

def plot_and_format_axes(ax, x, f, fontsize):
    ax.plot(x, f(x), linewidth=2)
    ax.xaxis.set_major_locator(mpl.ticker.MaxNLocator(5))
    ax.yaxis.set_major_locator(mpl.ticker.MaxNLocator(4))
    ax.set_xlabel(r"$x$", fontsize=fontsize)
    ax.set_ylabel(r"$f(x)$", fontsize=fontsize)

# main graph
ax = fig.add_axes([0.1, 0.15, 0.8, 0.8])
x = np.linspace(-4, 14, 1000)
plot_and_format_axes(ax, x, f, 18)

# inset
x0, x1 = 2.5, 3.5
ax.axvline(x0, ymax=0.3, color="grey", linestyle=":")
ax.axvline(x1, ymax=0.3, color="grey", linestyle=":")

ax = fig.add_axes([0.5, 0.5, 0.38, 0.42])
x = np.linspace(x0, x1, 1000)
plot_and_format_axes(ax, x, f, 14)
```

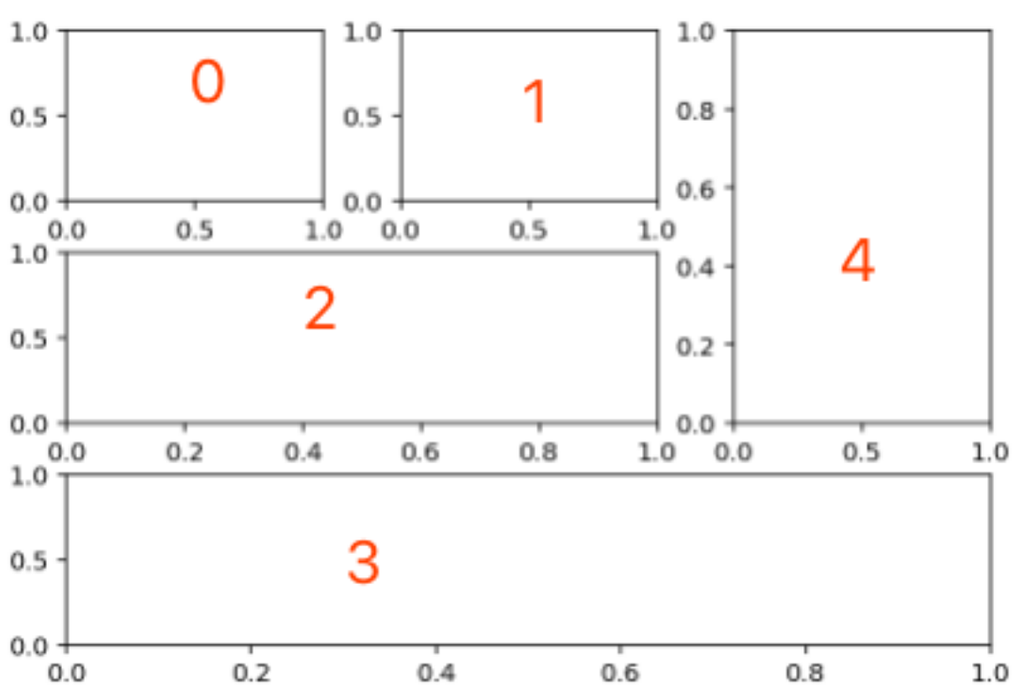


```python
x1 = np.random.randn(100)
x2 = np.random.randn(100)

fig, axes = plt.subplots(2, 2, figsize=(6, 6), sharex=True,
                         sharey=True)

axes[0, 0].set_title("Uncorrelated")
axes[0, 0].scatter(x1, x2)

axes[0, 1].set_title("Weakly positively correlated")
axes[0, 1].scatter(x1, x1 + x2)

axes[1, 0].set_title("Weakly negatively correlated")
axes[1, 0].scatter(x1, -x1 + x2)

axes[1, 1].set_title("Strongly correlated")
axes[1, 1].scatter(x1, x1 + 0.15 * x2)

axes[1, 1].set_xlabel("x")
axes[1, 0].set_xlabel("x")
axes[0, 0].set_ylabel("y")
axes[1, 0].set_ylabel("y")

plt.subplots_adjust(left=0.1, right=0.95, bottom=0.1, top=0.95,
                    wspace=0.1, hspace=0.2)
```



Subplot2grid

```
ax0 = plt.subplot2grid((3, 3), (0, 0))
ax1 = plt.subplot2grid((3, 3), (0, 1))

ax2 = plt.subplot2grid((3, 3), (1, 0), colspan=2)
ax3 = plt.subplot2grid((3, 3), (2, 0), colspan=3)
ax4 = plt.subplot2grid((3, 3), (0, 2), rowspan=2)

plt.subplots_adjust(left=0.1, right=0.95, bottom=0.1, top=0.95,
                    wspace=0.3, hspace=0.3)
```



配置

https://matplotlib.org/users/customizing.html

- 直接写在代码中
- 写在配置文件中
- 多个配置文件

matplotlibrc

The `matplotlibrc` file

matplotlib uses `matplotlibrc` configuration files to customize all kinds of properties, which we call `rc settings` or `rc parameters`. You can control the defaults of almost every property in matplotlib: figure size and dpi, line width, color and style, axes, axis and grid properties, text and font properties and so on. matplotlib looks for `matplotlibrc` in four locations, in the following order:

1. `matplotlibrc` in the current working directory, usually used for specific customizations that you do not want to apply elsewhere.
2. `$MATPLOTLIBRC/matplotlibrc`.
3. It next looks in a user-specific place, depending on your platform:
   - On Linux and FreeBSD, it looks in `.config/matplotlib/matplotlibrc` (or `$XDG_CONFIG_HOME/matplotlib/matplotlibrc`) if you've customized your environment.
   - On other platforms, it looks in `.matplotlib/matplotlibrc`.

   See matplotlib configuration and cache directory locations.
4. *`INSTALL`*`/matplotlib/mpl-data/matplotlibrc`, where *`INSTALL`* is something like `/usr/lib/python3.5/site-packages` on Linux, and maybe `C:\Python35\Lib\site-packages` on Windows. Every time you install matplotlib, this file will be overwritten, so if you want your customizations to be saved, please move this file to your user-specific matplotlib directory.

To display where the currently active `matplotlibrc` file was loaded from, one can do the following:

```
>>> import matplotlib
>>> matplotlib.matplotlib_fname()
'/home/foo/.config/matplotlib/matplotlibrc'
```

https://matplotlib.org/users/customizing.html

中文

Sympy

https://github.com/jrjohansson/scientific-python-lectures