matplotlib 绘图先导课

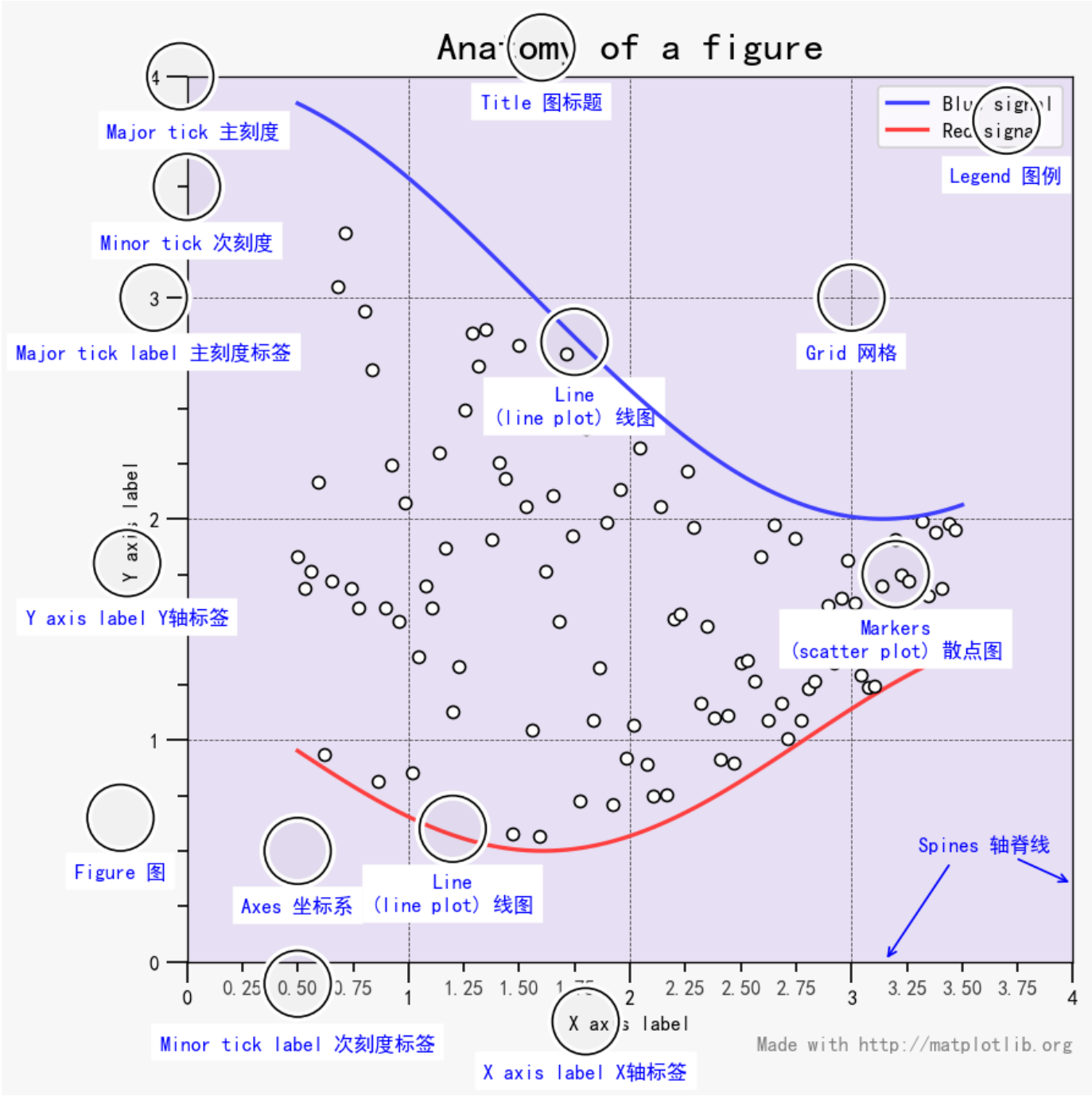
In [1]:

```
import numpy as np
import pandas as pd
```

一、matplotlib 中一幅图的构成

层级结构：figure 图 ～ axes 坐标系（绘图区域） ～ axis 坐标轴（y轴 Yaxis 和 x轴 Xaxis） ～ ticks 刻度（主刻度 MajorTicks 和次刻度 MinorTicks）

其他元素：标题（suptitle 图的中心标题 和 title 坐标系的子标题）、轴标签（xlabel 和 ylabel）、刻度标签（xticklabels 和 yticklabels）、轴脊线（spines）、图例（legend）、网格（grid）、各类图形（lines 线图、bar 柱状图、scatter散点图、hist 直方图、pie 饼图 ......）



二、如何用 **matplotlib** 画图 ?

matplotlib 共有如下 2 种绘图方式：

2.1 方式 1：基于 **pyplot** 接口的绘图方式

pyplot 接口是使 matplotlib 像 MATLAB 一样工作的**命令样式函数的集合**，每个 pyplot 函数都会对图形进行一些更改：例如，创建图形，在图形中创建绘图区域，在绘图区域中绘制曲线，用标签装饰绘图等。

- matplotlib.pyplot 官方文档：https://matplotlib.org/tutorials/introductory/pyplot.html#sphx-glr-tutorials-introductory-pyplot-py (https://matplotlib.org/tutorials/introductory/pyplot.html#sphx-glr-tutorials-introductory-pyplot-py)

In [2]:

```
import matplotlib.pyplot as plt   # 首先导入 pyplot 模块
```

In [3]:

```
np.linspace(0, 2, 100)
```

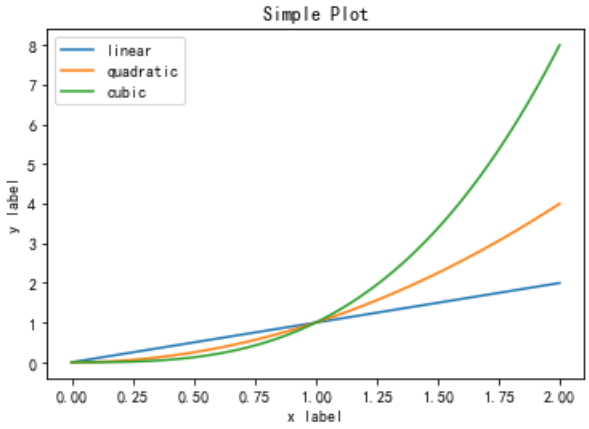
Out[3]:

```
array([0.        , 0.02020202, 0.04040404, 0.06060606, 0.08080808,
       0.1010101 , 0.12121212, 0.14141414, 0.16161616, 0.18181818,
       0.2020202 , 0.22222222, 0.24242424, 0.26262626, 0.28282828,
       0.3030303 , 0.32323232, 0.34343434, 0.36363636, 0.38383838,
       0.4040404 , 0.42424242, 0.44444444, 0.46464646, 0.48484848,
       0.50505051, 0.52525253, 0.54545455, 0.56565657, 0.58585859,
       0.60606061, 0.62626263, 0.64646465, 0.66666667, 0.68686869,
       0.70707071, 0.72727273, 0.74747475, 0.76767677, 0.78787879,
       0.80808081, 0.82828283, 0.84848485, 0.86868687, 0.88888889,
       0.90909091, 0.92929293, 0.94949495, 0.96969697, 0.98989899,
       1.01010101, 1.03030303, 1.05050505, 1.07070707, 1.09090909,
       1.11111111, 1.13131313, 1.15151515, 1.17171717, 1.19191919,
       1.21212121, 1.23232323, 1.25252525, 1.27272727, 1.29292929,
       1.31313131, 1.33333333, 1.35353535, 1.37373737, 1.39393939,
       1.41414141, 1.43434343, 1.45454545, 1.47474747, 1.49494949,
       1.51515152, 1.53535354, 1.55555556, 1.57575758, 1.5959596 ,
       1.61616162, 1.63636364, 1.65656566, 1.67676768, 1.6969697 ,
       1.71717172, 1.73737374, 1.75757576, 1.77777778, 1.7979798 ,
       1.81818182, 1.83838384, 1.85858586, 1.87878788, 1.8989899 ,
       1.91919192, 1.93939394, 1.95959596, 1.97979798, 2.        ])
```

In [5]:

```
# 基于 pyplot 接口 绘图的 简单例子

x = np.linspace(0, 2, 100)
plt.plot(x, x, label='linear')  # 调用 plot() 时，底层会默默的创建好图、坐标系、坐标轴，直接画就行
plt.plot(x, x**2, label='quadratic')
plt.plot(x, x**3, label='cubic')
plt.xlabel('x label')  # 添加坐标轴的标签
plt.ylabel('y label')
plt.title("Simple Plot")  # 添加图标题
plt.legend() # 添加图例
```

Out[5]:

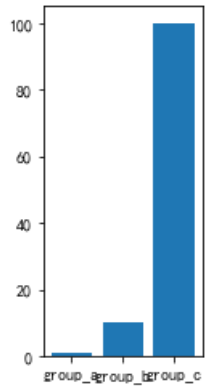


In [11]:

```
names = ['group_a', 'group_b', 'group_c']
values = [1, 10, 100]
plt.subplot(131)  # 往图中添加子图，返回的是子图的坐标系 axes
plt.bar(names, values)
```

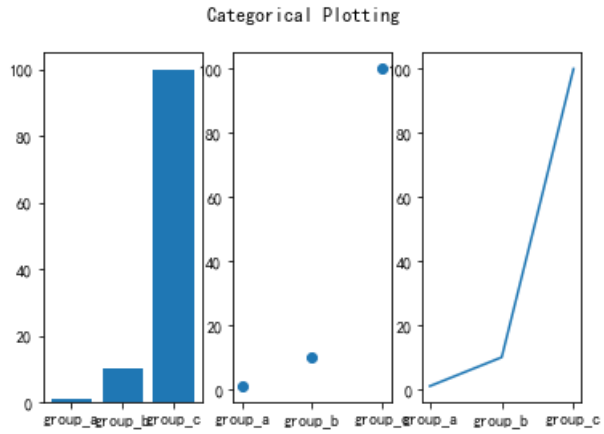
Out[11]:

```
<BarContainer object of 3 artists>
```



In [15]:

```
names = ['group_a', 'group_b', 'group_c']
values = [1, 10, 100]
plt.subplot(131)  # 往图中添加子图，返回的是子图的坐标系 axes
plt.bar(names, values)
plt.subplot(132)
plt.scatter(names, values)
plt.subplot(133)
plt.plot(names, values)
plt.suptitle('Categorical Plotting')
plt.show()
```



- pyplot-style的绘图方式：易学易上手（直接调用函数，无需了解底层结构，黑箱）、更适合在 jupyter notebook 中进行交互式画图（边画边出结果）；

2.2 方式 2：面向对象的绘图方式（object-oriented (OO) style）

面向对象的绘图方式**首先要创建好图 `figure` 和坐标系 `axes`**，然后以坐标系为基础，**直接调用坐标系上的各种绘图方法绘制图形**。

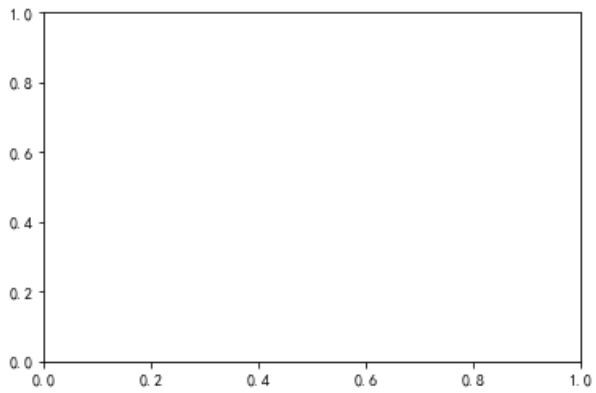
虽然 OO-style 的绘图方式有别与 pyplot-style 绘图方式，但图 figure 和坐标系 axes 还是需要通过调用 pyplot 接口来创建。

- matplotlib.figure.Figure 官方文档： https://matplotlib.org/api/_as_gen/matplotlib.figure.Figure.html#matplotlib.figure.Figure (https://matplotlib.org/api/_as_gen/matplotlib.figure.Figure.html#matplotlib.figure.Figure)
- matplotlib.axes.Axes 官方文档： https://matplotlib.org/api/axes_api.html#matplotlib.axes.Axes (https://matplotlib.org/api/axes_api.html#matplotlib.axes.Axes)

In [19]:

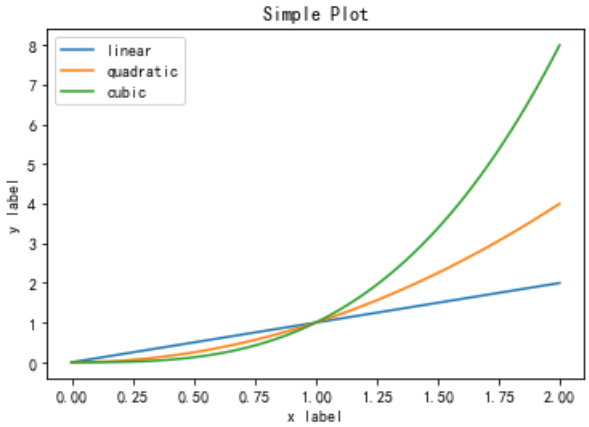
```
import matplotlib.pyplot as plt    # 首先导入 pyplot 模块
fig, ax = plt.subplots()  # 创建图和坐标系，然后在坐标系中添加图形和相关元素
print(fig,"123",ax)
```

```
Figure(432x288) 123 AxesSubplot(0.125,0.125;0.775x0.755)
```



In [20]:

```
fig, ax = plt.subplots()  # 同时创建了 图 figure 和 坐标系 axes

x = np.linspace(0, 2, 100)
ax.plot(x, x, label='linear')
ax.plot(x, x**2, label='quadratic')
ax.plot(x, x**3, label='cubic')
ax.set_xlabel('x label')
ax.set_ylabel('y label')
ax.set_title("Simple Plot")
ax.legend()

plt.show()

# 如何放大图形 figsize=(10,8)
```



- OO-style的绘图方式：需要了解绘图的底层结构和绘图逻辑、比 pyplot-style 更为灵活和复杂、更适用于非交互式绘图（如在较大项目中作为一部分重复使用的函数或在脚本中编写绘图函数）。

三、一些常用图元素的实现

3.1 如何创建图和坐标系 ？

主要包括：

- 3.1.1 创建 figure；
- 3.1.2 创建坐标系；
- 3.1.3 创建子图；
- 3.1.4 创建分布不规则的子图；
- 3.1.5 总结 。

3.1.1 创建 figure

方式1（pyplot-style）：`plt.figure()`

```
matplotlib.pyplot.figure(num=None, figsize=None, dpi=None, facecolor=None, edgecolor
=None, frameon=True, FigureClass=<class 'matplotlib.figure.Figure'>, clear=False, **
kwargs)
返回的是：Figure 对象
```

In [19]:

```
# 方式1: plt.figure()
fig = plt.figure(num=2,figsize=(10,8), facecolor='#f5f5f5')  # 创建了一个图，没有坐标系的图
```

```
<Figure size 720x576 with 0 Axes>
```

In [4]:

```
print(fig)
```

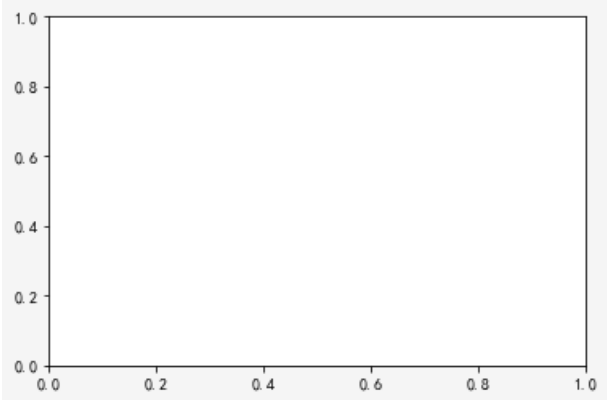
```
Figure(720x576)
```

方式2：`plt.subplots()`，同时创建 figure 和 多幅子图 subplot

```
matplotlib.pyplot.subplots(nrows=1, ncols=1, sharex=False, sharey=False, squeeze=Tru
e, subplot_kw=None, gridspec_kw=None, **fig_kw)
返回的是：Figure 对象和 axes.Axes 对象
```

In [6]:

```
# 方式2：plt.subplots()
fig,ax = plt.subplots(facecolor='#f5f5f5')  # 同时生成了坐标系，所以能在眼前展示出来
```



In [26]:

```
print(fig)
```

```
Figure(432x288)
```

3.1.2 创建坐标系

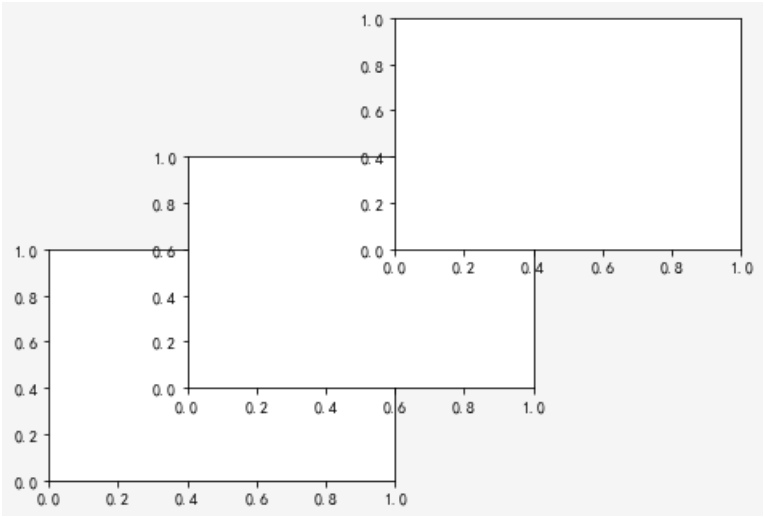
方式1（pyplot-style）：`plt.axes()` 在创建完 figure 后接着创建 坐标系

```
plt.axes()
plt.axes(rect=[left, bottom, width, height], projection=None, polar=False, **kwargs)

plt.axes(ax)
其中：left 代表坐标系左边到 Figure 左边的水平距离；bottom 代表坐标系底边到 Figure 底边的垂直
距离；width 代表坐标系的宽度；height 代表坐标系的高度
```

In [11]:

```
# 方式1
plt.figure( facecolor='#f5f5f5')
plt.axes([0,0,0.5,0.5])
plt.axes([0.2,0.2,0.5,0.5])
plt.axes([0.5,0.5,0.5,0.5])
```

Out[11]:



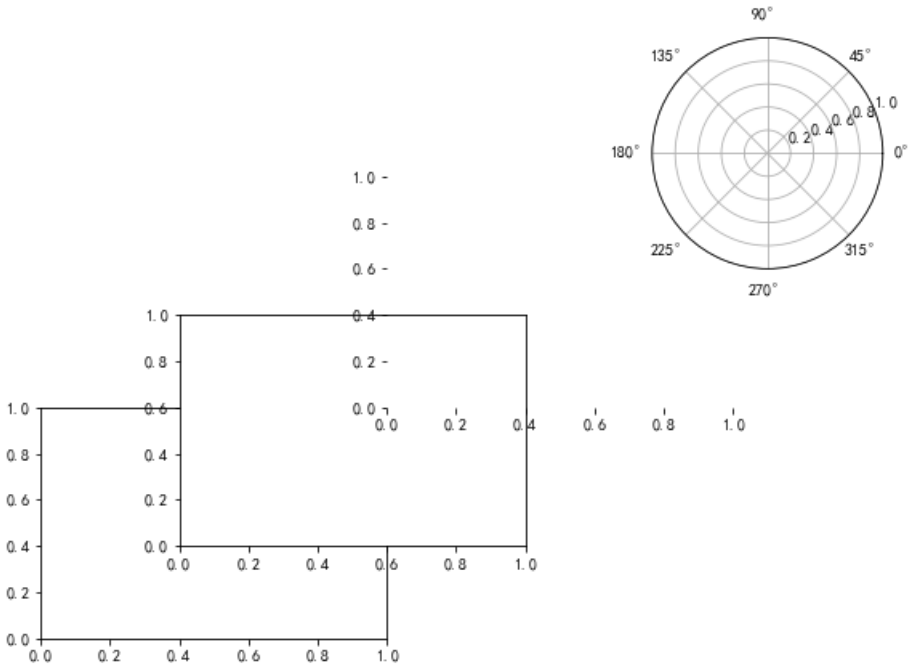
方式2（OO-style）：`Figure.add_axes()` 基于 figure 对象，在 figure 上添加坐标系

```
ax = fig.add_axes(rect=[left, bottom, width, height], projection=None, polar=False,
**kwargs)  # 添加坐标系
ax = fig.add_axes(ax)
ax = fig.delaxes(ax)  # 删除坐标系
返回：axes.Axes 对象
```

In [22]:

```
# 简单的例子
fig = plt.figure()
fig.add_axes([0,0,0.5,0.5])
fig.add_axes([0.2,0.2,0.5,0.5])
fig.add_axes([0.5,0.5,0.5,0.5], frame_on=False)  # 不显示坐标系的矩形补丁
ax = fig.add_axes([0.8,0.8,0.5,0.5], polar=True)
#ax = fig.add_axes([0.8,0.8,0.5,0.5], projection='polar')
fig.delaxes(ax)
fig.add_axes(ax)
```

Out[22]:



3.1.3 创建子图

方式 1（pyplot-style）：`plt.subplots()`，同时创建 figure 和 多幅子图（子图对应的还是坐标系 axes 这个对象）

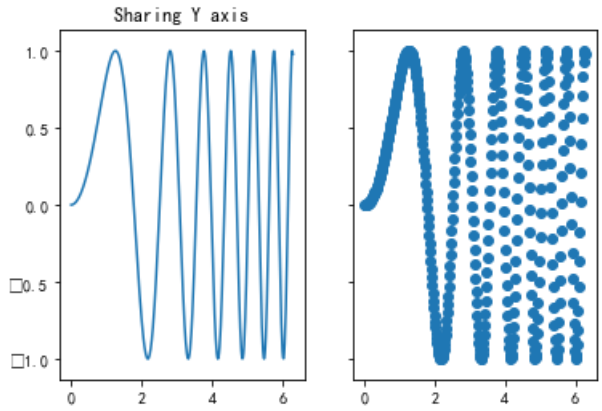
常用形式：

```
fig, ax = plt.subplots()  #默认下只创建一幅子图
fig, axs = plt.subplots(2, 2)  #创建了4幅子图，2*2 的子图矩阵
fig, (ax1, ax2) = plt.subplot(1, 2)  #可通过元组拆分的形式获取相应的子图
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplot(2, 2)
```

In [23]:

```
x = np.linspace(0, 2*np.pi, 400)
y = np.sin(x**2)
fig, (ax1, ax2) = plt.subplots(1, 2, sharey=True)
ax1.plot(x, y)
ax1.set_title('Sharing Y axis')
ax2.scatter(x, y)
```

Out[23]:

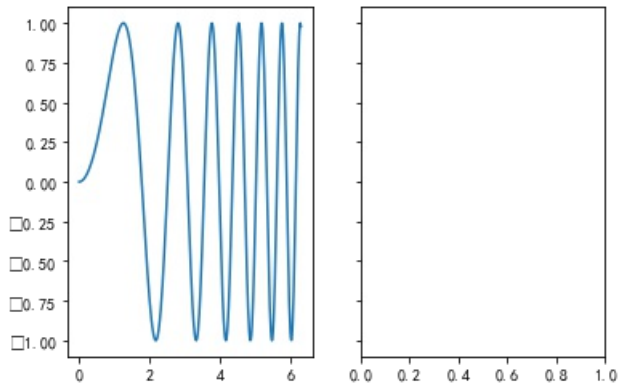


In [24]:

```
fig, axs = plt.subplots(1, 2, sharey=True)
print(axs)  # 多个子图返回的是制图 的list，可通过索引方式获取相应的子图
axs[0].plot(x,y)
```

```
[<matplotlib.axes._subplots.AxesSubplot object at 0x7fd9c7e68f60>
 <matplotlib.axes._subplots.AxesSubplot object at 0x7fd9c75c5208>]
```

Out[24]:



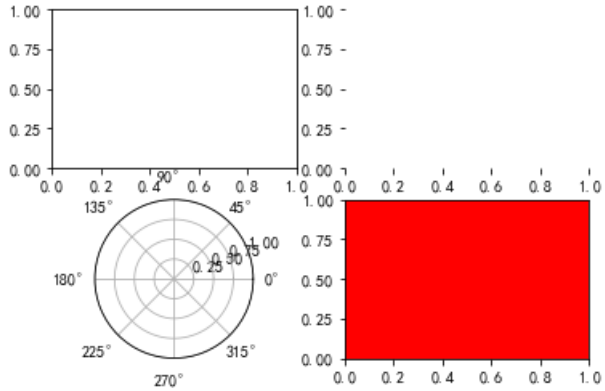
方式 2（pyplot-style）： `plt.subplot()` 往创建的 figure 中添加一副子图

```
plt.subplot(nrows, ncols, index, **kwargs)  # n*n 子图矩阵中的第 index 幅子图
plt.subplot(pos, **kwargs)
plt.subplot(ax)
```

In [25]:

```
plt.subplot(221)
ax1=plt.subplot(2, 2, 1)  # 与上面是等价的

ax2=plt.subplot(222, frame_on=False)  # 没有框架

plt.subplot(223, projection='polar')

plt.subplot(224, sharex=ax2, facecolor='red')  # 与第一幅图的坐标共享x轴

# plt.delaxes(ax2)
# plt.subplot(ax2)
```

Out[25]:



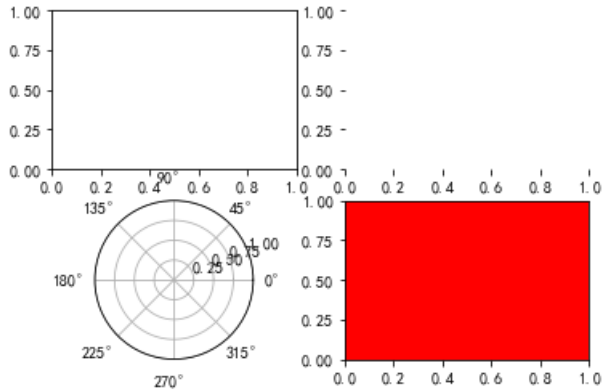
方式 3（OO-style）： `Figure.add_subplot()` 往 figure 中添加子图的坐标系

```
fig.add_subplot(nrows, ncols, index, **kwargs)
fig.add_subplot(pos, **kwargs)
fig.add_subplot(ax)
fig.add_subplot()
```

In [28]:

```
fig = plt.figure()
fig.add_subplot(221)

ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(222, frameon=False)
fig.add_subplot(223, projection='polar')
fig.add_subplot(224, sharex=ax1, facecolor='red')
#fig.delaxes(ax2)
#fig.add_subplot(ax2)
```

Out[28]:



3.1.4 创建分布不规则的子图

方式 1：使用 `gridspec.GridSpec()`函数

首先：需要借助 gridspec 模块中的 GridSpec 函数将 figure 分割成几个基本单元（一个矩阵）

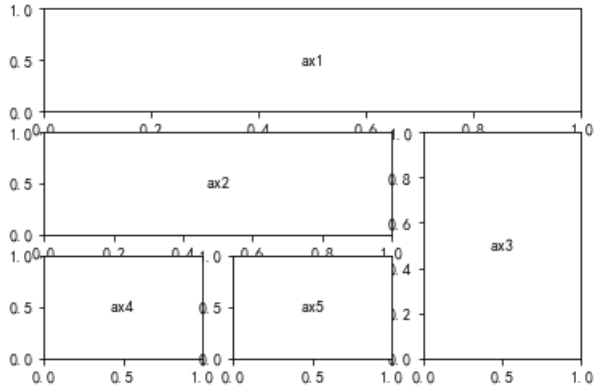
```
import matplotlib.gridspec as gridspec
matplotlib.gridspec.GridSpec(nrows, ncols, figure=None, left=None, bottom=None, righ
t=None, top=None, wspace=None, hspace=None, width_ratios=None, height_ratios=None)
```

然后，结合`plt.subplot()` 函数或`fig.add_subplot()`函数，通过组合不同位置上的基本单元来分布子图的大小和位置。

- 绘制不规则子图各类方法的官方文档： Customizing Figure Layouts Using GridSpec and Other Functions (https://matplotlib.org/tutorials/intermediate/gridspec.html#sphx-glr-tutorials-intermediate-gridspec-py)

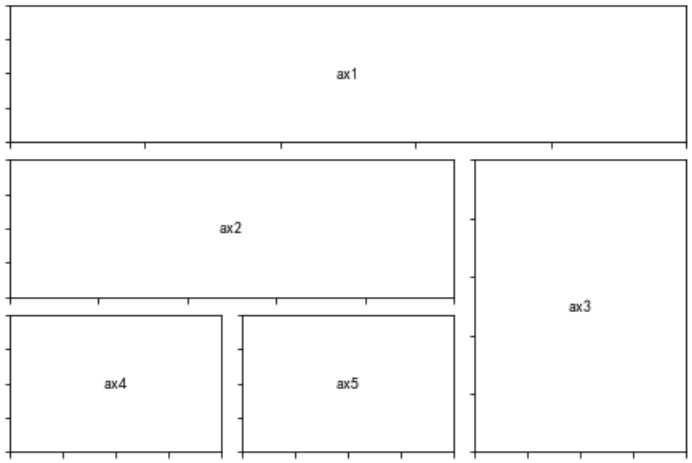
In [37]:

```
# 结合 plt.subplot()的方式
import matplotlib.gridspec as gridspec

gs =gridspec.GridSpec(3,3)
ax1 = plt.subplot(gs[0, :]) # identical to ax1 = plt.subplot(gs.new_subplotspec((0, 0), colspan=3))
ax1.text(0.5, 0.5, 'ax1', va="center", ha="center")
ax2 = plt.subplot(gs[1, :-1])
ax2.text(0.5, 0.5, 'ax2', va="center", ha="center")
ax3 = plt.subplot(gs[1:, -1])
ax3.text(0.5, 0.5, 'ax3', va="center", ha="center")
ax4 = plt.subplot(gs[-1, 0])
ax4.text(0.5, 0.5, 'ax4', va="center", ha="center")
ax5 =plt.subplot(gs[-1, -2])
ax5.text(0.5, 0.5, 'ax5', va="center", ha="center")

plt.show()
```



In [34]:

```
# 结合 fig.add_subplot() 的方式
import matplotlib.gridspec as gridspec

def format_axes(fig):
    for i, ax in enumerate(fig.axes):
        ax.text(0.5, 0.5, "ax%d" % (i+1), va="center", ha="center")
        ax.tick_params(labelbottom=False, labelleft=False)  # 隐藏坐标刻度标签

fig = plt.figure(constrained_layout=True)
gs =gridspec. GridSpec(3, 3, figure=fig)
ax1 = fig.add_subplot(gs[0, :]) # identical to ax1 = plt.subplot(gs.new_subplotspec((0, 0),
colspan=3))
ax2 = fig.add_subplot(gs[1, :-1])
ax3 = fig.add_subplot(gs[1:, -1])
ax4 = fig.add_subplot(gs[-1, 0])
ax5 = fig.add_subplot(gs[-1, -2])

format_axes(fig)
```

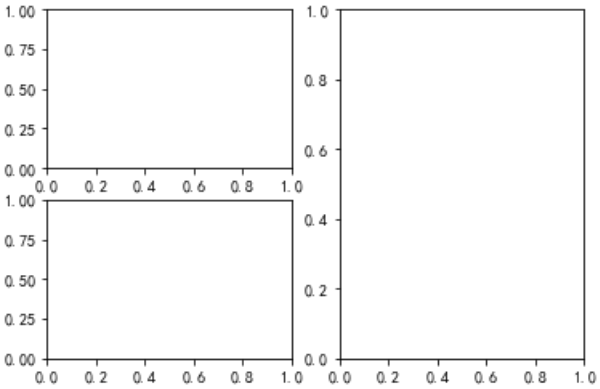


方式 2：在 figure 上直接调用`fig.add_gridspec()`函数进行图形分割，然后使用`fig.add_subplot()`获取绘图的组合区域

```
fig.add_gridspec(self, nrows, ncols, **kwargs)
```

In [38]:

```
fig = plt.figure()
gs = fig.add_gridspec(2, 2)
ax1 = fig.add_subplot(gs[0, 0])
ax2 = fig.add_subplot(gs[1, 0])
ax3 = fig.add_subplot(gs[:, 1])  # 合并两行
```



3.1.5 总结

- **无论是 pyplot-style 绘图方式还是 OO-style 绘图方式，都需要调用 `pyplot` 接口。**前者直接调用 pyplot 接口中的函数进行画图，后者需要借助 pyplot 接口来显示的构建最顶层的绘图元素 Figure （Figure is the top level container for all the plot elements.）或是显示的构建 坐标系 Axes （The Axes contains most of the figure elements: Axis, Tick, Line2D, Text, Polygon, etc., and sets the coordinate system.），若不显示的构建 Figure 对象或 Axes 对象，就无法调用这两个对象的各类绘图方法进行 OO-style 绘图。
- **推荐使用一次性生成图和子图坐标系的方式 `fig, ax = plt.subplots()` 来画图，操作方便且高效。**

3.2 如何设置坐标轴和刻度？（基于 OO-style）

- **先弄清楚 5 个概念：**

  major ticks : 主刻度

  minor ticks ： 次刻度

  tick locator : 刻度定位器，用于设置刻度的位置

  tick formatting：刻度格式，用于设置刻度标签的格式

  axis label : 轴标签，用于设置 轴 的名字

- **matplotlib.axis 模块中包含 2 个对象：**

  对象1：轴（matplotlib.axis.Axis），包括 x 轴（matplotlib.axis.XAxis）和 y 轴（matplotlib.axis.YAxis），对应一系列方法（get_xxx()、set_xxx()）；

  对象2：刻度（matplotlib.axis.Tick），包括 x 轴 上的刻度（matplotlib.axis.XTick） 和 y 轴上的刻度（matplotlib.axis.YTick），同样对应一系列方法（get_xxx()、set_xxx()）。

- matplotlib.axis 官方文档： https://matplotlib.org/api/axis_api.html (https://matplotlib.org/api/axis_api.html)
- matplotlib.ticker 官方文档： https://matplotlib.org/api/ticker_api.html (https://matplotlib.org/api/ticker_api.html)

3.2.1 设置轴刻度位置（主刻度位置和次刻度位置）

```
ax.xaxis.set_major_locator(xmajor_locator)  # 设置 x 轴的主次刻度位置
ax.xaxis.set_minor_locator(xminor_locator)
ax.yaxis.set_major_locator(ymajor_locator)  # 设置 y 轴的主次刻度位置
ax.yaxis.set_minor_locator(yminor_locator)
```

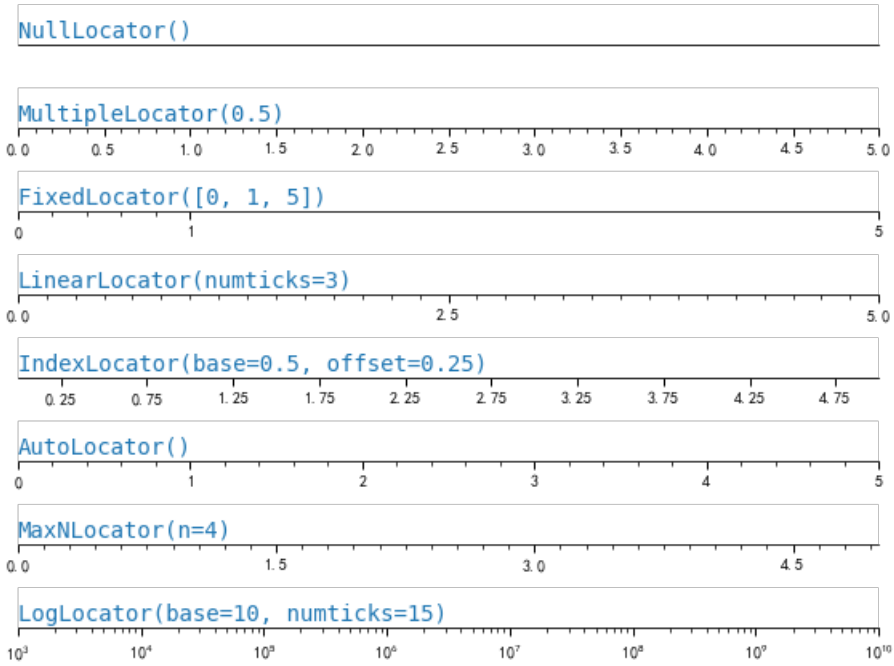
当然也有对应的 get_major_locator() 和 get_minor_locator() 。

- **matplotlib.ticker** 模块中可调用的定位器

定位器	含义
AutoLocator	自动定位器，大多数绘图的默认刻度线定位器。
MaxNLocator	在最合适的位置找到带有刻度的最大间隔数。
LinearLocator	线性定位器，基于刻度线的个数，从最小到最大均匀分布刻度间隔。
LogLocator	对数定位器，刻度间隔从最小到最大取对数。
MultipleLocator	多重定位器，根据刻度间隔进行定位，刻度和范围是间隔的倍数，既适用于整数，也适用于浮点数。
FixedLocator	固定定位器，刻度线位置是固定的。
IndexLocator	索引定位器，适用于带有索引数据的刻度进行定位，例如 where x = range(len(y))
NullLocator	空定位器，不对刻度进行定位，不会显示刻度。
SymmetricalLogLocator	与符号规范一起使用的定位器。
LogitLocator	用于 logit 缩放的定位器。
OldAutoLocator	旧的自动定位器，选择一个MultipleLocator并动态重新分配它，以便在导航期间智能显示刻度。
AutoMinorLocator	轴为线性且主刻度线等距分布时，副刻度线的定位器，将主刻度间隔细分为指定数量的次间隔，根据主间隔默认为4或5。

In [39]:

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker   # matplotlib.ticker 模块中


def setup(ax, title):
    """Set up common parameters for the Axes in the example."""
    # only show the bottom spine
    ax.yaxis.set_major_locator(ticker.NullLocator())
    ax.spines['right'].set_color('none')
    ax.spines['left'].set_color('none')
    ax.spines['top'].set_color('none')

    ax.xaxis.set_ticks_position('bottom')
    ax.tick_params(which='major', width=1.00, length=5)
    ax.tick_params(which='minor', width=0.75, length=2.5)
    ax.set_xlim(0, 5)  # 默认整个x轴的取值为 0-5
    ax.set_ylim(0, 1)
    ax.text(0.0, 0.2, title, transform=ax.transAxes,
            fontsize=14, fontname='Monospace', color='tab:blue')


fig, axs = plt.subplots(8, 1, figsize=(8, 6))

# Null Locator
setup(axs[0], title="NullLocator()")
axs[0].xaxis.set_major_locator(ticker.NullLocator())
axs[0].xaxis.set_minor_locator(ticker.NullLocator())

# Multiple Locator
setup(axs[1], title="MultipleLocator(0.5)")
axs[1].xaxis.set_major_locator(ticker.MultipleLocator(0.5))  # 基于间隔进行定位
axs[1].xaxis.set_minor_locator(ticker.MultipleLocator(0.1))

# Fixed Locator
setup(axs[2], title="FixedLocator([0, 1, 5])")
axs[2].xaxis.set_major_locator(ticker.FixedLocator([0, 1, 5]))  # 基于输入的刻度位置固定刻度
axs[2].xaxis.set_minor_locator(ticker.FixedLocator(np.linspace(0.2, 0.8, 4)))

# Linear Locator
setup(axs[3], title="LinearLocator(numticks=3)")  # 基于刻度个数分配刻度位置
axs[3].xaxis.set_major_locator(ticker.LinearLocator(3))
axs[3].xaxis.set_minor_locator(ticker.LinearLocator(31))

# Index Locator
setup(axs[4], title="IndexLocator(base=0.5, offset=0.25)")
axs[4].plot(range(0, 5), [0]*5, color='white')
axs[4].xaxis.set_major_locator(ticker.IndexLocator(base=0.5, offset=0.25))     # 从 0.25 开始定位，刻度间隔
为 0.5

# Auto Locator
setup(axs[5], title="AutoLocator()")
axs[5].xaxis.set_major_locator(ticker.AutoLocator())
axs[5].xaxis.set_minor_locator(ticker.AutoMinorLocator())

# MaxN Locator
setup(axs[6], title="MaxNLocator(n=4)")
axs[6].xaxis.set_major_locator(ticker.MaxNLocator(4))
axs[6].xaxis.set_minor_locator(ticker.MaxNLocator(40))

# Log Locator
setup(axs[7], title="LogLocator(base=10, numticks=15)")
axs[7].set_xlim(10**3, 10**10)
axs[7].set_xscale('log')
axs[7].xaxis.set_major_locator(ticker.LogLocator(base=10, numticks=15))

plt.tight_layout()
plt.show()
```



3.2.2 设置轴刻度样式（主刻度样式和次刻度样式）

```
ax.xaxis.set_major_formatter(xmajor_formatter)  # 设置 x 轴的主次刻度样式
ax.xaxis.set_minor_formatter(xminor_formatter)
ax.yaxis.set_major_formatter(ymajor_formatter)  # 设置 y 轴的主次刻度样式
ax.yaxis.set_minor_formatter(yminor_formatter)
```

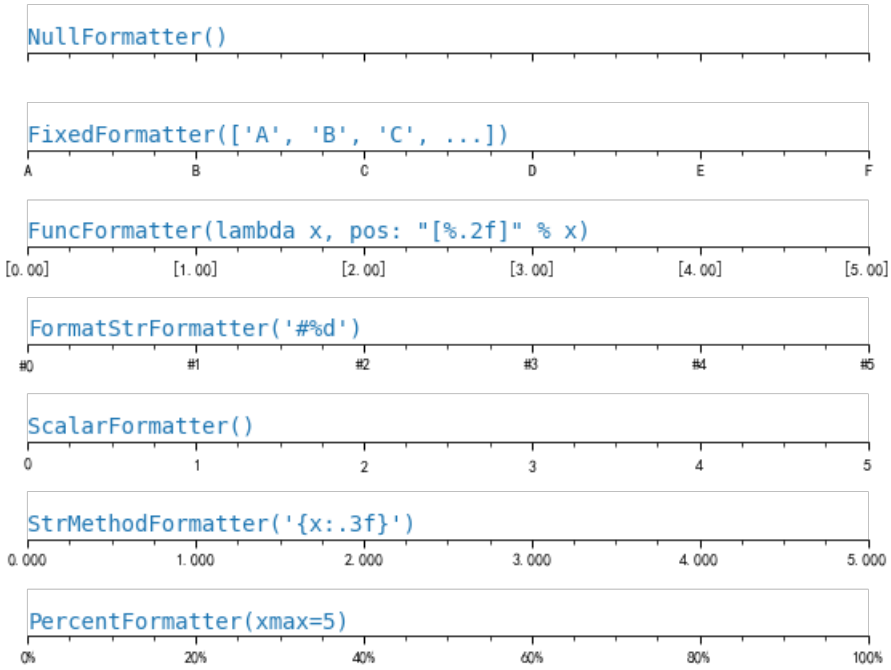
当然也有对应的 get_major_formatter() 和 get_minor_formatter() 。

- **`matplotlib.ticker`** 模块中可调用的格式化函数

刻度格式	含义
NullFormatter	刻度线上没有标签。
`IndexFormatter`	从标签列表中设置字符串。
`FixedFormatter`	固定格式器，手动设置标签的字符串。
`FuncFormatter`	用户定义的功能设置标签。
StrMethodFormatter	使用字符串格式方法设置刻度样式。
`FormatStrFormatter`	使用旧式的 sprintf 格式字符串。
ScalarFormatter	标量的默认格式化程序：自动选择格式字符串。
`LogFormatter`	日志轴的格式化程序。
LogFormatterExponent	使用指数= log_base（值）格式化对数轴的值。
LogFormatterMathtext	使用Math文本使用exponent = log_base（value）格式化对数轴的值。
LogFormatterSciNotation	使用科学计数法设置对数轴的值格式。
LogitFormatter	概率格式器。
EngFormatter	以工程符号格式格式化标签。
`PercentFormatter`	将标签格式化为百分比。

In [40]:

```
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker


def setup(ax, title):
    """Set up common parameters for the Axes in the example."""
    # only show the bottom spine
    ax.yaxis.set_major_locator(ticker.NullLocator())
    ax.spines['right'].set_color('none')
    ax.spines['left'].set_color('none')
    ax.spines['top'].set_color('none')

    # define tick positions
    ax.xaxis.set_major_locator(ticker.MultipleLocator(1.00))
    ax.xaxis.set_minor_locator(ticker.MultipleLocator(0.25))

    ax.xaxis.set_ticks_position('bottom')
    ax.tick_params(which='major', width=1.00, length=5)
    ax.tick_params(which='minor', width=0.75, length=2.5, labelsize=10)
    ax.set_xlim(0, 5)
    ax.set_ylim(0, 1)
    ax.text(0.0, 0.2, title, transform=ax.transAxes,
            fontsize=14, fontname='Monospace', color='tab:blue')


fig, axs = plt.subplots(7, 1, figsize=(8, 6))

# Null formatter
setup(axs[0], title="NullFormatter()")
axs[0].xaxis.set_major_formatter(ticker.NullFormatter())

# Fixed formatter
setup(axs[1], title="FixedFormatter(['A', 'B', 'C', ...])")
# FixedFormatter should only be used together with FixedLocator.
# Otherwise, one cannot be sure where the labels will end up.
positions = [0, 1, 2, 3, 4, 5]
labels = ['A', 'B', 'C', 'D', 'E', 'F']
axs[1].xaxis.set_major_locator(ticker.FixedLocator(positions))
axs[1].xaxis.set_major_formatter(ticker.FixedFormatter(labels))   # 对固定的刻度一对一设置刻度标签样式


# FuncFormatter can be used as a decorator
@ticker.FuncFormatter
def major_formatter(x, pos):
    return "[%.2f]" % x


setup(axs[2], title='FuncFormatter(lambda x, pos: "[%.2f]" % x)')
axs[2].xaxis.set_major_formatter(major_formatter)

# FormatStr formatter
setup(axs[3], title="FormatStrFormatter('#%d')")
axs[3].xaxis.set_major_formatter(ticker.FormatStrFormatter("#%d"))   # 基于字符串格式化函数来设置

# Scalar formatter
setup(axs[4], title="ScalarFormatter()")
axs[4].xaxis.set_major_formatter(ticker.ScalarFormatter(useMathText=True))

# StrMethod formatter
setup(axs[5], title="StrMethodFormatter('{x:.3f}')")
axs[5].xaxis.set_major_formatter(ticker.StrMethodFormatter("{x:.3f}"))

# Percent formatter
setup(axs[6], title="PercentFormatter(xmax=5)")
axs[6].xaxis.set_major_formatter(ticker.PercentFormatter(xmax=5))

plt.tight_layout()
plt.show()
```



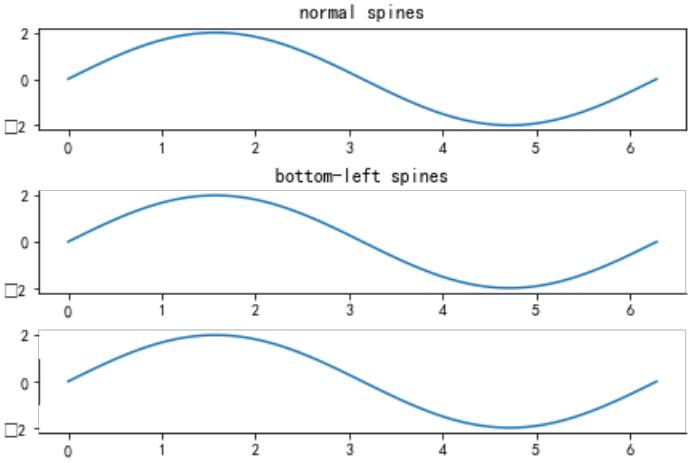
3.2.3 关于轴脊线（spine）的设置

- 什么是轴脊线：轴脊线-记录数据区域边界的线（An axis spine -- the line noting the data area boundaries）
- 常用形式：

```
ax.spines[direction].set_visible(bool)  # 是否可见
ax.spines[direction].set_bounds(low=None, high=None)  # 设置边界
ax.spines[direction].set_color(c)  # 设置颜色
ax.spines[direction].set_position(position=(position type, amount))  # 设置位置
```

In [41]:

```
x = np.linspace(0, 2 * np.pi, 100)
y = 2 * np.sin(x)

fig, (ax0, ax1, ax2) = plt.subplots(nrows=3, constrained_layout=True)

ax0.plot(x, y)
ax0.set_title('normal spines')

ax1.plot(x, y)
ax1.set_title('bottom-left spines')
ax1.spines['right'].set_visible(False) # 是否可见
ax1.spines['top'].set_visible(False)
ax1.yaxis.set_ticks_position('left')  # 只显示y轴左侧的刻度
ax1.xaxis.set_ticks_position('bottom')  # 只显示x轴底部的刻度

ax2.plot(x, y)
ax2.spines['left'].set_bounds(-1, 1)  # 设置 轴脊线 的范围
ax2.spines['right'].set_visible(False)
ax2.spines['top'].set_visible(False)
ax2.yaxis.set_ticks_position('left')
ax2.xaxis.set_ticks_position('bottom')

plt.show()
```

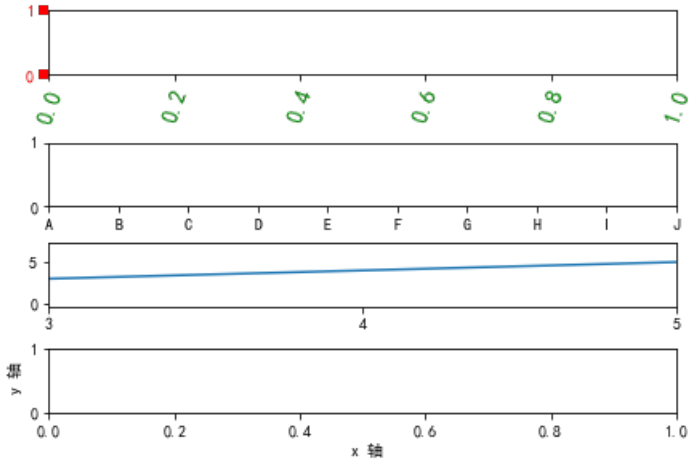


3.2.4 关于轴或刻度的其他常用设置

```
Axes.tick_params(self, axis='both', **kwargs)  # 更改刻度线，刻度线标签和网格线的样式
Axes.set_xticks(self, ticks, *, minor=False)  # 设置 x 刻度的位置
Axes.set_yticks(self, ticks, *, minor=False)  # 设置 y 刻度的位置
Axes.set_xticklabels(self, labels, fontdict=None, minor=False, **kwargs)  # 设置 x 刻度标签
Axes.set_yticklabels(self, labels, fontdict=None, minor=False, **kwargs)  # 设置 y 刻度标签
Axes.set_xlim((left, right))  # 设置 x 轴的可视化边界
Axes.set_ylim((left, right))  # 设置 y 轴的可视化边界
Axes.set_xlabel(self, xlabel, fontdict=None, labelpad=None, **kwargs)  # 设置 x 轴的标签
Axes.set_ylabel(self, yabel, fontdict=None, labelpad=None, **kwargs)  # 设置 y轴的标签
```

In [60]:

```
fig, (ax0, ax1, ax2,ax3) = plt.subplots(nrows=4, constrained_layout=True)

ax0.tick_params(axis='y', direction='out',  length=6, width=6, colors='r', grid_color='r', grid_alpha=0
.5)
ax0.tick_params(axis='x', direction='out',  labelsize='x-large', labelcolor='g', labelrotation=70)

ax1.set_xticks(range(10))
ax1.set_xticklabels(list('ABCDEFGHIJ'))

ax2.plot(range(8))
ax2.set_xticks(range(10))
ax2.set_xlim((3,5))  # 只绘制了 3-5 这个范围的内容

ax3.set_xlabel(' x 轴')
ax3.set_ylabel(' y 轴')

plt.show()
```



3.3 如何绘制各类图形（基于 OO-style）

建议大家采用面向对象的编程方式，以坐标系为基础绘制图形（前提：先显示的创建好 Figure 和 Axes）：ax.plot()、ax.bar()、ax.scatter()、ax.pie() .....

- matplotlib.axes.Axes 官方文档： https://matplotlib.org/api/axes_api.html#matplotlib.axes.Axes (https://matplotlib.org/api/axes_api.html#matplotlib.axes.Axes)

3.3.1 常用的绘图函数

对于绘图函数中各参数的含义直接参考官方文档：

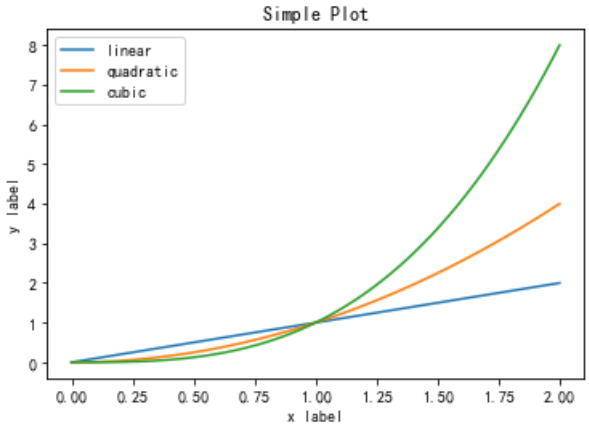
```
Axes.plot(self, *args, scalex=True, scaley=True, data=None, **kwargs)  # 绘制曲线

Axes.scatter(self, x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None, vmax=None, alpha=None, linewidths=None, verts=<deprecated parameter>, edgecolors=None, *, plotnonfinite=False, data=None, **kwargs)  # 绘制散点图

Axes.bar(self, x, height, width=0.8, bottom=None, *, align='center', data=None, **kwargs)  # 绘制柱状图（垂直）

Axes.barh(self, y, width, height=0.8, left=None, *, align='center', **kwargs)  # 绘制柱状图（水平）

Axes.pie(self, x, explode=None, labels=None, colors=None, autopct=None, pctdistance=0.6, shadow=False, labeldistance=1.1, startangle=None, radius=None, counterclock=True, wedgeprops=None, textprops=None, center=(0, 0), frame=False, rotatelabels=False, *, data=None)  # 绘制饼图

Axes.step(self, x, y, *args, where='pre', data=None, **kwargs)   # 绘制阶梯图

Axes.stem(self, *args, linefmt=None, markerfmt=None, basefmt=None, bottom=0, label=None, use_line_collection=False, data=None)  # 绘制茎图（棒棒糖）

Axes.stackplot(axes, x, *args, labels=(), colors=None, baseline='zero', data=None, **kwargs)  # 绘制堆积图

Axes.hist(self, x, bins=None, range=None, density=False, weights=None, cumulative=False, bottom=None, histtype='bar', align='mid', orientation='vertical', rwidth=None, log=False, color=None, label=None, stacked=False, *, data=None, **kwargs)  # 绘制直方图

Axes.boxplot(self, x, notch=None, sym=None, vert=None, whis=None, positions=None, widths=None, patch_artist=None, bootstrap=None, usermedians=None, conf_intervals=None, meanline=None, showmeans=None, showcaps=None, showbox=None, showfliers=None, boxprops=None, labels=None, flierprops=None, medianprops=None, meanprops=None, capprops=None, whiskerprops=None, manage_ticks=True, autorange=False, zorder=None, *, data=None)
# 绘制箱线图

Axes.violinplot(self, dataset, positions=None, vert=True, widths=0.5, showmeans=False, showextrema=True, showmedians=False, quantiles=None, points=100, bw_method=None, *, data=None)  # 绘制小提琴图

Axes.table(ax, cellText=None, cellColours=None, cellLoc='right', colWidths=None, rowLabels=None, rowColours=None, rowLoc='left', colLabels=None, colColours=None, colLoc='center', loc='bottom', bbox=None, edges='closed', **kwargs)  # 绘制表格

Axes.annotate(self, s, xy, *args, **kwargs)  # 绘制标签
Axes.text(self, x, y, s, fontdict=None, withdash=<deprecated parameter>, **kwargs)
```

3.3.2 如何叠加绘图？

绘制图中常常遇到在一个坐标系中绘制多条曲线或多种图形的情况，只需：**画完一条接着画另一条。**

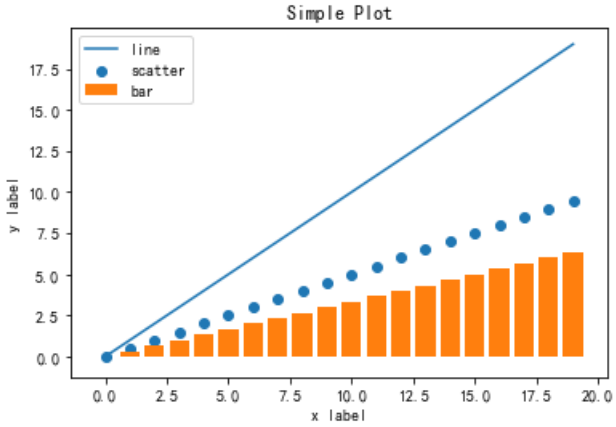
In [61]:

```
# 同一类型，多条

fig, ax = plt.subplots()  # 同时创建了 图 figure 和 坐标系 axes

x = np.linspace(0, 2, 100)
ax.plot(x, x, label='linear')
ax.plot(x, x**2, label='quadratic')
ax.plot(x, x**3, label='cubic')
ax.set_xlabel('x label')
ax.set_ylabel('y label')
ax.set_title("Simple Plot")
ax.legend()

plt.show()
```



In [78]:

```
# 不同类型

fig, ax = plt.subplots()  # 同时创建了 图 figure 和 坐标系 axes

x = np.arange(0,20)
ax.plot(x, x, label='line')
ax.scatter(x, x/2, label='scatter')
ax.bar(x, x/3, label='bar')

ax.set_xlabel('x label')
ax.set_ylabel('y label')
ax.set_title("Simple Plot")
ax.legend()

plt.show()
```

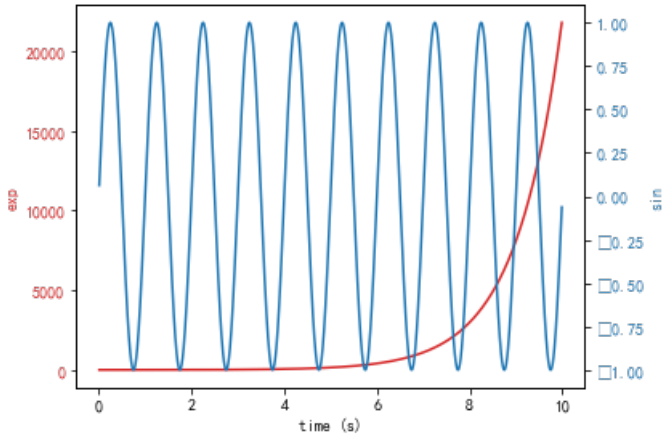


3.3.3 如何绘制双坐标图？

利用**Axes.twinx()** 或 **Axes.twiny()** 创建 双轴图，一般**适用于分别绘制不同数量级的数据**。

In [79]:

```
t = np.arange(0.01, 10.0, 0.01)
data1 = np.exp(t)
data2 = np.sin(2 * np.pi * t)

fig, ax1 = plt.subplots()

color = 'tab:red'
ax1.set_xlabel('time (s)')
ax1.set_ylabel('exp', color=color)
ax1.plot(t, data1, color=color)
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()  # 创建 x 轴的兄弟轴

color = 'tab:blue'
ax2.set_ylabel('sin', color=color)
ax2.plot(t, data2, color=color)
ax2.tick_params(axis='y', labelcolor=color)

fig.tight_layout()
plt.show()
```



3.4 如何设置图中的图例和标题（基于OO-style）

3.4.1 设置图例

- **图例有关概念：**

  图例条目（legend entry）：图例由一个或多个图例条目组成，一项条目仅由一个键和一个标签组成；

  图例键（legend key）：每个图例标签左侧的图案标记；

  图例标签（legend label）：描述图例键的标签；

  图例手柄（legend handle）：用于在图例中生成相应条目的原始对象。

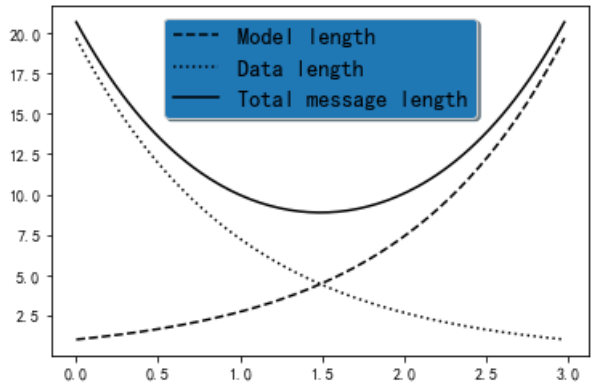
- **方式1**：通过设置绘图函数中的 `label` 参数， 调用 **`ax.legend()`** 自动检测要在图例中显示的元素；
- **方式2**：在**`ax.legend(label)`** 中设置图标签，在图形较多时，容易混淆，不建议采用这种方式；
- **方式3**：通过定义图例中的元素来设置图例：**`ax.legend(handles, labels)`**，可通过 ax.get_legend_handles_labels() 获取图例的 handle 和 label：

```
handles, labels = ax.get_legend_handles_labels()
ax.legend(handles, labels)
```

- 图例设置教程官方文档：https://matplotlib.org/tutorials/intermediate/legend_guide.html#sphx-glr-tutorials-intermediate-legend-guide-py (https://matplotlib.org/tutorials/intermediate/legend_guide.html#sphx-glr-tutorials-intermediate-legend-guide-py)
- legend() 函数官方文档：https://matplotlib.org/api/_as_gen/matplotlib.pyplot.legend.html#matplotlib.pyplot.legend (https://matplotlib.org/api/_as_gen/matplotlib.pyplot.legend.html#matplotlib.pyplot.legend)

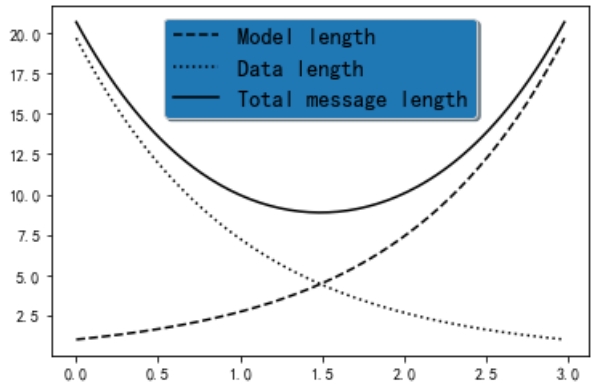
In [92]:

```
# 方式1：自动检测并显示

# Make some fake data.
a = b = np.arange(0, 3, .02)
c = np.exp(a)
d = c[::-1]

# Create plots with pre-defined labels.
fig, ax = plt.subplots()
ax.plot(a, c, 'k--', label='Model length')
ax.plot(a, d, 'k:', label='Data length')
ax.plot(a, c + d, 'k', label='Total message length')

legend = ax.legend(loc='upper center', shadow=True, fontsize='x-large')

# Put a nicer background color on the legend.
legend.get_frame().set_facecolor('C0')

plt.show()
```



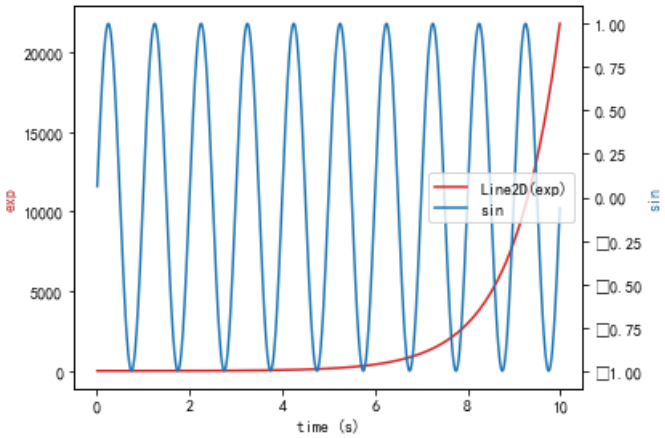
In [93]:

```
# 方式2：单独设置图例标签
# Make some fake data.
a = b = np.arange(0, 3, .02)
c = np.exp(a)
d = c[::-1]

# Create plots with pre-defined labels.
fig, ax = plt.subplots()
ax.plot(a, c, 'k--' )
ax.plot(a, d, 'k:' )
ax.plot(a, c + d, 'k')

legend = ax.legend(['Model length','Data length','Total message length' ],loc='upper center', shadow=True, fontsize='x-large')
legend.get_frame().set_facecolor('C0')

plt.show()
```



In [102]:

```
# 方式3：通过 handle 和 label 设置图例
t = np.arange(0.01, 10.0, 0.01)
data1 = np.exp(t)
data2 = np.sin(2 * np.pi * t)

fig, ax1 = plt.subplots()

color = 'tab:red'
ax1.set_xlabel('time (s)')
ax1.set_ylabel('exp', color=color)
ax1.plot(t, data1, color=color,label='exp')

ax2 = ax1.twinx()  # 创建 x 轴的兄弟轴

color = 'tab:blue'
ax2.set_ylabel('sin', color=color)
ax2.plot(t, data2, color=color,label='sin')

h1,b1 = ax1.get_legend_handles_labels()
h2,b2 = ax2.get_legend_handles_labels()
plt.legend(h1+h2,h1+b2)
#fig.legend(h1+h2,h1+b2)

fig.tight_layout()
plt.show()
```



3.4.2 设置图标题

- **情况1：给坐标系添加标题（也适用于子图）**

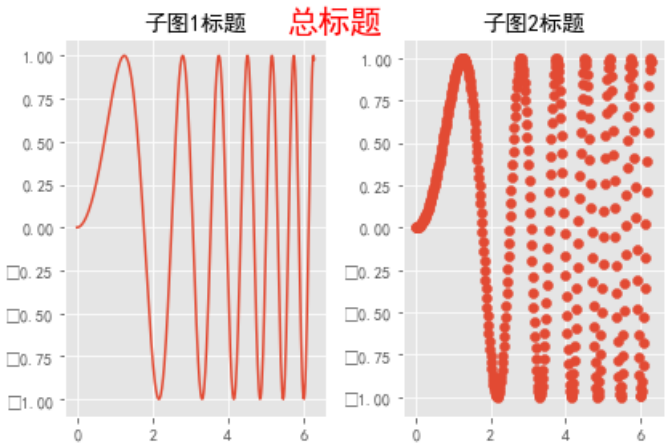
```
Axes.set_title(self, label, fontdict=None, loc=None, pad=None, **kwargs)
```

- **情况2：给整幅图添加总标题（适用于图）**

```
Figure.suptitle(self, t, **kwargs)
```

In [31]:

```
x = np.linspace(0, 2*np.pi, 400)
y = np.sin(x**2)
fig, (ax1, ax2) = plt.subplots(1, 2)
ax1.plot(x, y)
ax1.set_title('子图1标题')
ax2.scatter(x, y)
ax2.set_title('子图2标题')
fig.suptitle('总标题',fontsize=20,c='r')
fig.tight_layout()
plt.show()
```



四、其他需注意画图事项

4.1 画图风格

- 画图风格展示官方文档：https://matplotlib.org/gallery/style_sheets/style_sheets_reference.html#sphx-glr-gallery-style-sheets-style-sheets-reference-py (https://matplotlib.org/gallery/style_sheets/style_sheets_reference.html#sphx-glr-gallery-style-sheets-style-sheets-reference-py)
- 答印所有的风格：**`print(plt.style.available)`**
- 通过 **`plt.style.context(style)`**，在with 代码块中进行局部设置（只对 with 代码块中绘制的图形有效）
- 通过 **`plt.style.use(style)`** 来进行全局设置.
- 通过保留默认风格来恢复之前的风格：**`plt.style.use('default')`**

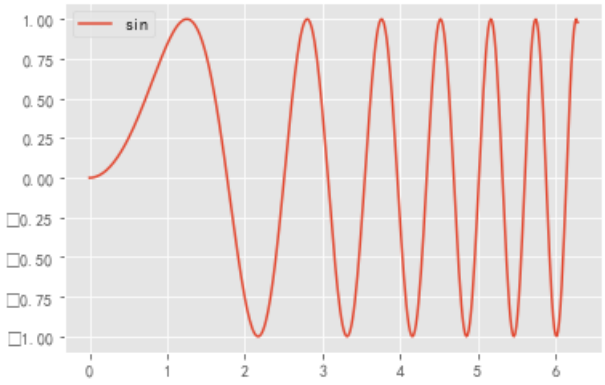
In [109]:

```
print(plt.style.available)  # 打印风格
```

```
['Solarize_Light2', '_classic_test', 'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', '
ggplot', 'grayscale', 'seaborn-bright', 'seaborn-colorblind', 'seaborn-dark-palette', 'seaborn-dark',
'seaborn-darkgrid', 'seaborn-deep', 'seaborn-muted', 'seaborn-notebook', 'seaborn-paper', 'seaborn-pas
tel', 'seaborn-poster', 'seaborn-talk', 'seaborn-ticks', 'seaborn-white', 'seaborn-whitegrid', 'seabor
n', 'tableau-colorblind10']
```

In [26]:

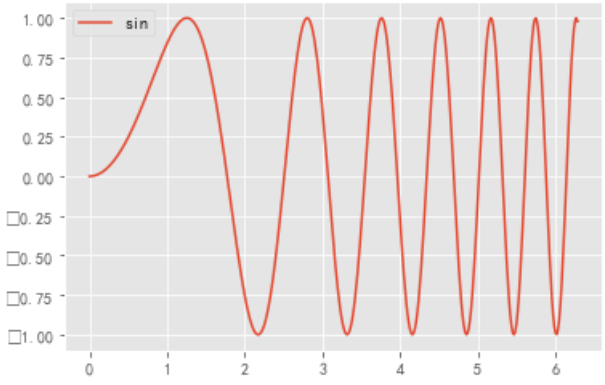
```
x = np.linspace(0, 2*np.pi, 400)
y = np.sin(x**2)
with plt.style.context('ggplot'):    #将use换成context
    plt.plot(x,y,label='sin')
    plt.legend()
    pass
```



In [30]:

```
x = np.linspace(0, 2*np.pi, 400)
y = np.sin(x**2)
plt.plot(x,y,label='sin')  # with 之外的风格仍保持不变
plt.legend()
```

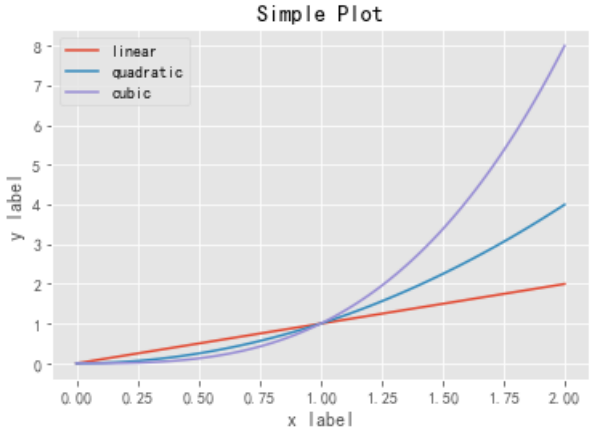
Out[30]:



In [28]:

```
plt.style.use('ggplot')    # 采用 ggplot 的绘图风格
x = np.linspace(0, 2, 100)
plt.plot(x, x, label='linear')
plt.plot(x, x**2, label='quadratic')
plt.plot(x, x**3, label='cubic')
plt.xlabel('x label')
plt.ylabel('y label')
plt.title("Simple Plot")
plt.legend()
```

Out[28]:



In [107]:

```
plt.style.use('default')  # 保留默认风格
```

4.2 关于配置文件 rcParams

- rcParams 配置文件可用来自定义图形的各种默认属性，称之为 rc 配置或 rc 参数。通过 rc 参数可以修改默认的属性，包括窗体大小、每英寸的点数、线条宽度、颜色、样式、坐标轴、坐标和网络属性、文本、字体等；
- rcParams 和 style 配置官方文档：https://matplotlib.org/tutorials/introductory/customizing.html#customizing-with-matplotlibrc-files (https://matplotlib.org/tutorials/introductory/customizing.html#customizing-with-matplotlibrc-files)
- 如何查看默认的配置文件？

```
print(matplotlib.rc_params())
print(matplotlib.rcParamsDefault)    # plt.rcParamsDefault
print(matplotlib.rcParams)     # plt.rcParams
```

- 修改配置文件

```
plt.rcParams['lines.linewidth'] = 2, plt.rcParams['lines.color'] = 'r'    # 方式1

plt.rc('lines', linewidth=4, color='g')    # 方式2
```

- 恢复默认参数：**plt.rcdefaults()**

In [32]:

```
print(plt.rcParams)  # 打印默认的配置
```

```
_internal.classic_mode: False
agg.path.chunksize: 0
animation.avconv_args: []
animation.avconv_path: avconv
animation.bitrate: -1
animation.codec: h264
animation.convert_args: []
```

```
animation.convert_args: []
animation.convert_path: convert
animation.embed_limit: 20.0
animation.ffmpeg_args: []
animation.ffmpeg_path: ffmpeg
animation.frame_format: png
animation.html: none
animation.html_args: []
animation.writer: ffmpeg
axes.autolimit_mode: data
axes.axisbelow: True
axes.edgecolor: white
axes.facecolor: #E5E5E5
axes.formatter.limits: [-7, 7]
axes.formatter.min_exponent: 0
axes.formatter.offset_threshold: 4
axes.formatter.use_locale: False
axes.formatter.use_mathtext: False
axes.formatter.useoffset: True
axes.grid: True
axes.grid.axis: both
axes.grid.which: major
axes.labelcolor: #555555
axes.labelpad: 4.0
axes.labelsize: large
axes.labelweight: normal
axes.linewidth: 1.0
axes.prop_cycle: cycler('color', ['#E24A33', '#348ABD', '#988ED5', '#777777', '#FBC15E', '#8EBA42', '#
FFB5B8'])
axes.spines.bottom: True
axes.spines.left: True
axes.spines.right: True
axes.spines.top: True
axes.titlepad: 6.0
axes.titlesize: x-large
axes.titleweight: normal
axes.unicode_minus: True
axes.xmargin: 0.05
axes.ymargin: 0.05
axes3d.grid: True
backend: module://ipykernel.pylab.backend_inline
backend_fallback: True
boxplot.bootstrap: None
boxplot.boxprops.color: black
boxplot.boxprops.linestyle: -
boxplot.boxprops.linewidth: 1.0
boxplot.capprops.color: black
boxplot.capprops.linestyle: -
boxplot.capprops.linewidth: 1.0
boxplot.flierprops.color: black
boxplot.flierprops.linestyle: none
boxplot.flierprops.linewidth: 1.0
boxplot.flierprops.marker: o
boxplot.flierprops.markeredgecolor: black
boxplot.flierprops.markeredgewidth: 1.0
boxplot.flierprops.markerfacecolor: none
boxplot.flierprops.markersize: 6.0
boxplot.meanline: False
boxplot.meanprops.color: C2
boxplot.meanprops.linestyle: --
boxplot.meanprops.linewidth: 1.0
boxplot.meanprops.marker: ^
boxplot.meanprops.markeredgecolor: C2
boxplot.meanprops.markerfacecolor: C2
boxplot.meanprops.markersize: 6.0
boxplot.medianprops.color: C1
boxplot.medianprops.linestyle: -
boxplot.medianprops.linewidth: 1.0
boxplot.notch: False
boxplot.patchartist: False
boxplot.showbox: True
boxplot.showcaps: True
boxplot.showfliers: True
boxplot.showmeans: False
boxplot.vertical: True
boxplot.whiskerprops.color: black
boxplot.whiskerprops.linestyle: -
boxplot.whiskerprops.linewidth: 1.0
```

```
boxplot.whiskerprops.linewidth: 1.0
boxplot.whiskers: 1.5
contour.corner_mask: True
contour.negative_linestyle: dashed
datapath: /opt/conda/lib/python3.6/site-packages/matplotlib/mpl-data
date.autoformatter.day: %Y-%m-%d
date.autoformatter.hour: %m-%d %H
date.autoformatter.microsecond: %M:%S.%f
date.autoformatter.minute: %d %H:%M
date.autoformatter.month: %Y-%m
date.autoformatter.second: %H:%M:%S
date.autoformatter.year: %Y
docstring.hardcopy: False
errorbar.capsize: 0.0
examples.directory:
figure.autolayout: False
figure.constrained_layout.h_pad: 0.04167
figure.constrained_layout.hspace: 0.02
figure.constrained_layout.use: False
figure.constrained_layout.w_pad: 0.04167
figure.constrained_layout.wspace: 0.02
figure.dpi: 72.0
figure.edgecolor: 0.50
figure.facecolor: white
figure.figsize: [6.0, 4.0]
figure.frameon: True
figure.max_open_warning: 20
figure.subplot.bottom: 0.125
figure.subplot.hspace: 0.2
figure.subplot.left: 0.125
figure.subplot.right: 0.9
figure.subplot.top: 0.88
figure.subplot.wspace: 0.2
figure.titlesize: large
figure.titleweight: normal
font.cursive: ['Apple Chancery', 'Textile', 'Zapf Chancery', 'Sand', 'Script MT', 'Felipa', 'cursive']
font.family: ['sans-serif']
font.fantasy: ['Comic Sans MS', 'Chicago', 'Charcoal', 'Impact', 'Western', 'Humor Sans', 'xkcd', 'fan
tasy']
font.monospace: ['DejaVu Sans Mono', 'Bitstream Vera Sans Mono', 'Computer Modern Typewriter', 'Andale
Mono', 'Nimbus Mono L', 'Courier New', 'Courier', 'Fixed', 'Terminal', 'monospace']
font.sans-serif: ['SimHei', 'DejaVu Sans', 'Bitstream Vera Sans', 'Computer Modern Sans Serif', 'Lucid
a Grande', 'Verdana', 'Geneva', 'Lucid', 'Arial', 'Helvetica', 'Avant Garde', 'sans-serif']
font.serif: ['DejaVu Serif', 'Bitstream Vera Serif', 'Computer Modern Roman', 'New Century Schoolbook'
, 'Century Schoolbook L', 'Utopia', 'ITC Bookman', 'Bookman', 'Nimbus Roman No9 L', 'Times New Roman',
'Times', 'Palatino', 'Charter', 'serif']
font.size: 10.0
font.stretch: normal
font.style: normal
font.variant: normal
font.weight: normal
grid.alpha: 1.0
grid.color: white
grid.linestyle: -
grid.linewidth: 0.8
hatch.color: black
hatch.linewidth: 1.0
hist.bins: 10
image.aspect: equal
image.cmap: viridis
image.composite_image: True
image.interpolation: nearest
image.lut: 256
image.origin: upper
image.resample: True
interactive: True
keymap.all_axes: ['a']
keymap.back: ['left', 'c', 'backspace', 'MouseButton.BACK']
keymap.copy: ['ctrl+c', 'cmd+c']
keymap.forward: ['right', 'v', 'MouseButton.FORWARD']
keymap.fullscreen: ['f', 'ctrl+f']
keymap.grid: ['g']
keymap.grid_minor: ['G']
keymap.help: ['f1']
keymap.home: ['h', 'r', 'home']
keymap.pan: ['p']
keymap.quit: ['ctrl+w', 'cmd+w', 'q']
keymap.quit_all: ['W', 'cmd+W', 'Q']
```

keymap.quit_all: ['w', 'cmd+w', 'Q']
keymap.save: ['s', 'ctrl+s']
keymap.xscale: ['k', 'L']
keymap.yscale: ['l']
keymap.zoom: ['o']
legend.borderaxespad: 0.5
legend.borderpad: 0.4
legend.columnspacing: 2.0
legend.edgecolor: 0.8
legend.facecolor: inherit
legend.fancybox: True
legend.fontsize: medium
legend.framealpha: 0.8
legend.frameon: True
legend.handleheight: 0.7
legend.handlelength: 2.0
legend.handletextpad: 0.8
legend.labelspacing: 0.5
legend.loc: best
legend.markerscale: 1.0
legend.numpoints: 1
legend.scatterpoints: 1
legend.shadow: False
legend.title_fontsize: None
lines.antialiased: True
lines.color: C0
lines.dash_capstyle: butt
lines.dash_joinstyle: round
lines.dashdot_pattern: [6.4, 1.6, 1.0, 1.6]
lines.dashed_pattern: [3.7, 1.6]
lines.dotted_pattern: [1.0, 1.65]
lines.linestyle: -
lines.linewidth: 1.5
lines.marker: None
lines.markeredgecolor: auto
lines.markeredgewidth: 1.0
lines.markerfacecolor: auto
lines.markersize: 6.0
lines.scale_dashes: True
lines.solid_capstyle: projecting
lines.solid_joinstyle: round
markers.fillstyle: full
mathtext.bf: sans:bold
mathtext.cal: cursive
mathtext.default: it
mathtext.fallback_to_cm: True
mathtext.fontset: dejavusans
mathtext.it: sans:italic
mathtext.rm: sans
mathtext.sf: sans
mathtext.tt: monospace
patch.antialiased: True
patch.edgecolor: #EEEEEE
patch.facecolor: #348ABD
patch.force_edgecolor: False
patch.linewidth: 0.5
path.effects: []
path.simplify: True
path.simplify_threshold: 0.1111111111111111
path.sketch: None
path.snap: True
pdf.compression: 6
pdf.fonttype: 3
pdf.inheritcolor: False
pdf.use14corefonts: False
pgf.preamble:
pgf.rcfonts: True
pgf.texsystem: xelatex
polaraxes.grid: True
ps.distiller.res: 6000
ps.fonttype: 3
ps.papersize: letter
ps.useafm: False
ps.usedistiller: False
savefig.bbox: None
savefig.directory: ~
savefig.dpi: figure
savefig.edgecolor: white

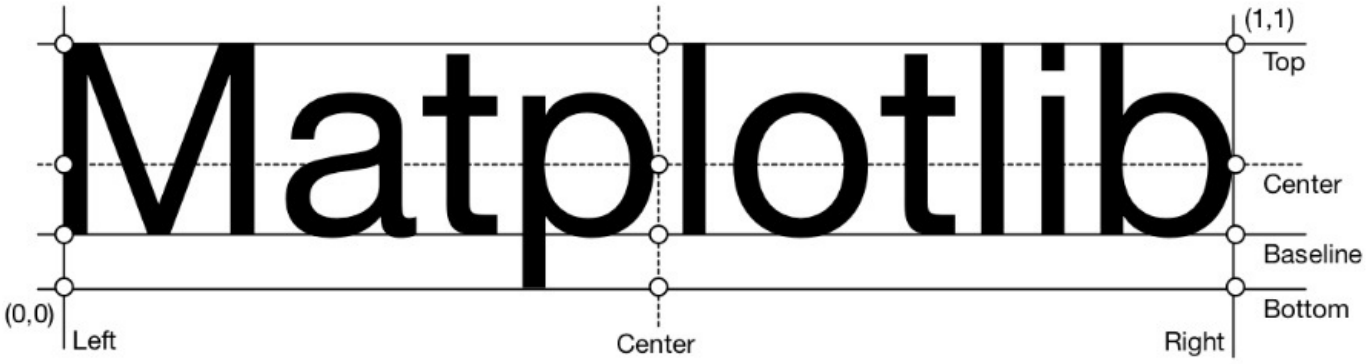
```
savefig.edgecolor: white
savefig.facecolor: white
savefig.format: png
savefig.frameon: True
savefig.jpeg_quality: 95
savefig.orientation: portrait
savefig.pad_inches: 0.1
savefig.transparent: False
scatter.edgecolors: face
scatter.marker: o
svg.fonttype: path
svg.hashsalt: None
svg.image_inline: True
text.antialiased: True
text.color: black
text.hinting: auto
text.hinting_factor: 8
text.latex.preamble:
text.latex.preview: False
text.latex.unicode: True
text.usetex: False
timezone: UTC
tk.window_focus: False
toolbar: toolbar2
verbose.fileo: sys.stdout
verbose.level: silent
webagg.address: 127.0.0.1
webagg.open_in_browser: True
webagg.port: 8988
webagg.port_retries: 50
xtick.alignment: center
xtick.bottom: True
xtick.color: #555555
xtick.direction: out
xtick.labelbottom: True
xtick.labelsize: medium
xtick.labeltop: False
xtick.major.bottom: True
xtick.major.pad: 3.5
xtick.major.size: 3.5
xtick.major.top: True
xtick.major.width: 0.8
xtick.minor.bottom: True
xtick.minor.pad: 3.4
xtick.minor.size: 2.0
xtick.minor.top: True
xtick.minor.visible: False
xtick.minor.width: 0.6
xtick.top: False
ytick.alignment: center_baseline
ytick.color: #555555
ytick.direction: out
ytick.labelleft: True
ytick.labelright: False
ytick.labelsize: medium
ytick.left: True
ytick.major.left: True
ytick.major.pad: 3.5
ytick.major.right: True
ytick.major.size: 3.5
ytick.major.width: 0.8
ytick.minor.left: True
ytick.minor.pad: 3.4
ytick.minor.right: True
ytick.minor.size: 2.0
ytick.minor.visible: False
ytick.minor.width: 0.6
ytick.right: False
```

In [5]:

```
plt.rcParams['font.sans-serif'] = ['SimHei']   # 设置字体
plt.rcParams['axes.unicode_minus']=False     # 显示中文和特殊符号
```

4.4 绘图宝典

- github 上有用的绘图工具：https://github.com/rougier/matplotlib-cheatsheet (https://github.com/rougier/matplotlib-cheatsheet)



In [23]:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator, MultipleLocator, FuncFormatter

np.random.seed(19680801)

X = np.linspace(0.5, 3.5, 100)
Y1 = 3+np.cos(X)
Y2 = 1+np.cos(1+X/0.75)/2
Y3 = np.random.uniform(Y1, Y2, len(X))

fig = plt.figure(figsize=(8, 8), dpi=120,facecolor='#f7f7f7')
ax = fig.add_subplot(1, 1, 1, aspect=1, facecolor='#e7dff3')


def minor_tick(x, pos):
    if not x % 1.0:
        return ""
    return "%.2f" % x

ax.xaxis.set_major_locator(MultipleLocator(1.000))
ax.xaxis.set_minor_locator(AutoMinorLocator(4))
ax.yaxis.set_major_locator(MultipleLocator(1.000))
ax.yaxis.set_minor_locator(AutoMinorLocator(4))
ax.xaxis.set_minor_formatter(FuncFormatter(minor_tick))

ax.set_xlim(0, 4)
ax.set_ylim(0, 4)

ax.tick_params(which='major', width=1.0)
ax.tick_params(which='major', length=10)
ax.tick_params(which='minor', width=1.0, labelsize=10)
ax.tick_params(which='minor', length=5, labelsize=10, labelcolor='0.25')

ax.grid(linestyle="--", linewidth=0.5, color='.25', zorder=-10)

ax.plot(X, Y1, c=(0.25, 0.25, 1.00), lw=2, label="Blue signal", zorder=10)
ax.plot(X, Y2, c=(1.00, 0.25, 0.25), lw=2, label="Red signal")
ax.plot(X, Y3, linewidth=0,
        marker='o', markerfacecolor='w', markeredgecolor='k')

ax.set_title("Anatomy of a figure", fontsize=20, verticalalignment='bottom')
ax.set_xlabel("X axis label")
ax.set_ylabel("Y axis label")

ax.legend()


def circle(x, y, radius=0.15):
    from matplotlib.patches import Circle
    from matplotlib.patheffects import withStroke
    circle = Circle((x, y), radius, clip_on=False, zorder=10, linewidth=1,
                    edgecolor='black', facecolor=(0, 0, 0, .0125),
                    path_effects=[withStroke(linewidth=5, foreground='w')])
```

```
    ax.add_artist(circle)


def text(x, y, text):
    ax.text(x, y, text, backgroundcolor="white",
            ha='center', va='top', weight='bold', color='blue')


# Minor tick
circle(0.50, -0.10)
text(0.50, -0.32, "Minor tick label 次刻度标签")

# Major tick
circle(-0.03, 4.00)
text(0.03, 3.80, "Major tick 主刻度")

# Minor tick
circle(0.00, 3.50)
text(0.00, 3.30, "Minor tick 次刻度")

# Major tick label
circle(-0.15, 3.00)
text(-0.15, 2.80, "Major tick label 主刻度标签")

# X Label
circle(1.80, -0.27)
text(1.80, -0.45, "X axis label X轴标签")

# Y Label
circle(-0.27, 1.80)
text(-0.27, 1.6, "Y axis label Y轴标签")

# Title
circle(1.60, 4.13)
text(1.60, 3.93, "Title 图标题")

# Blue plot
circle(1.75, 2.80)
text(1.75, 2.60, "Line\n(line plot) 线图")

# Red plot
circle(1.20, 0.60)
text(1.20, 0.40, "Line\n(line plot) 线图")

# Scatter plot
circle(3.20, 1.75)
text(3.20, 1.55, "Markers\n(scatter plot) 散点图")

# Grid
circle(3.00, 3.00)
text(3.00, 2.80, "Grid 网格")

# Legend
circle(3.70, 3.80)
text(3.70, 3.60, "Legend 图例")

# Axes
circle(0.5, 0.5)
text(0.5, 0.3, "Axes 坐标系")

# Figure
circle(-0.3, 0.65)
text(-0.3, 0.45, "Figure 图")

color = 'blue'
ax.annotate('Spines 轴脊线', xy=(4.0, 0.35), xytext=(3.3, 0.5),
            weight='bold', color=color,
            arrowprops=dict(arrowstyle='->',
                            connectionstyle="arc3",
                            color=color))

ax.annotate('', xy=(3.15, 0.0), xytext=(3.45, 0.45),
            weight='bold', color=color,
            arrowprops=dict(arrowstyle='->',
                            connectionstyle="arc3",
                            color=color))
```

```
ax.text(4.0, -0.4, "Made with http://matplotlib.org",
        fontsize=10, ha="right", color='.5')

plt.show()
```