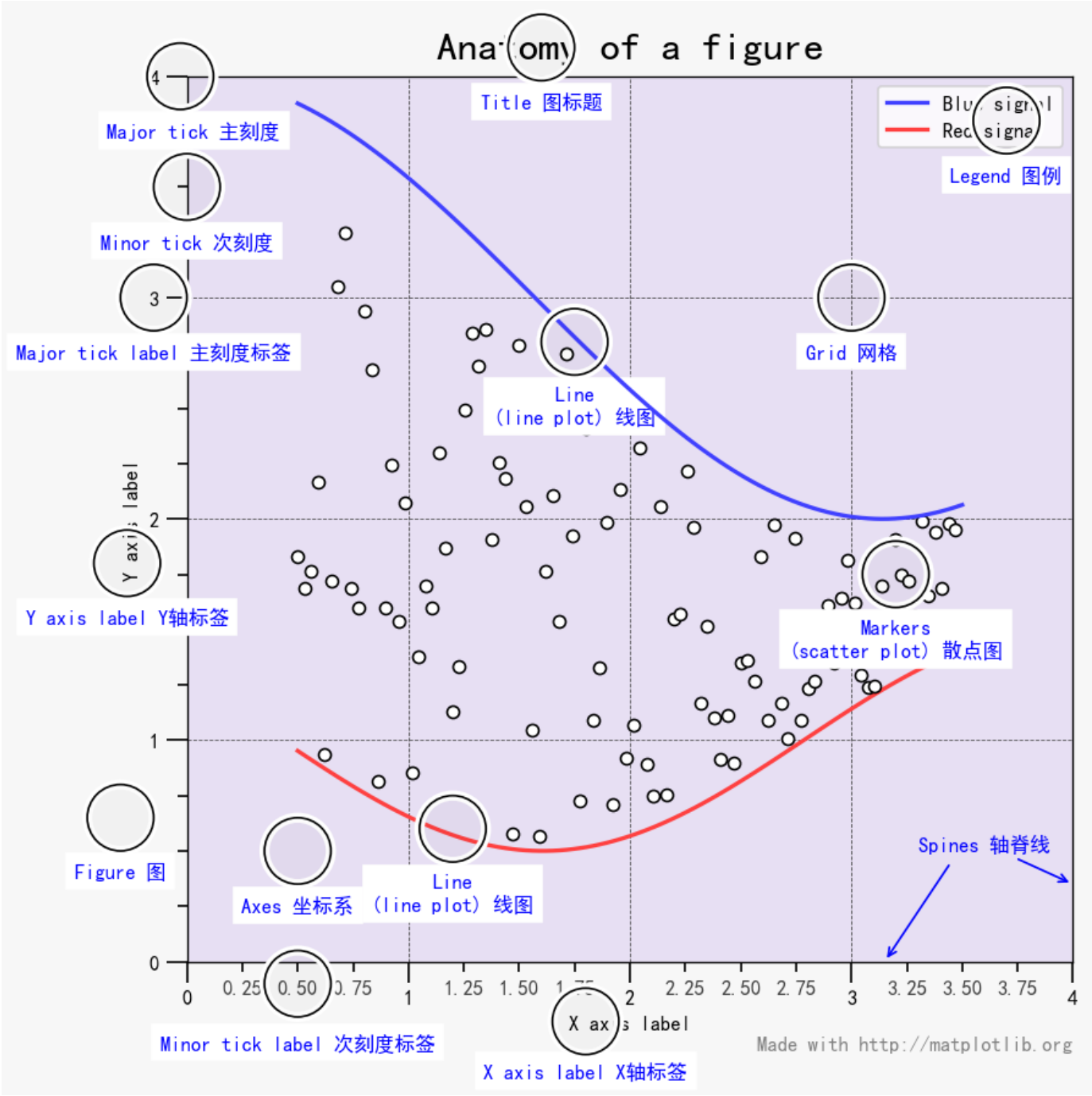
一、matplotlib 中一幅图的构成

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 [9]:

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

In [10]:

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

Out[10]:

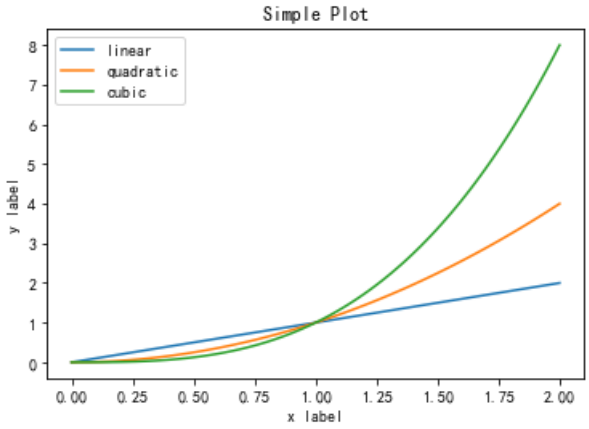
```
array([0.        , 0.02020202, 0.04040404, 0.06060606, 0.08080808,
       0.1010101 , 0.12121212, 0.14141414, 0.16161616, 0.18181818,
       0.2020202 , 0.22222222, 0.24242424, 0.26262626, 0.28282828,
       0.3030303 , 0.32323232, 0.34343434, 0.36363636, 0.38383838,
       0.4040404 , 0.42424242, 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 [11]:

```
# 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[11]:

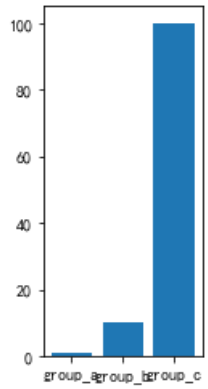


In [12]:

```
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)
```

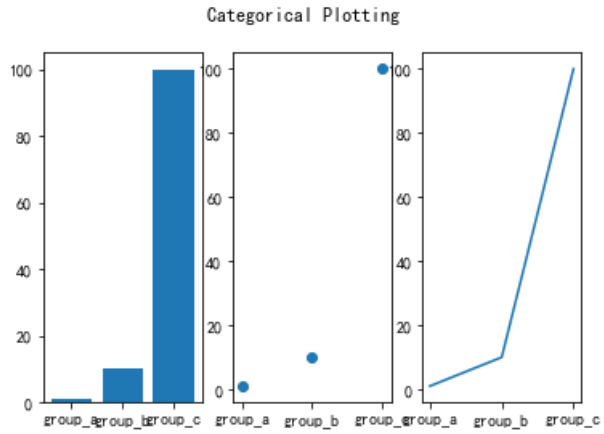
Out[12]:

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



In [13]:

```
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）

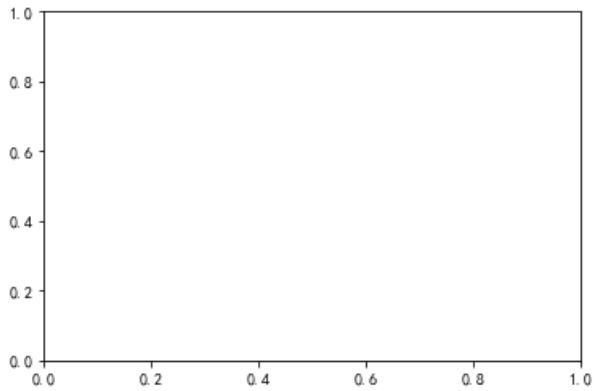
面向对象的绘图方式**首先要创建好图 `figure` 和坐标系 `axes`**，然后以坐标系为基础，**直接调用坐标系上的各种绘图方法绘制图形**。

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 [14]:

```
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
```



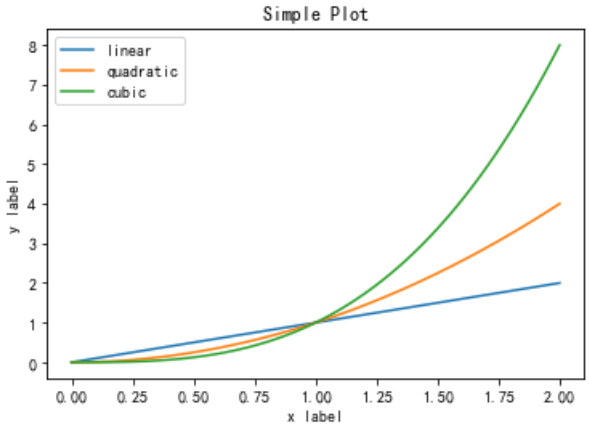
In [15]:

```
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)
```

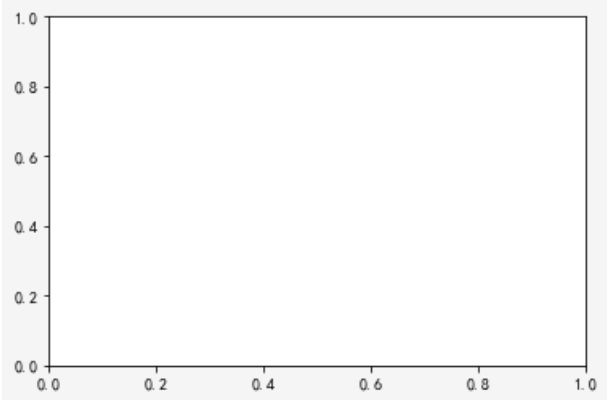
```
Figure(720x576)
```

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

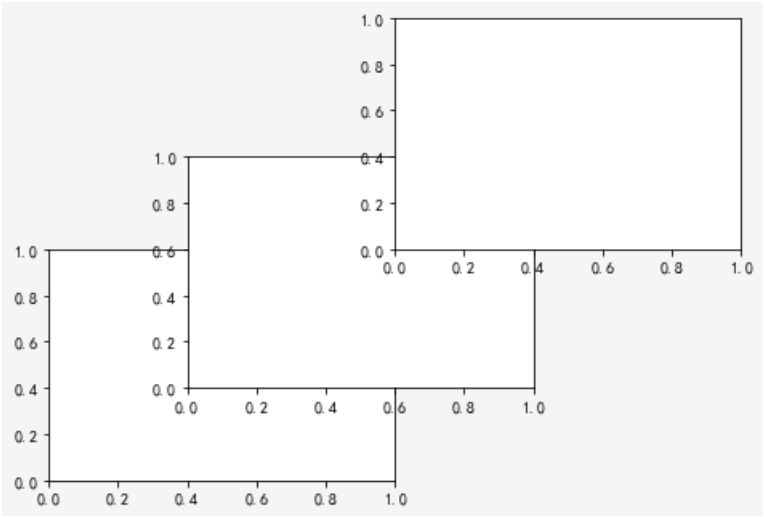
```
plt.axes()
plt.axes(rect=[left, bottom, width, height], projection=None, polar=False, **kwargs)

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]:



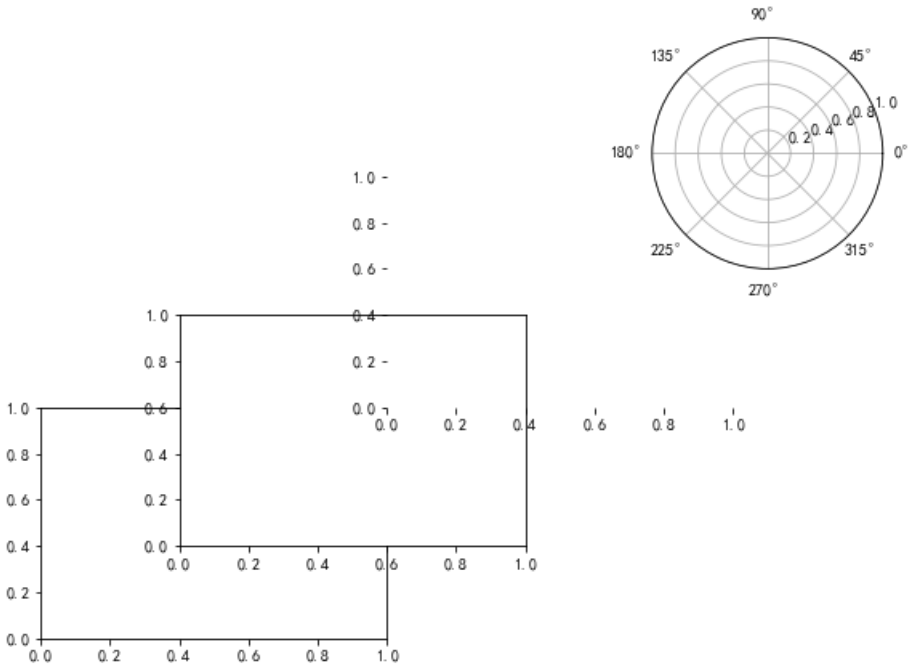
方式2（OO-style）：`Figure.add_axes()` 基于 figure 对象，在 figure 上添加坐标系

```
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)
```