

Matplotlib 是 Python 的2D绘图库（其实也是可以画3D图，甚至是动画图的）

plt.plot() 绘图

plt.plot(x, y, color, linestyle, linewidth, marker, markerfacecolor, markersize, label)

- x: x轴数据（列表或数组）
- y: y轴数据（列表或数组）
- color: 线的颜色
- linestyle: 线条样式
- linewidth: 线条宽度（用数字表示大小）
- marker: 标记的样式
- markerfacecolor: 标记填充颜色
- markersize: 标记尺寸（用数字表示大小）
- label: 线条的标签（后文结合 legend() 创建图例来讲）

以上就是plot()函数常用的参数，下面的表格是对应参数可供选择的值（只要记住常用的就行，其他的需要用到再来查就可以了）

颜色（颜色和十六进制颜色是对应的，都可以的）

颜色	十六进制 颜色	颜色	十六进制 颜色	颜色	十六进制 颜色
'aliceblue'	'#F0F8FF'	'antiquewhite'	'#FAEBD7'	'aqua'	'#00FFFF'
'aquamarine'	'#7FFFD4'	'azure'	'#F0FFFF'	'beige'	'#F5F5DC'
'bisque'	'#FFE4C4'	'black'	'#000000'	'blanchedalmond'	'#FFEBCD'
'blue'	'#0000FF'	'blueviolet'	'#8A2BE2'	'brown'	'#A52A2A'
'burlywood'	'#DEB887'	'cadetblue'	'#5F9EA0'	'chartreuse'	'#7FFF00'
'chocolate'	'#D2691E'	'coral'	'#FF7F50'	'cornflowerblue'	'#6495ED'

颜色	十六进制 颜色	颜色	十六进制 颜色	颜色	十六进制 颜色
'cornsilk'	'#FFF8DC'	'crimson'	'#DC143C'	'cyan'	'#00FFFF'
'darkblue'	'#00008B'	'darkcyan'	'#008B8B'	'darkgoldenrod'	'#B8860B'
'darkgray'	'#A9A9A9'	'darkgreen'	'#006400'	'darkkhaki'	'#BDB76B'
'darkmagenta'	'#8B008B'	'darkolivegreen'	'#556B2F'	'darkorange'	'#FF8C00'
'darkorchid'	'#9932CC'	'darkred'	'#8B0000'	'darksalmon'	'#E9967A'
'darkseagreen'	'#8FBC8F'	'darkslateblue'	'#483D8B'	'darkslategray'	'#2F4F4F'
'darkturquoise'	'#00CED1'	'darkviolet'	'#9400D3'	'deeppink'	'#FF1493'
'deepskyblue'	'#00BFFF'	'dimgray'	'#696969'	'dodgerblue'	'#1E90FF'
'firebrick'	'#B22222'	'floralwhite'	'#FFFAF0'	'forestgreen'	'#228B22'
'fuchsia'	'#FF00FF'	'gainsboro'	'#DCDCDC'	'ghostwhite'	'#F8F8FF'
'gold'	'#FFD700'	'goldenrod'	'#DAA520'	'gray'	'#808080'
'green'	'#008000'	'greenyellow'	'#ADFF2F'	'honeydew'	'#F0FFF0'
'hotpink'	'#FF69B4'	'indianred'	'#CD5C5C'	'indigo'	'#4B0082'
'ivory'	'#FFFFFF0'	'khaki'	'#F0E68C'	'lavender'	'#E6E6FA'
'lavenderblush'	'#FFF0F5'	'lawngreen'	'#7CFC00'	'lemonchiffon'	'#FFFACD'
'lightblue'	'#ADD8E6'	'lightcoral'	'#F08080'	'lightcyan'	'#E0FFFF'
'lightgoldenrodyellow'	'#FAFAD2'	'lightgreen'	'#90EE90'	'lightgray'	'#D3D3D3'
'lightpink'	'#FFB6C1'	'lightsalmon'	'#FFA07A'	'lightseagreen'	'#20B2AA'
'lightskyblue'	'#87CEFA'	'lightslategray'	'#778899'	'lightsteelblue'	'#B0C4DE'
'lightyellow'	'#FFFFE0'	'lime'	'#00FF00'	'limegreen'	'#32CD32'
'linen'	'#FAF0E6'	'magenta'	'#FF00FF'	'maroon'	'#800000'
'mediumaquamarine'	'#66CDAA'	'mediumblue'	'#0000CD'	'mediumorchid'	'#BA55D3'
'mediumpurple'	'#9370DB'	'mediumseagreen'	'#3CB371'	'mediumslateblue'	'#7B68EE'
'mediumspringgreen'	'#00FA9A'	'mediumturquoise'	'#48D1CC'	'mediumvioletred'	'#C71585'
'midnightblue'	'#191970'	'mintcream'	'#F5FFFA'	'mistyrose'	'#FFE4E1'
'moccasin'	'#FFE4B5'	'navajowhite'	'#FFDEAD'	'navy'	'#000080'
'oldlace'	'#FDF5E6'	'olive'	'#808000'	'olivedrab'	'#6B8E23'
'orange'	'#FFA500'	'orangered'	'#FF4500'	'orchid'	'#DA70D6'
'palegoldenrod'	'#EEE8AA'	'palegreen'	'#98FB98'	'paleturquoise'	'#AFEEEE'

颜色	十六进制 颜色	颜色	十六进制 颜色	颜色	十六进制 颜色
'palevioletred'	'#DB7093'	'papayawhip'	'#FFBED5'	'peachpuff'	'#FFDAB9'
'peru'	'#CD853F'	'pink'	'#FFC0CB'	'plum'	'#DDA0DD'
'powderblue'	'#B0E0E6'	'purple'	'#800080'	'red'	'#FF0000'
'rosybrown'	'#BC8F8F'	'royalblue'	'#4169E1'	'saddlebrown'	'#8B4513'
'salmon'	'#FA8072'	'sandybrown'	'#FAA460'	'seagreen'	'#2E8B57'
'seashell'	'#FFF5EE'	'sienna'	'#A0522D'	'silver'	'#C0C0C0'
'skyblue'	'#87CEEB'	'slateblue'	'#6A5ACD'	'slategray'	'#708090'
'snow'	'#FFFAFA'	'springgreen'	'#00FF7F'	'steelblue'	'#4682B4'
'tan'	'#D2B48C'	'teal'	'#008080'	'thistle'	'#D8BFD8'
'tomato'	'#FF6347'	'turquoise'	'#40E0D0'	'violet'	'#EE82EE'
'wheat'	'#F5DEB3'	'white'	'#FFFFFF'	'whitesmoke'	'#F5F5F5'
'yellow'	'#FFFF00'	'yellowgreen'	'#9ACD32'		

linestyle 线条样式（样式一和样式二是对应的，都可以的）

样式一	样式二	说明
'-'	'solid'	实线
'--'	'dashed'	虚线
'-.'	'dashdot'	点画线
':'	'dotted'	点线
''	'None'	不显示线条了

marker 标记样式

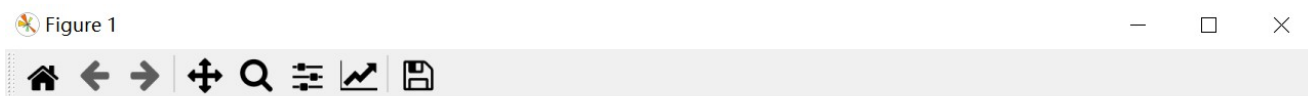
样式	说明	样式	说明
'.'	点标记	'1'	下花三角标记
','	像素标记（极小点）	'2'	上花三角标记
'o'	实心圆标记	'3'	左花三角标记
'v'	倒三角标记	'4'	右花三角标记

样式	说明	样式	说明
'^'	上三角标记	's'	实心方形标记
'>'	右三角标记	'p'	实心五角标记
'<'	左三角标记	'*'	星形标记
'h'	竖六边形标记	'd'	瘦菱形标记
'H'	横六边形标记	'D'	菱形标记
'+'	十字标记	' '	垂直线标记
'x'	x标记	'_'	水平线标记

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

x = np.linspace(-3, 3, 50)
y = np.sin(x)

plt.plot(x, y, color='skyblue', linestyle='-.',
         linewidth=2,
         marker='h', markerfacecolor='gold', markersize=8)
plt.show()
```



plt.figure() 创建画布

想要绘图，必须先要创建一个 **figure**（画布），还要有 **axes**（坐标系）

隐式创建 **figure**

- 当第一次执行 `plt.xxx()` 画图代码时，会自动判断是否已经存在画布
- 如果没有，则自动创建一个画布，并且在该画布上自动创建一个坐标系
- 如果不设置画布，一个画布上只有一个坐标系

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

```
x = np.linspace(-3, 3, 50)
```

```
y1 = 2*x + 2
```

```
y2 = x**2
```

```
y3 = np.sin(x)
```

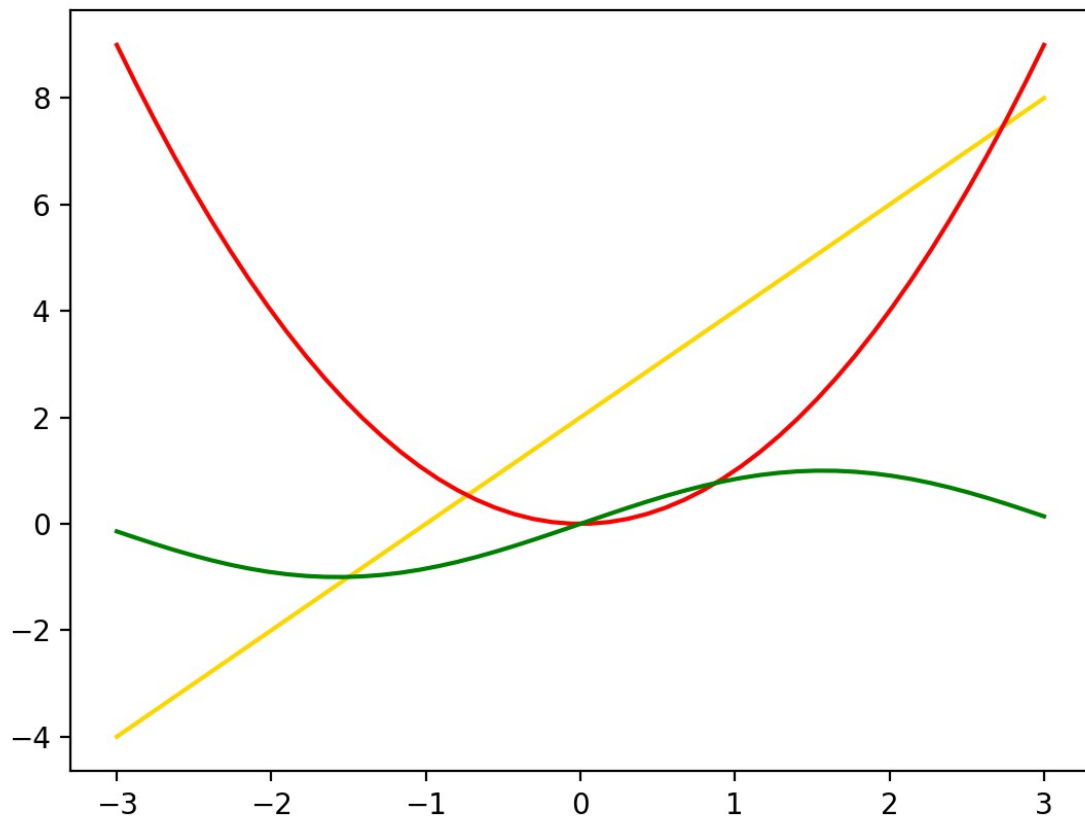
```
plt.plot(x, y1, color='gold')
```

```
plt.plot(x, y2, color='red')
```

```
plt.plot(x, y3, color='green')
```

```
plt.show()
```

Figure 1



显式创建 figure

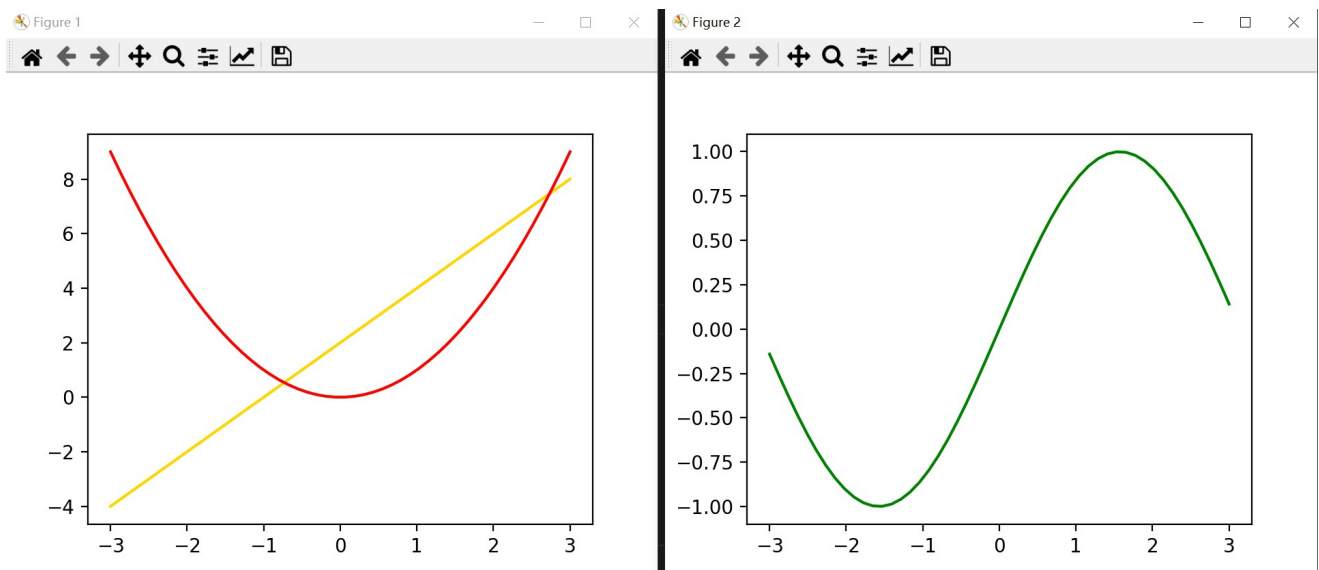
- 利用 `plt.figure()` 手动创建画布，可以创建多个画布
- 在 `plt.figure()` 下面的 `plt.xxx()` 画图代码都会画在对应的画布上面

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

x = np.linspace(-3, 3, 50)
y1 = 2*x + 2
y2 = x**2
y3 = np.sin(x)

plt.figure() # 下面两个画图操作都属于这个画布
plt.plot(x, y1, color='gold')
plt.plot(x, y2, color='red')

plt.figure() # 下面一个画图操作属于这个画布
plt.plot(x, y3, color='green')
plt.show()
```



`plt.figure(num=None, figsize=None, dpi=None, facecolor=None, edgecolor=None, frameon=True)`

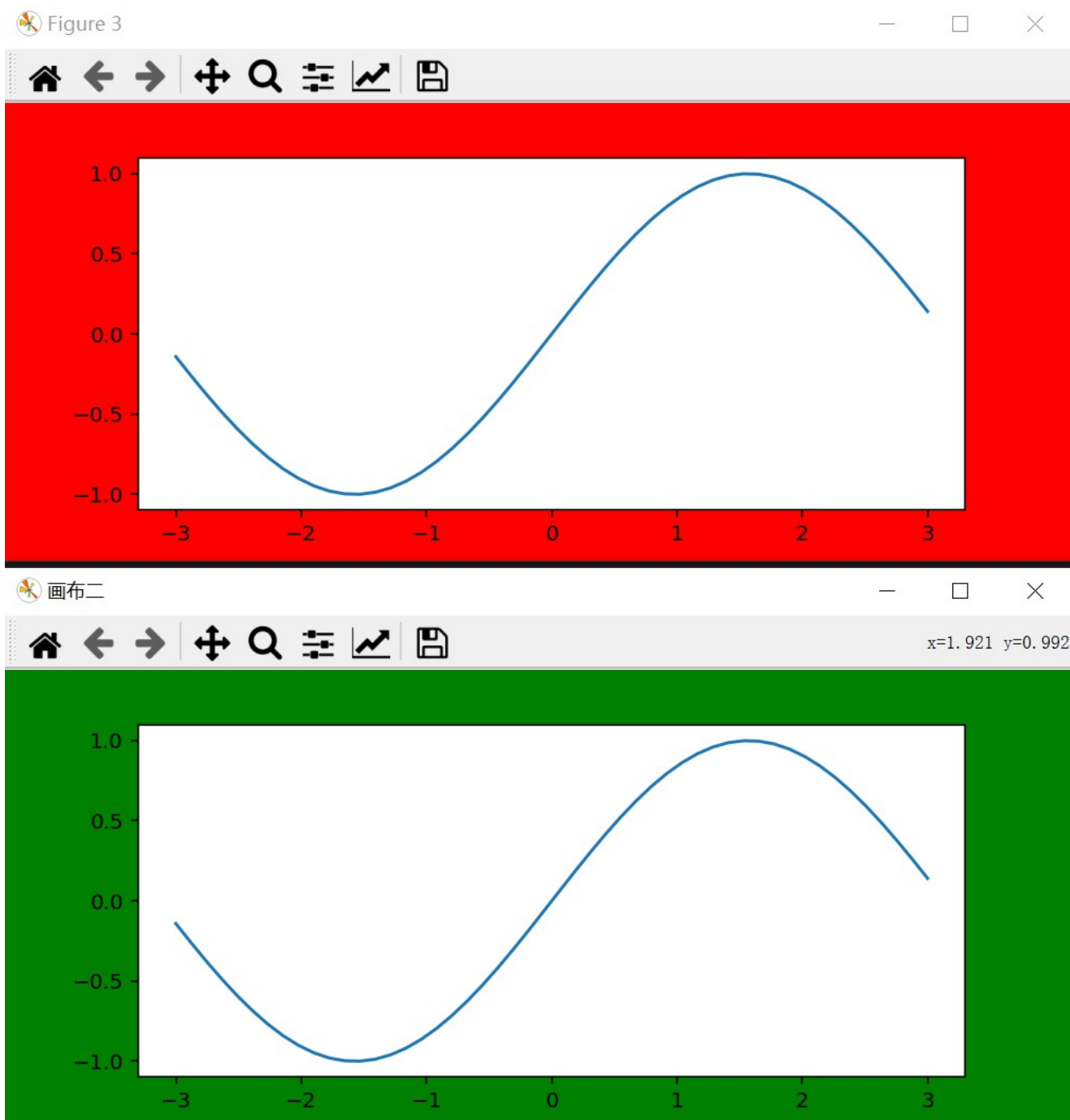
- **num:** 画布编号或名称（数字为编号，字符串为名称），不指定则按照数字顺序
- **figsize:** 指定 figure 的宽和高（英寸），默认为（6.4, 4.8）
- **dpi:** 指定画布显示的分辨率（用数字表示大小，默认为100）
- **facecolor:** 背景的颜色（除了画图区域的颜色）
- **edgecolor:** 边框线的颜色（看起来不是特别明显）
- **frameon:** 是否显示边框（不显示边框，则背景颜色也显示不了）

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

x = np.linspace(-3, 3, 50)
y = np.sin(x)

plt.figure(num=3, figsize=(7, 3), dpi=72, facecolor="red",
edgecolor="green", frameon=True)
plt.plot(x, y)

plt.figure(num="画布二", figsize=(7, 3), dpi=72,
facecolor="green", edgecolor="red", frameon=True)
plt.plot(x, y)
plt.show()
```

中文/负号显示问题

- matplotlib 在使用过程中，可能会遇到中文或者负号显示乱码的问题，把下面代码粘贴到matplotlib 使用的最前面即可

```
import matplotlib.pyplot as plt

# 通过rc参数修改字体为黑体，就可以显示中文
plt.rcParams['font.sans-serif'] = ['SimHei']
# 通过rc参数修改字符显示，就可以正常显示负号
plt.rcParams['axes.unicode_minus'] = False
```

设置坐标轴

限制显示区域

- 可以用 `plt.xlim()` 和 `plt.ylim()` 来限制图像横纵坐标轴的显示区域，既可以限制轴的最小值或最大值，又可以同时限制最小值和最大值
- 注意：只是限制显示区域，在窗口还是可以通过拖动功能来查看其他位置

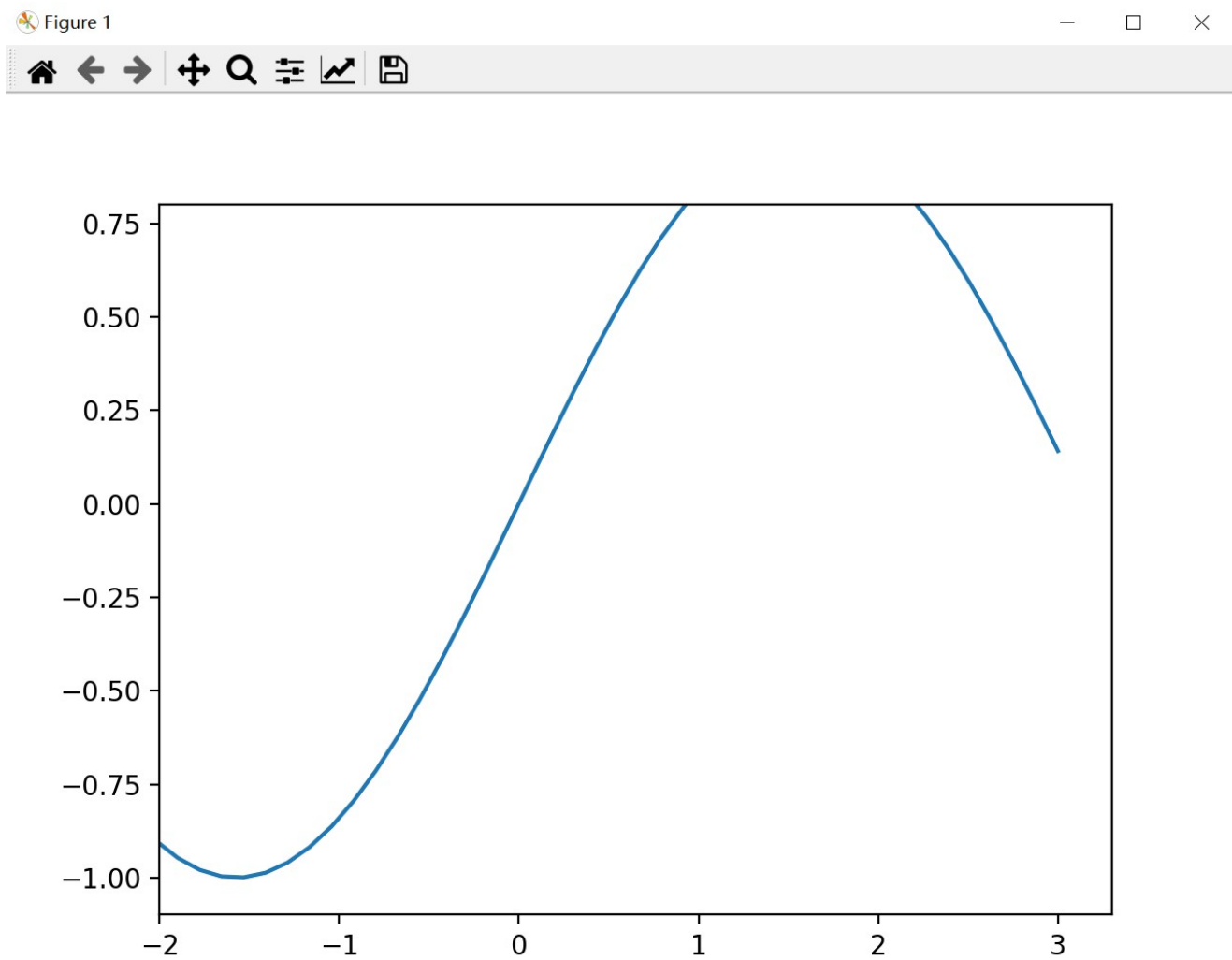
```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(-3, 3, 50)
y = np.sin(x)

plt.figure() # 创建画布1
plt.plot(x, y)
plt.xlim(xmin=-2) # 限制横坐标显示最小值为 -2
plt.ylim(ymax=0.8) # 限制纵坐标显示最大值为 0.8

plt.figure() # 创建画布2
plt.plot(x, y)
plt.xlim((-2, 4)) # 限制横坐标显示范围[-2, 4]
```

```
plt.ylim(ymin=-0.5, ymax=0.8) # 限制纵坐标显示范围[-0.5, 0.8]  
plt.show()
```





设置坐标轴标签

- 可以用 `plt.xlabel()` 和 `plt.ylabel()` 来指定图像横纵坐标轴的标签

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

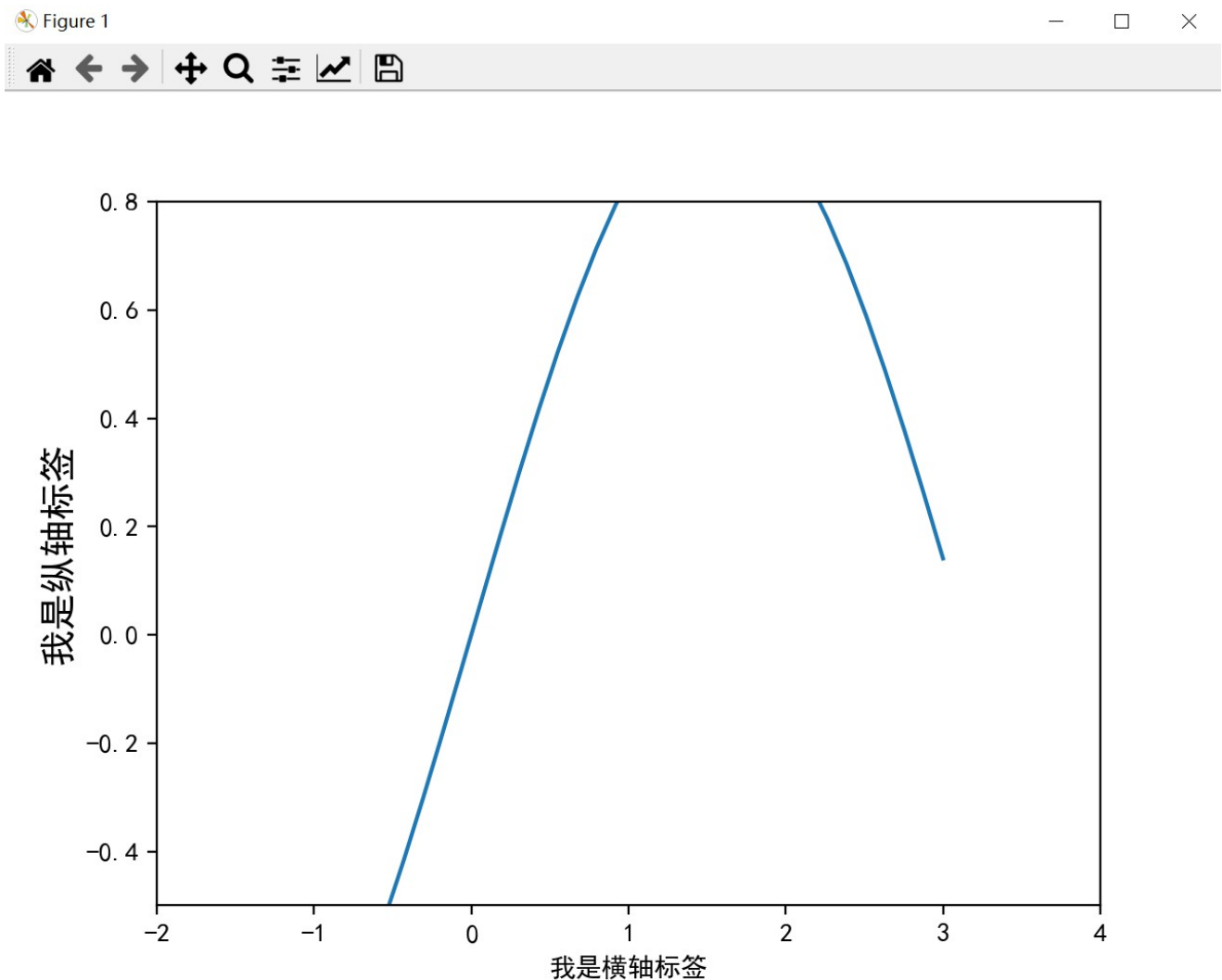
# 规避中文或者负号可能显示乱码的问题
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

x = np.linspace(-3, 3, 50)
y = np.sin(x)

plt.figure()
```

```
plt.plot(x, y)
plt.xlim((-2, 4))
plt.ylim((-0.5, 0.8))

# 指定横轴标签
plt.xlabel("我是横轴标签")
# 指定纵轴标签，并设置字体大小为14
plt.ylabel("我是纵轴标签", fontsize=14)
plt.show()
```



设置坐标轴刻度

`plt.xticks(ticks=None, labels=None) / plt.yticks(ticks=None, labels=None)`

- **ticks:** 刻度点的位置组成的列表（可以指定为空列表，则去掉刻度，但轴还在）
- **labels:** 刻度点的位置上的标签组成的列表（`labels`不指定，则标签显示 `ticks`值）
- 当不指定参数时，返回对应轴当前刻度点的位置和标签
- 指定参数时，则为设置对应轴的刻度点的位置和标签

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

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

x = np.linspace(-3, 3, 50)
y = np.sin(x)

plt.figure()
plt.plot(x, y)
plt.xlim(-2, 4)
plt.ylim((-0.4, 0.8))
plt.xlabel("我是横轴标签", fontsize=14)
plt.ylabel("我是纵轴标签", fontsize=14)

# 当不指定参数时，返回对应轴当前刻度点的位置和标签
print(plt.xticks())
print(plt.yticks())
plt.show()
```

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

```
x = np.linspace(-3, 3, 50)
```

```
y = np.sin(x)
```

```
plt.figure() # 画布1
```

```
plt.plot(x, y)
```

```
# ticks的值作为刻度点的位置, labels的值作为刻度点的位置上的标签
```

```
plt.yticks(ticks=[-1, -0.8, -0.5, -0.1, 1], labels=["a",  
"b", "c", "d", "e"])
```

```
plt.figure() # 画布2
```

```
plt.plot(x, y)
```

```
# ticks指定为空列表, 去掉刻度, 但轴还在
```

```
plt.xticks(ticks=[])
```

```
plt.figure() # 画布3
```

```
plt.plot(x, y)
```

```
plt.axis("off") # 把轴去掉, 刻度一起没了
```

```
plt.figure() # 画布4
```

```
plt.plot(x, y)
```

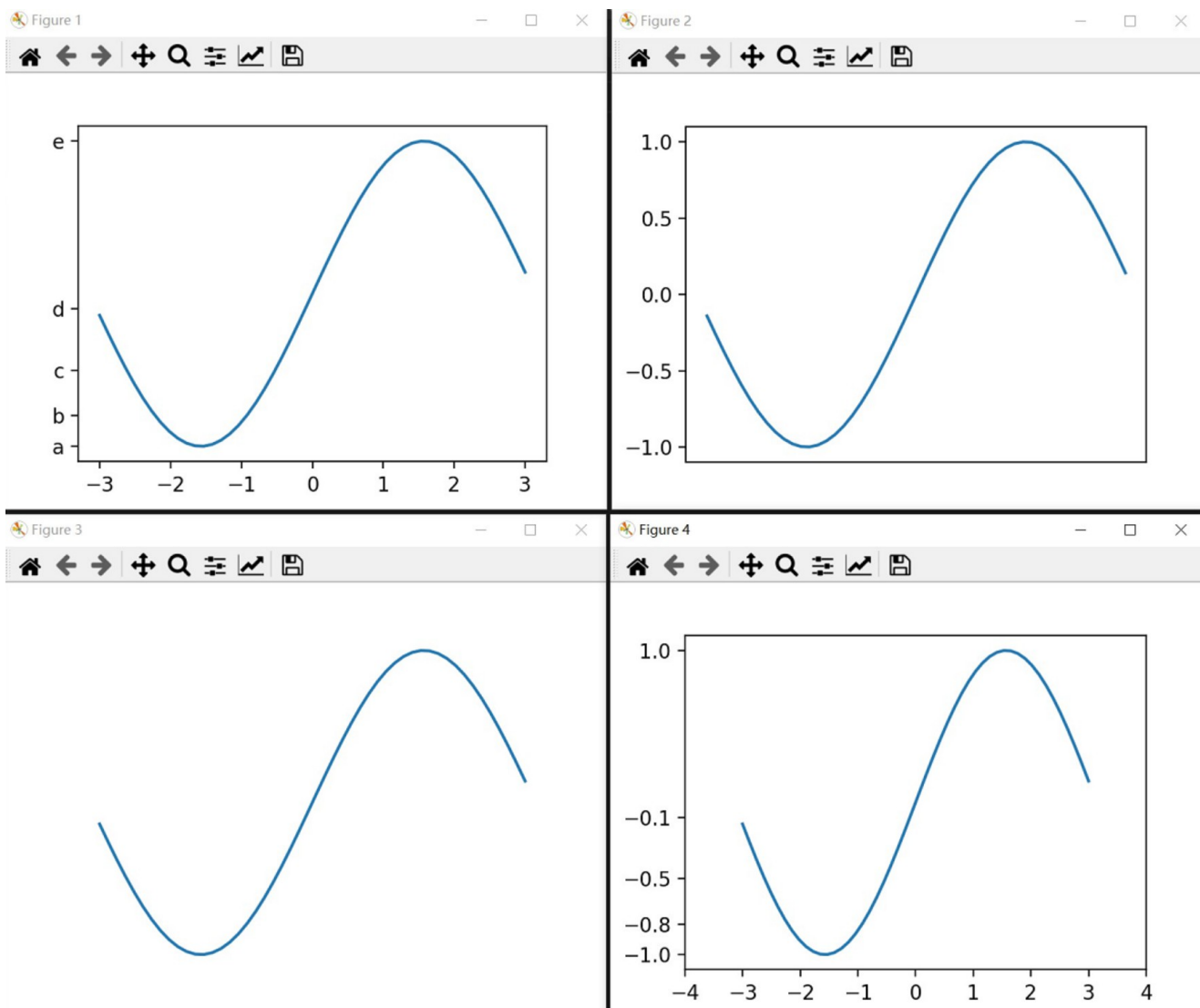
```
new_xticks = np.linspace(-4, 4, 9)
```

```
# 因为没有指定labels参数, 所以ticks的值既作为刻度点的位置又作为刻  
度的标签
```

```
plt.xticks(ticks=new_xticks)
```

```
plt.yticks(ticks=[-1, -0.8, -0.5, -0.1, 1])
```

```
plt.show()
```



设置坐标边框颜色

1. 用 `plt.gca()` 获取到坐标体系（矩形坐标框）
2. 调用坐标体系的 `spines` 属性，通过 `'top'`、`'bottom'`、`'left'`、`'right'` 参数获得指定的边框
3. 结合 `set_color()` 方法来指定颜色（当颜色指定为 `'None'` 时，表示不显示该边框）

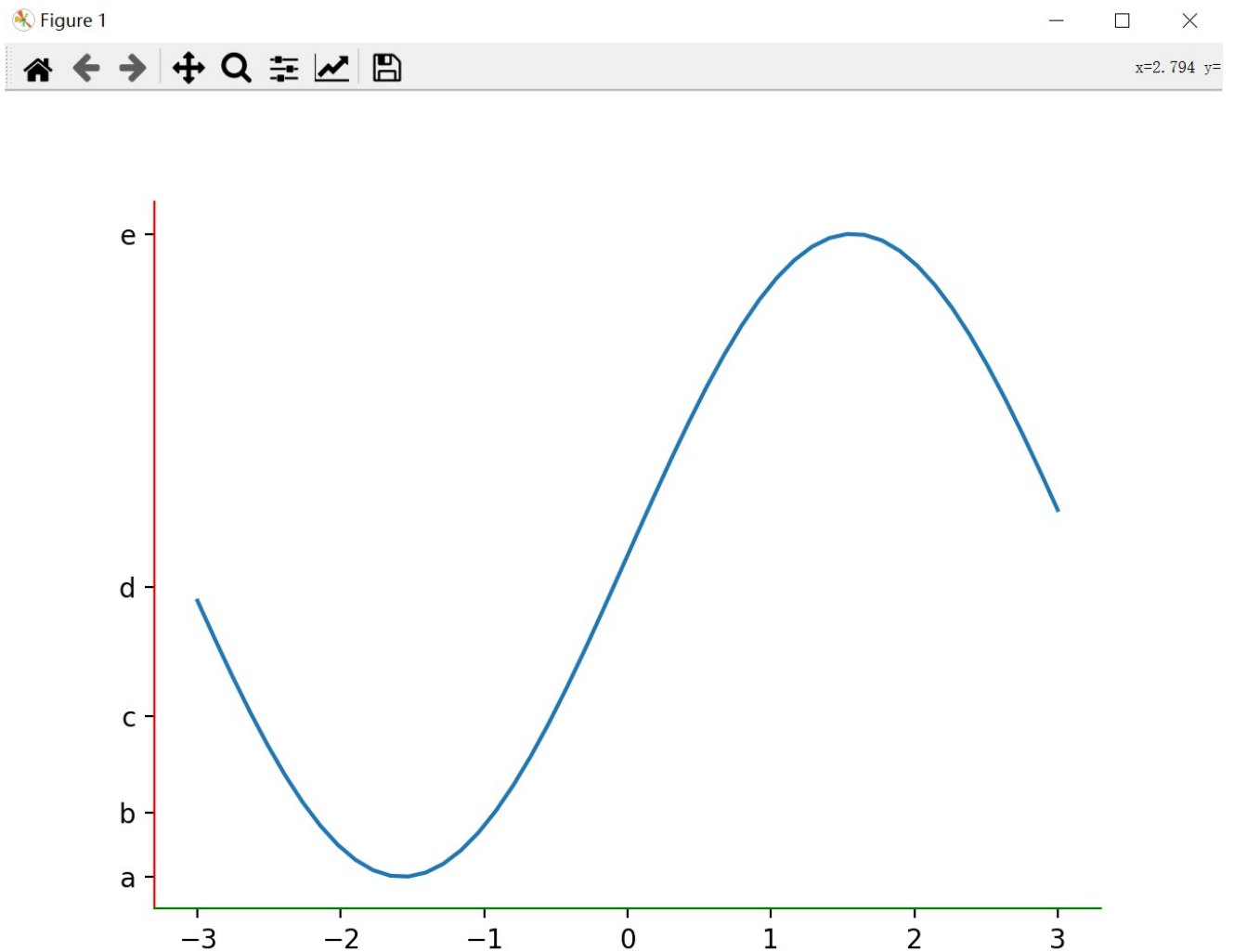
```
import matplotlib.pyplot as plt
import numpy as np
```

```
x = np.linspace(-3, 3, 50)
y = np.sin(x)
```



```
plt.figure()
plt.plot(x, y)
plt.yticks(ticks=[-1, -0.8, -0.5, -0.1, 1], labels=["a",
"b", "c", "d", "e"])

ax = plt.gca()
ax.spines['right'].set_color('None')
ax.spines['top'].set_color('None')
ax.spines['left'].set_color('red')
ax.spines['bottom'].set_color('green')
plt.show()
```



指定边框为坐标轴

1. 用 `plt.gca()` 获取到坐标体系（矩形坐标框）
2. 坐标体系指定轴来调用 `set_ticks_position()` 方法，通过 'top'、'bottom'、'left'、'right' 参数指定边框为坐标轴

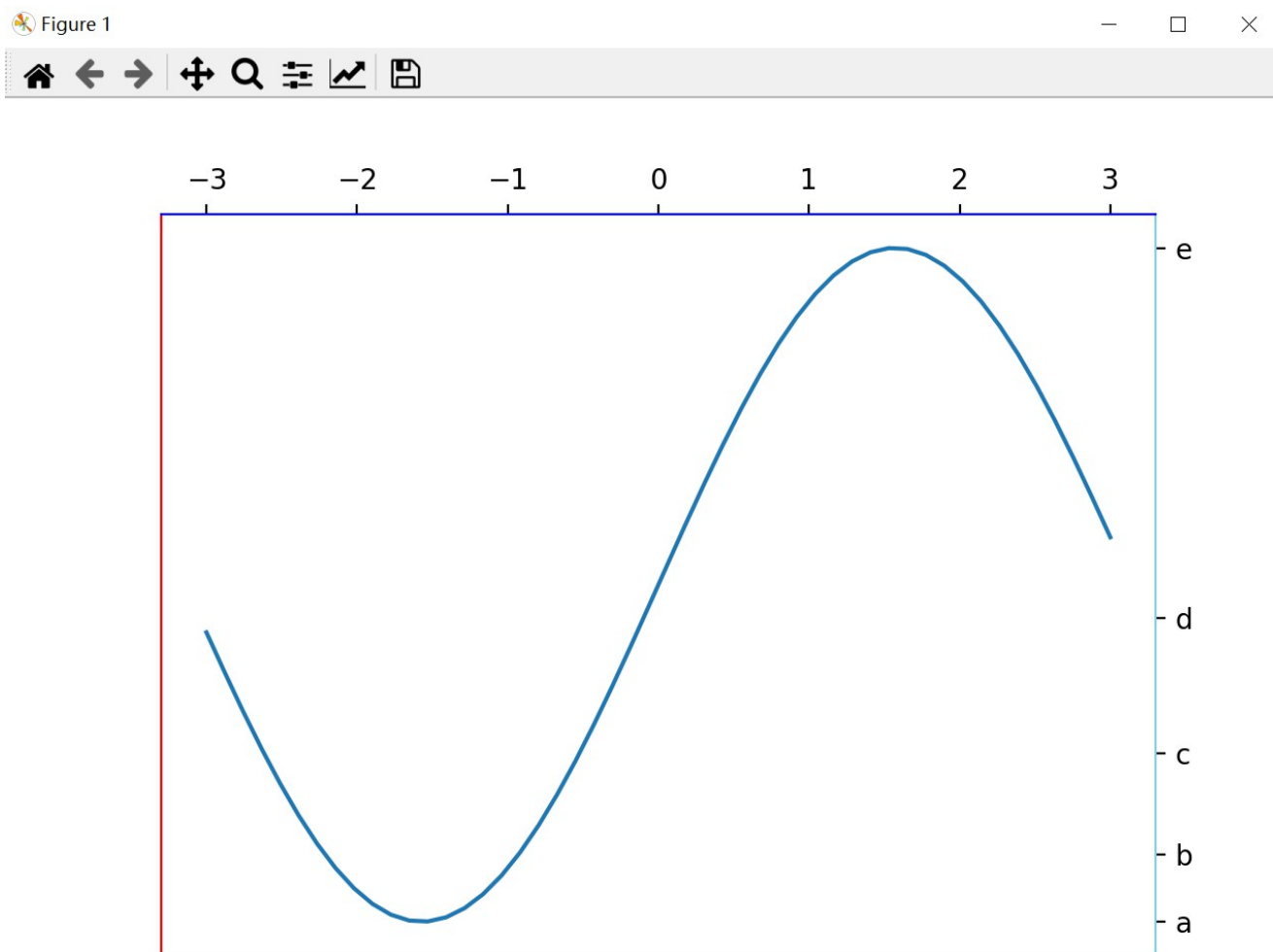
```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(-3, 3, 50)
y = np.sin(x)

plt.figure()
plt.plot(x, y)
plt.yticks(ticks=[-1, -0.8, -0.5, -0.1, 1], labels=['a',
'b', 'c', 'd', 'e'])

ax = plt.gca()
ax.spines['right'].set_color('skyblue')
ax.spines['top'].set_color('blue')
ax.spines['left'].set_color('red')
ax.spines['bottom'].set_color('green')

# 把坐标体系的上边框作为x轴
ax.xaxis.set_ticks_position('top')
# 把坐标体系的右边框作为y轴
ax.yaxis.set_ticks_position('right')
plt.show()
```



移动坐标边框

1. 用 `plt.gca()` 获取到坐标系（矩形坐标框）
2. 调用坐标体系的 `spines` 属性，通过 'top'、'bottom'、'left'、'right' 参数获得指定的边框
3. 结合 `set_position()` 方法来指定边框位置

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

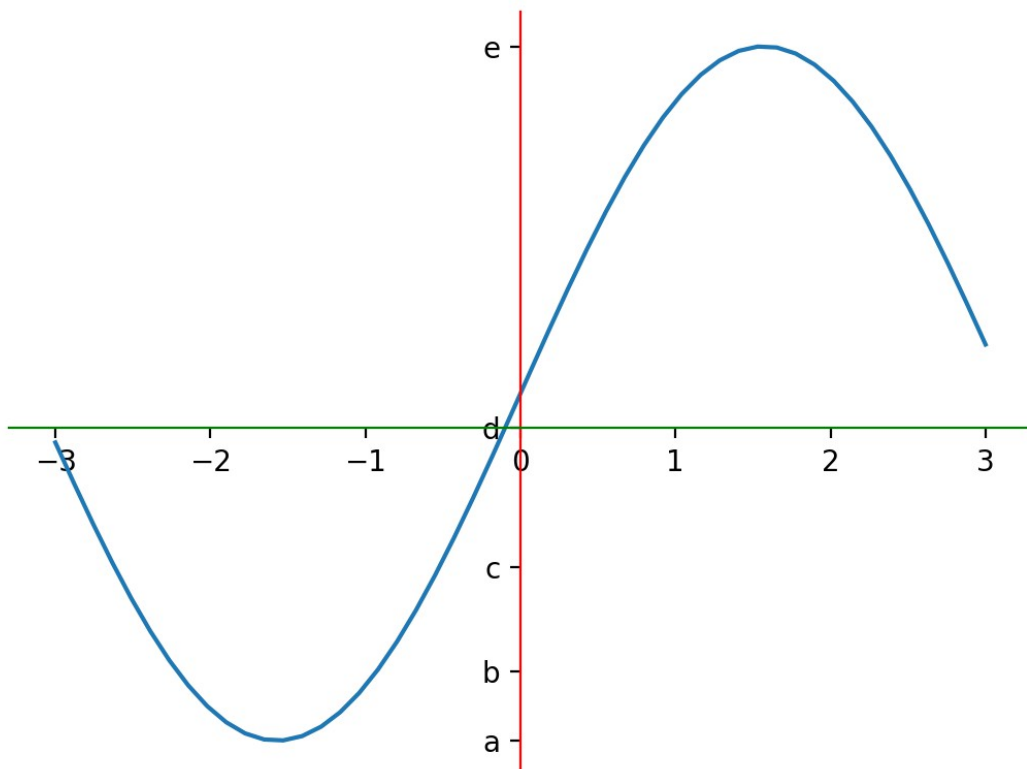
```
x = np.linspace(-3, 3, 50)
y = np.sin(x)
```

```
plt.figure()
plt.plot(x, y)
plt.yticks(ticks=[-1, -0.8, -0.5, -0.1, 1], labels=['a',
'b', 'c', 'd', 'e'])

ax = plt.gca()
ax.spines['right'].set_color('None')
ax.spines['top'].set_color('None')
ax.spines['left'].set_color('red')
ax.spines['bottom'].set_color('green')

# 选择坐标体系的左边框，设置位置到数据为0的地方(即x轴原点)
ax.spines['left'].set_position(('data', 0))
# 选择坐标体系的底边框，设置位置到数据为-0.1的地方(即y轴的'd'点)
ax.spines['bottom'].set_position(('data', -0.1))
plt.show()
```

Figure 1



plt.legend() 创建图例

plt.legend(handles, labels, loc, fontsize, frameon, edgecolor, facecolor)

- handles: 控制柄，默认是一个画布中所有线对象组成的列表
- labels: 图例标签，默认是绘图函数中 label 参数值组成的列表
- loc: 图例创建的位置（默认是loc="best"，代表自动找最好的位置）
- fontsize: 图例字体大小（用数字大小表示）
- frameon: 是否打开图例边框，默认为 True
- edgecolor: 图例边框颜色
- facecolor: 图例背景颜色（若未打开边框，参数无效）

LOC 参数可选值:

"best"	"upper right"	"upper left"	"lower right"
"lower left"	"right"	"center left"	
"lower center"	"upper center"	"center"	

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

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

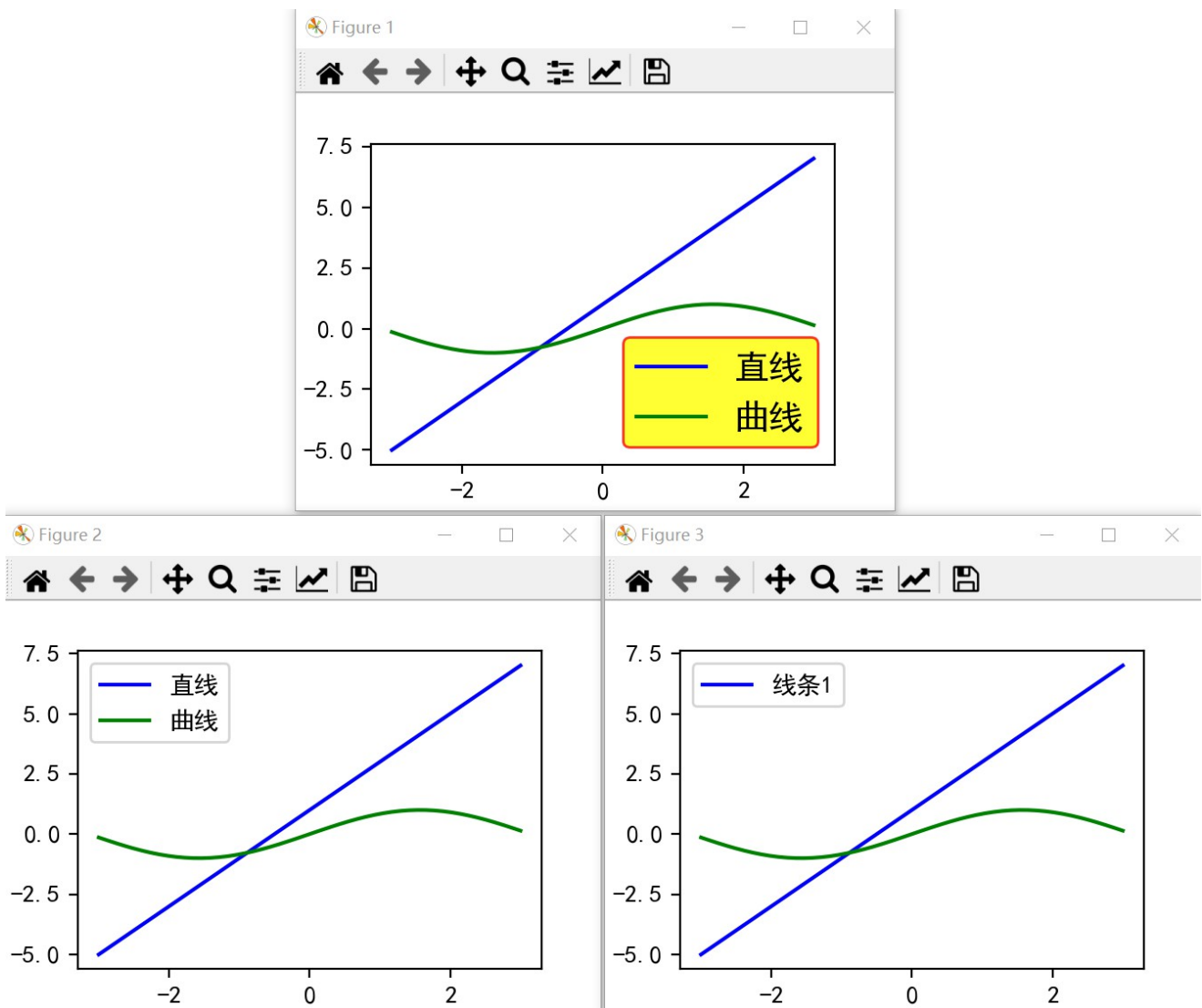
x = np.linspace(-3, 3, 50)
y1 = 2*x + 1
y2 = np.sin(x)

plt.figure()
plt.plot(x, y1, color='blue', label='直线')
plt.plot(x, y2, color='green', label='曲线')
```

```
plt.legend(loc='lower right', fontsize=14, frameon=True,  
edgecolor='red', facecolor='yellow')
```

```
plt.figure()  
plt.plot(x, y1, color='blue')  
plt.plot(x, y2, color='green')  
# 指定labels  
plt.legend(labels=['直线', '曲线'])
```

```
plt.figure()  
# plot返回Line2D对象，装在列表里，所以解包  
line1, = plt.plot(x, y1, color='blue', label='直线')  
line2, = plt.plot(x, y2, color='green', label='曲线')  
print(line1)  
print(line2)  
# 指定第一条线创建图例，且图例标签改为'线条1'  
plt.legend(handles=[line1, ], labels=['线条1', ])  
plt.show()
```



plt.text() 文字说明

`plt.text(x, y, s, size, color, ha, va)`

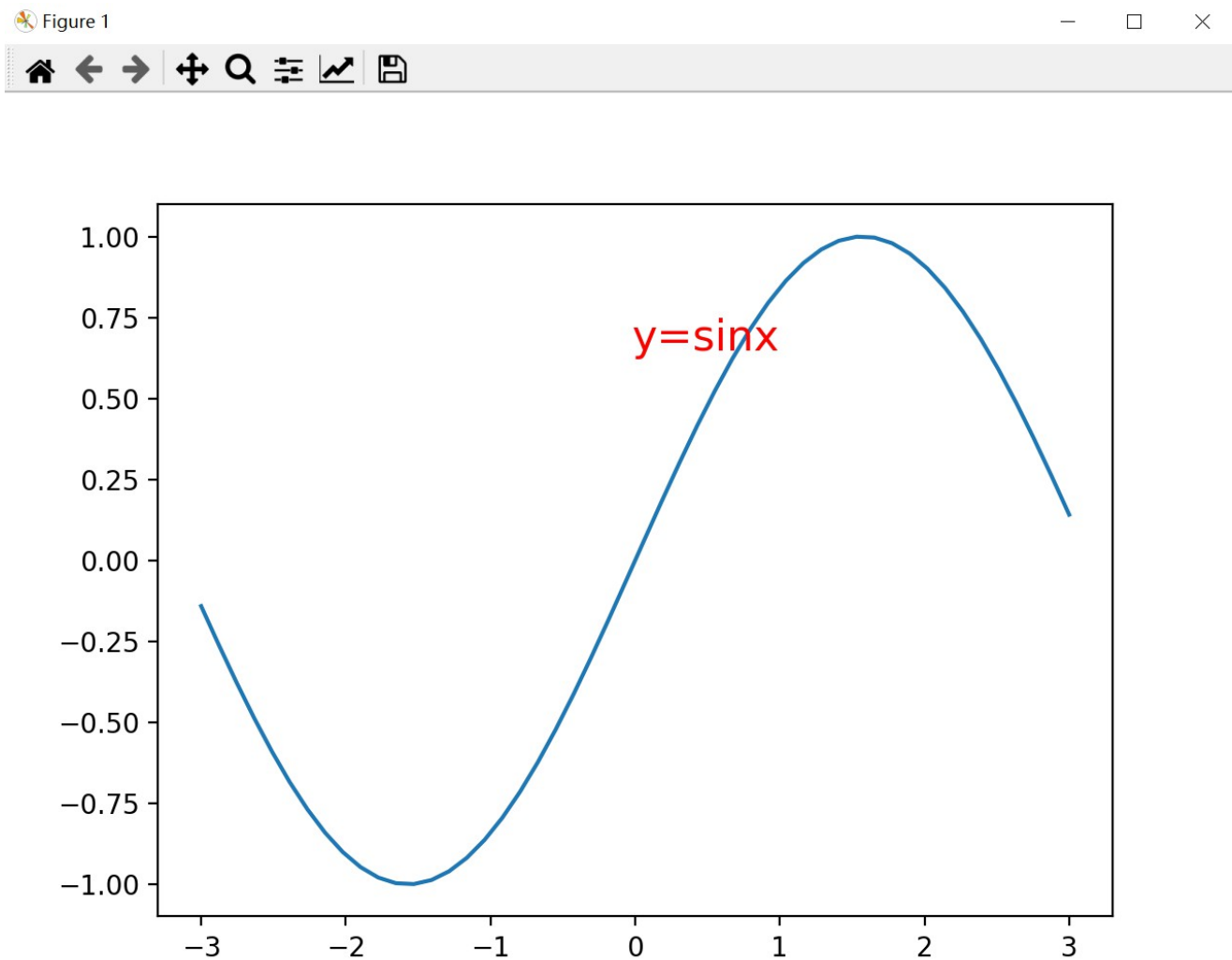
- **x:** 文字开始写的 x 位置
- **y:** 文字开始写的 y 位置
- **s:** 需要写的文字（**string** 格式）
- **size:** 文字大小
- **color:** 文字颜色
- **ha:** 设置垂直对齐方式，可选参数：'left', 'right', 'center'
- **va:** 设置水平对齐方式，可选参数：'center', 'top', 'bottom', 'baseline'

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

x = np.linspace(-3, 3, 50)
y = np.sin(x)

plt.figure()
plt.plot(x, y)

plt.text(x=1.1, y=0.6, s="y=sinx", size=16, color="red")
plt.show()
```



plt.scatter() 画散点图

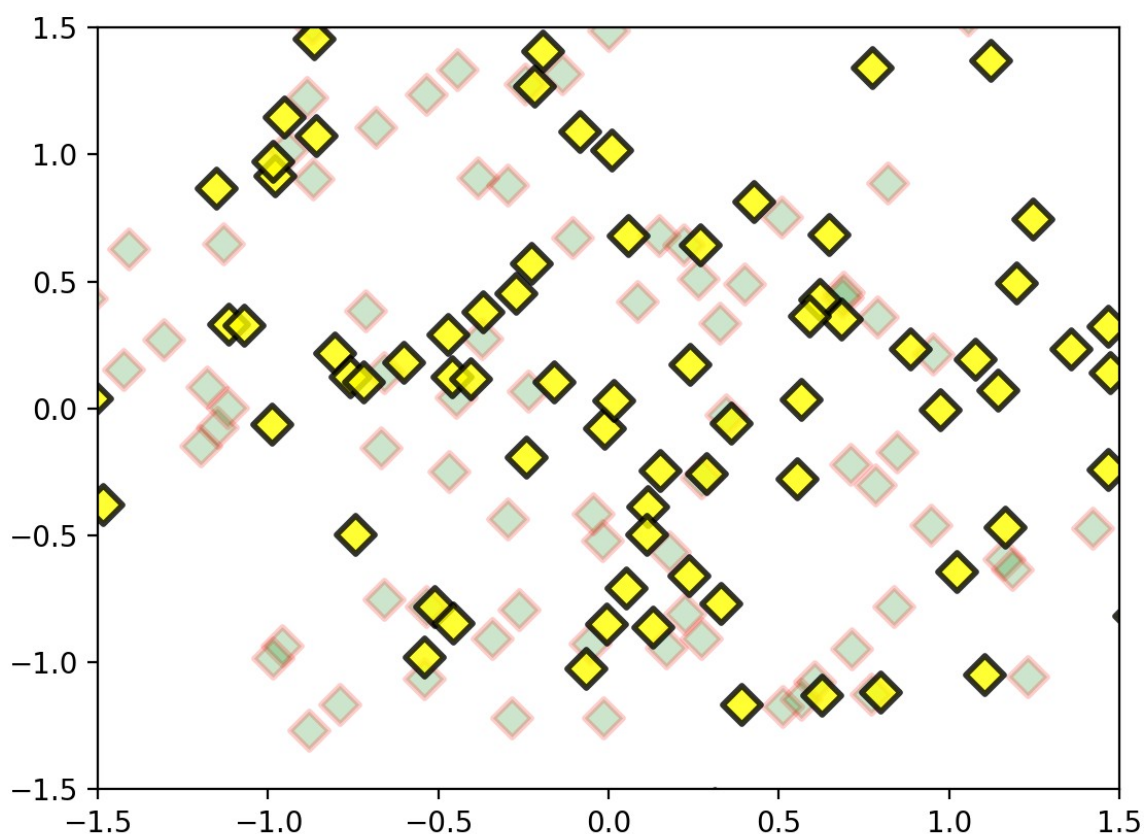
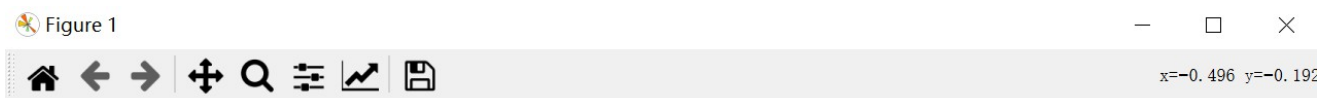
plt.scatter(x, y, s, c, marker, alpha, linewidths, edgecolors)

- x: 形状为(n,)的数组，绘图的 x 轴数据
- y: 形状为(n,)的数组，绘图的 y 轴数据
- s: 标记点的大小（用数字大小表示）
- c: 标记点的颜色
- marker: 标记的样式，默认的是 'o'
- alpha: 透明度（0-1之间）
- linewidths: 标记点边框线的宽度
- edgecolors: 标记点的边框线颜色

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

# 从正态分布中抽取100个随机样本
x1 = np.random.normal(0, 1, 100)
y1 = np.random.normal(0, 1, 100)
x2 = np.random.normal(0, 1, 100)
y2 = np.random.normal(0, 1, 100)

plt.scatter(x1, y1, s=90, c='green', marker='D', alpha=0.2,
            linewidths=2, edgecolors='red')
plt.scatter(x2, y2, s=90, c='yellow', marker='D',
            alpha=0.8, linewidths=2, edgecolors='black')
plt.xlim((-1.5, 1.5))
plt.ylim((-1.5, 1.5))
plt.show()
```



plt.bar() 画条形图

`plt.bar(x, height, width, color, edgecolor, alpha, linewidth, bottom, align)`

- **x:** x 坐标
- **height:** 条形的高度
- **width:** 条形的宽度，默认是0.8
- **color:** 条形的颜色
- **edgecolor:** 条形边框的颜色
- **alpha:** 颜色的透明度（0~1）
- **linewidth:** 条形边框的宽度
- **bottom:** 条形底边的起始位置（即y轴的起始坐标）

- align: 条形的对齐方式（默认为 "center", 表示刻度和条形中心对齐, "edge" 表示刻度和条形左边对齐）

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

x0 = np.arange(12)
# 从均匀分布中抽取12个随机样本
h1 = np.random.uniform(0.5, 1.0, 12)
h2 = np.random.uniform(0.5, 1.0, 12)

plt.figure()
# 用负号来画关于x轴对称的条形图，底边各上浮0.02的距离使得分开
plt.bar(x0, +h1, color='yellow', edgecolor='blue',
        bottom=0.02)
plt.bar(x0, -h2, color='green', edgecolor='red',
        bottom=-0.02)

for i in range(len(x0)):
    # y值保留小数点后两位，数字上浮0.03为了好看
    plt.text(x0[i], h1[i]+0.03, f"{h1[i]:.2f}",
             va='bottom', ha='center')
    plt.text(x0[i], -h2[i]-0.03, f"{-h2[i]:.2f}", va='top',
             ha='center')

# 限制坐标轴显示区域，为了图显示美观
plt.xlim(-1, 12)
plt.ylim(-1.2, 1.2)

# 去掉坐标轴刻度
plt.xticks([])
plt.yticks([])

plt.figure()
```

条形宽度默认是0.8，要把两个条形放在一起，那么每个宽度就是0.4，再互相错位0.2就可以刚好挨着

```
plt.bar(x0-0.2, h1, width=0.4, color='blue', alpha=0.4)
```

这里用了align参数，第一个参数指定为x0+0.2也可以

```
plt.bar(x0, h2, width=0.4, color='gray', align='edge')
```

限制坐标轴显示区域，为了图显示美观

```
plt.xlim(-1, 12)
```

```
plt.ylim(0, 1.1)
```

去掉坐标轴刻度

```
plt.xticks([])
```

```
plt.yticks([])
```

```
plt.show()
```

Figure 1

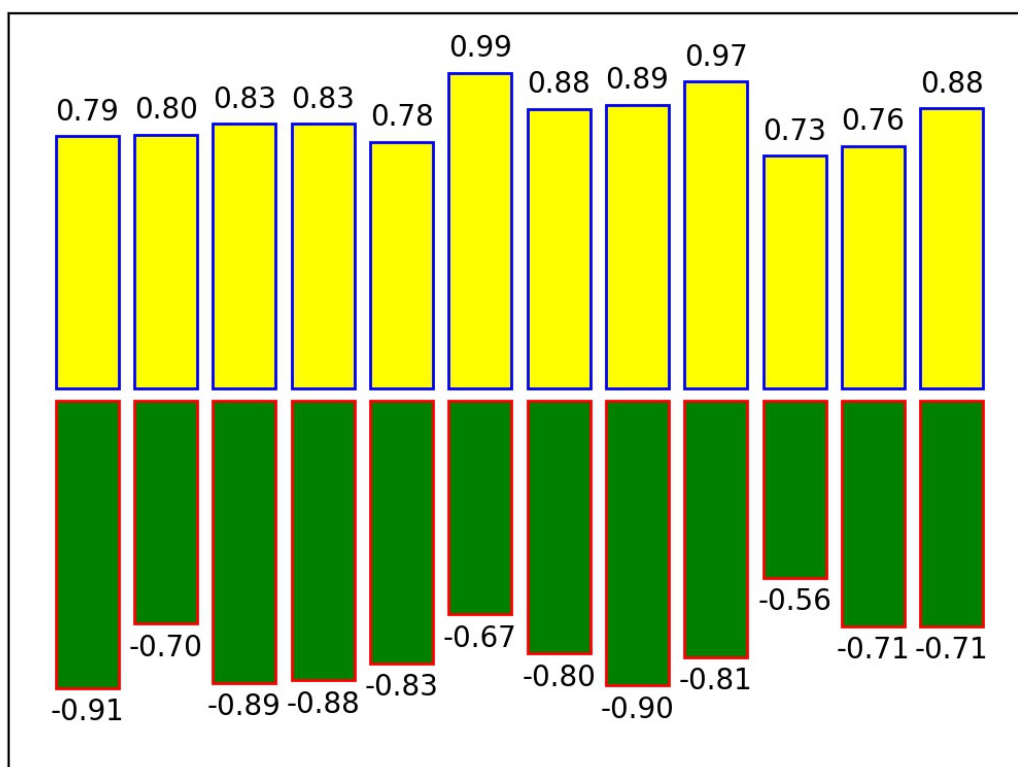
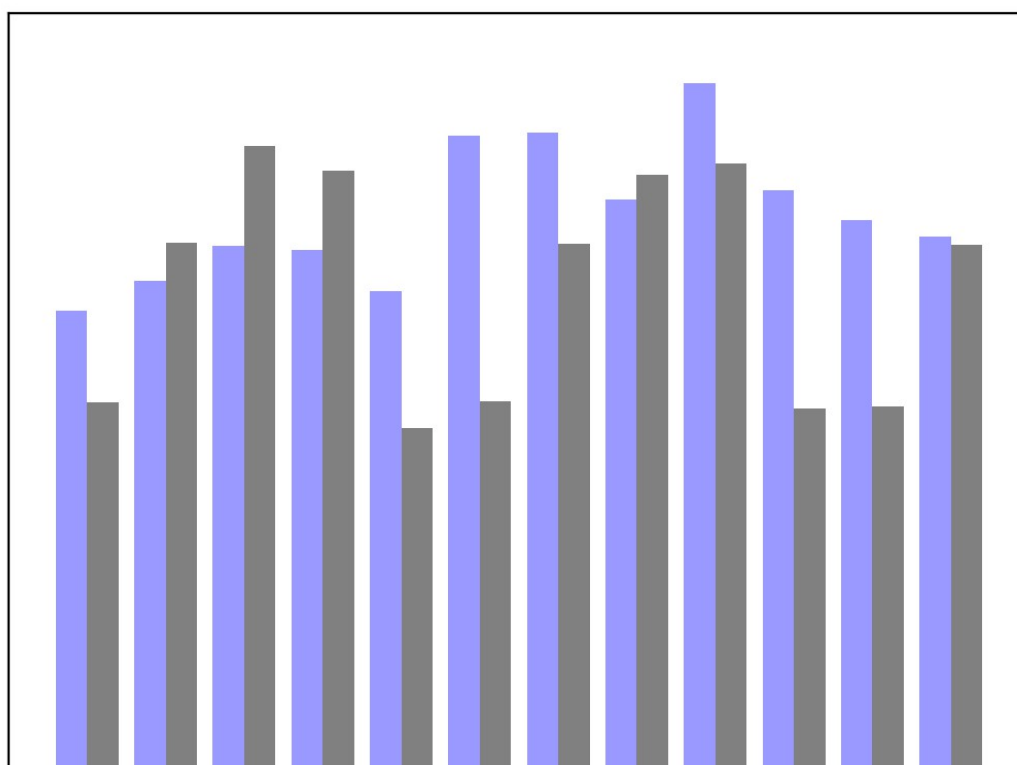


Figure 2



plt.imshow() 数据转图像

传入图像的像素值数据，就可以绘制出图像，然后通过 `plt.show()` 来显示图像

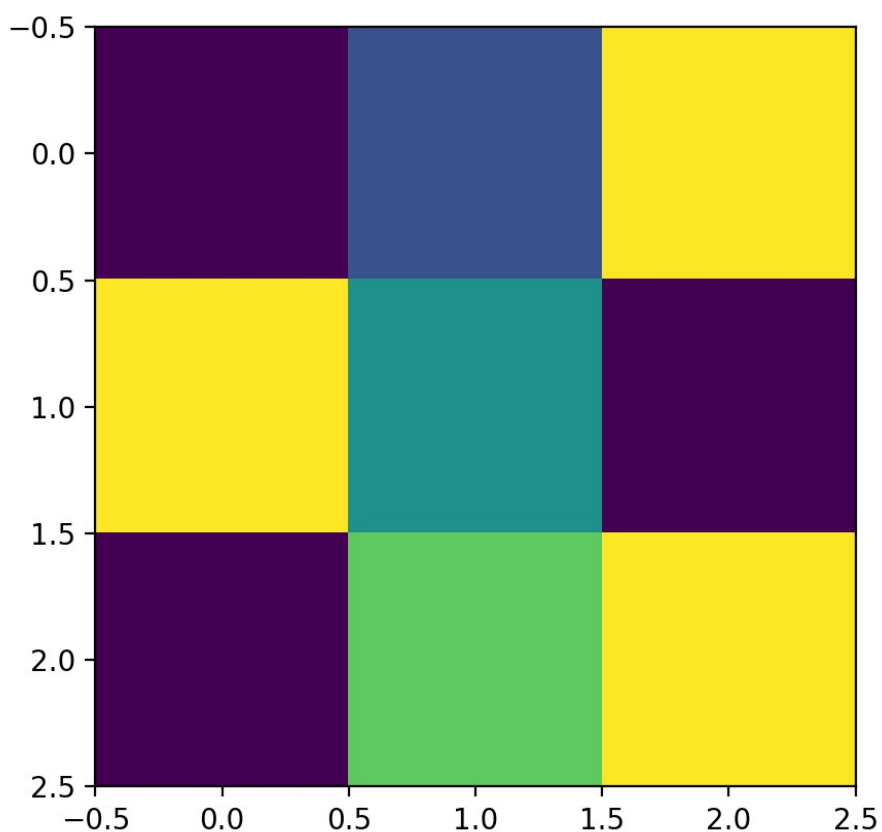
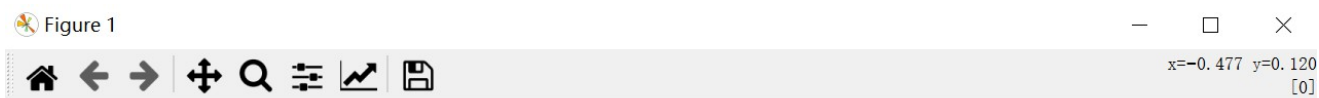
`plt.imshow(X, cmap, alpha)`

- **X**: 图像数据（可以是二维的，比如 灰度图；也可以是三维的，比如 RGB 图）
- **cmap**: 默认是彩色的，灰度图是 "Greys"，可以通过报错来知道有哪些值可选
- **alpha**: 透明度（0~1）

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

plt.figure()

# 自己造一个shape为(3, 3)的图像数据
data = np.array([[0, 50, 200], [200, 100, 0], [0, 150, 200]])
plt.imshow(data)
# plt.imshow(data, cmap="Greys")
# plt.imshow(data, cmap="Greys", alpha=0.3)
plt.show()
```



plt.subplot() 创建子图

plt.subplot(nrows, ncols, index)

- **nrows**: 行数
- **ncols**: 列数
- **index**: 指定第几个子图（从1开始）

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

```
x = np.linspace(-3, 3, 50)
y1 = 2*x + 1
```

```
y2 = x**2
```

```
y3 = np.sin(x)
```

```
y4 = np.tan(x)
```

```
plt.figure()
```

```
# 在画布上创建2行2列的子图，并在第1个子图中绘画
```

```
plt.subplot(2, 2, 1)
```

```
plt.plot(x, y1)
```

```
# 在画布上创建2行2列的子图，在第2个子图中绘画
```

```
plt.subplot(2, 2, 2)
```

```
plt.plot(x, y2)
```

```
# 在画布上创建2行2列的子图，在第3个子图中绘画
```

```
plt.subplot(2, 2, 3)
```

```
plt.plot(x, y3)
```

```
# 在画布上创建2行2列的子图，在第4个子图中绘画
```

```
plt.subplot(2, 2, 4)
```

```
plt.plot(x, y4)
```

```
plt.figure()
```

```
# 在画布上创建2行1列的子图，并在第1个子图中绘画
```

```
plt.subplot(2, 1, 1)
```

```
plt.plot(x, y1)
```

```
# 在画布上创建2行3列的子图，并在第4个子图中绘画
```

```
plt.subplot(2, 3, 4)
```

```
plt.plot(x, y2)
```

```
# 在画布上创建2行3列的子图，并在第5个子图中绘画
```

```
plt.subplot(2, 3, 5)
```

```
plt.plot(x, y3)
```

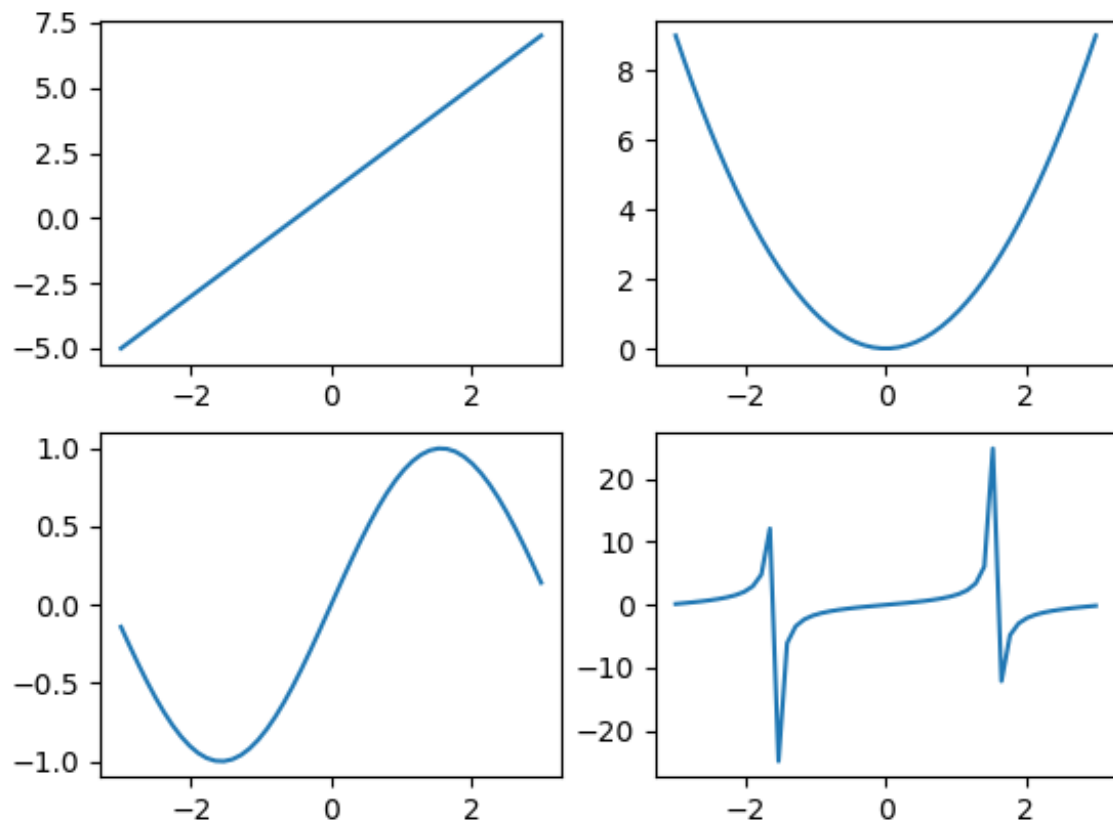
```
# 在画布上创建2行3列的子图，并在第6个子图中绘画
```

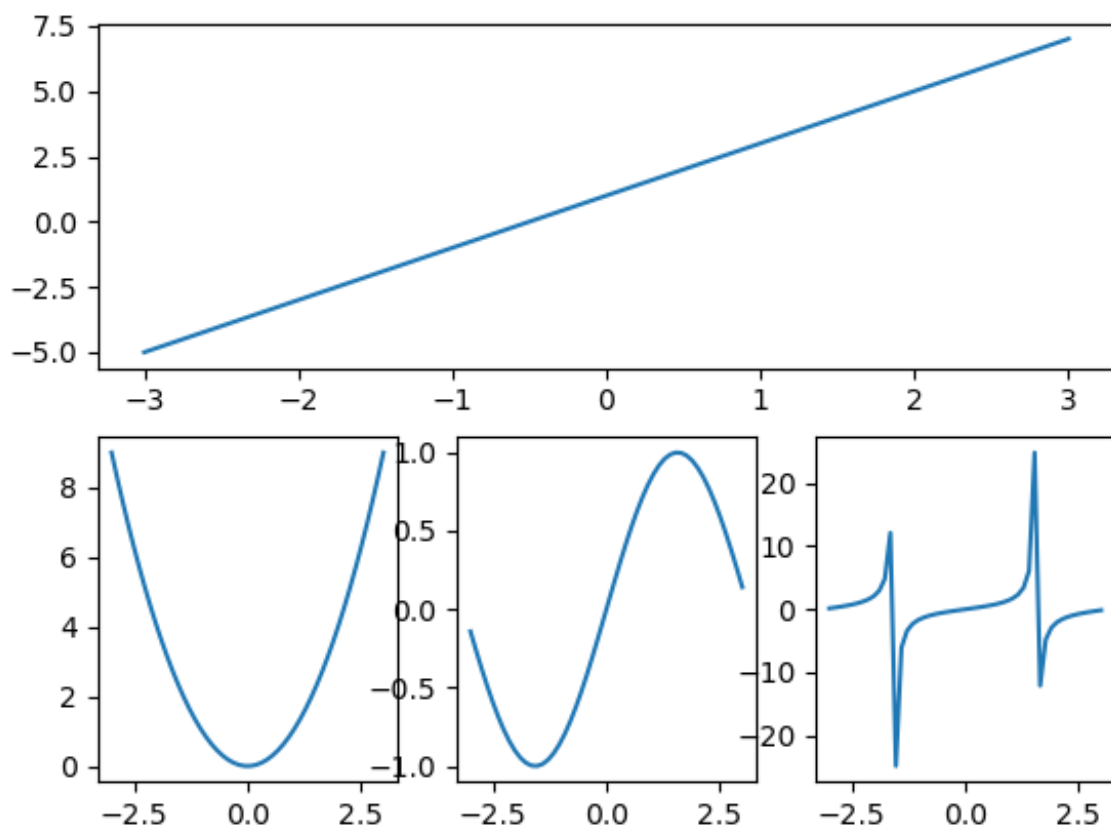


```
plt.subplot(2, 3, 6)
```

```
plt.plot(x, y4)
```

```
plt.show()
```





plt.axes() 画图中图

- `plt.axes()` 可以通过指定相对画布的位置和宽高在一个画布中定制多个坐标轴

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

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

x = np.linspace(-3, 3, 50)
y1 = 2*x + 1
```

```
y2 = x**2
y3 = np.sin(x)
```

```
plt.figure()
```

```
"""
```

对这四个数字的说明：

0.1, 0.1表示坐标框左下角的位置，x和y坐标在画布的10%的位置

0.8, 0.8表示坐标框的宽和高为画布的80%大小 """

```
plt.axes([0.1, 0.1, 0.8, 0.8])
```

```
plt.title('直线') # 标题
```

```
plt.plot(x, y1)
```

```
plt.axes([0.2, 0.6, 0.25, 0.25])
```

```
plt.title('抛物线')
```

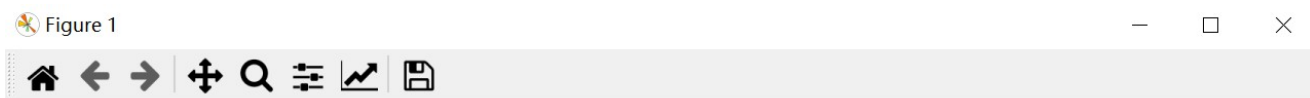
```
plt.plot(x, y2)
```

```
plt.axes([0.6, 0.2, 0.25, 0.25])
```

```
plt.plot(x, y3)
```

```
plt.title('余弦曲线')
```

```
plt.show()
```



plt.savefig() 保存图像

- 通过 `plt.savefig()` 保存图像到指定的路径

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

y2 = []
y1 = np.linspace(-3, 3, 50)

for i in y1**2:
    y2.append(i)
    plt.clf() # 清除数据，放在这个位置是为了循环的清除上一次的数据
    plt.plot(y2) # 只传一个参数时，当做y轴的坐标，x轴默认为[0, 1, 2, 3...]
    plt.pause(0.1) # 每张图暂停0.1秒
plt.savefig('./img.jpg') # 保存图像到指定的路径
```