matplotlib

写在前面

python最好的点在于：当你乱输入参数时，它会在报错里建议你输入一些参数

所以尽情地尝试吧

matplotlib对中文的支持极差，懒得折腾就别用中文

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

plt.show() # 把你设计的图像显示出来（不然只是一个对象）
```

Figure

Figure（图像）即程序运行后展现出的窗口，是matplotlib中的基本结构

Figue具有默认参数，可以不进行设置

参数	默认值	描述
num	1	图像的数量
figsize	figure.figsize	图像的长和宽（英寸）
dpi	figure.dpi	分辨率（点/英寸）
facecolor	figure.facecolor	绘图区域的背景颜色
edgecolor	figure.edgecolor	绘图区域边缘的颜色
frameon	True	是否绘制图像边缘

plot(subplot)

plot

plot(图样)是figure上的元素，plot方法的格式为

```
plt.plot(*args:ArrayLike,  #接受多个类列表参数（默认第一个为X轴）
                 scalex: bool = True, #x轴自动调节
                 scaley: bool = True, #y轴自动调节
                 data: Any | None = None, #可选择图的类型，如ob为散点图
```

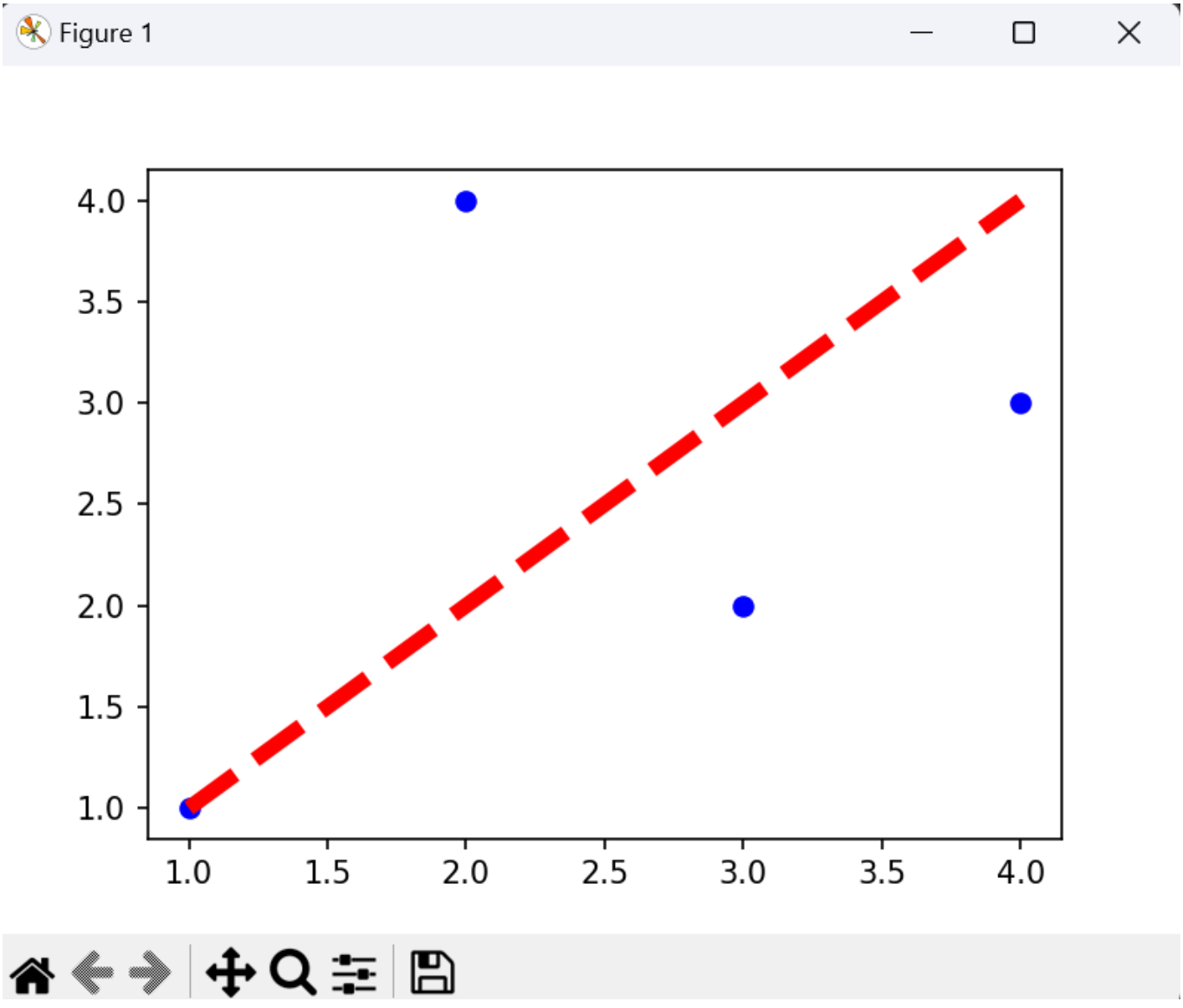
```
                **Kwargs: Any #可选择的参数
               ) -> list[Line2D]

kwargs 包括：label="图像的标签"；color="图像的颜色，支持英文";linewidth=int(线的粗细)；
linestyle='线的风格，如'-','--''等
```

比如我们可以简单绘制一个折线图和散点图

```
plt.plot([1,2,3,4],[1,4,2,3],"ob",color="blue")
plt.plot([1,2,3,4],[1,2,3,4],color="red",linewidth=5,linestyle='--')
plt.show()
```

得到图像为



但是反复在一个plot里画图可能导致线互相重叠严重，因为使用plot方法只有一个图像

subplot

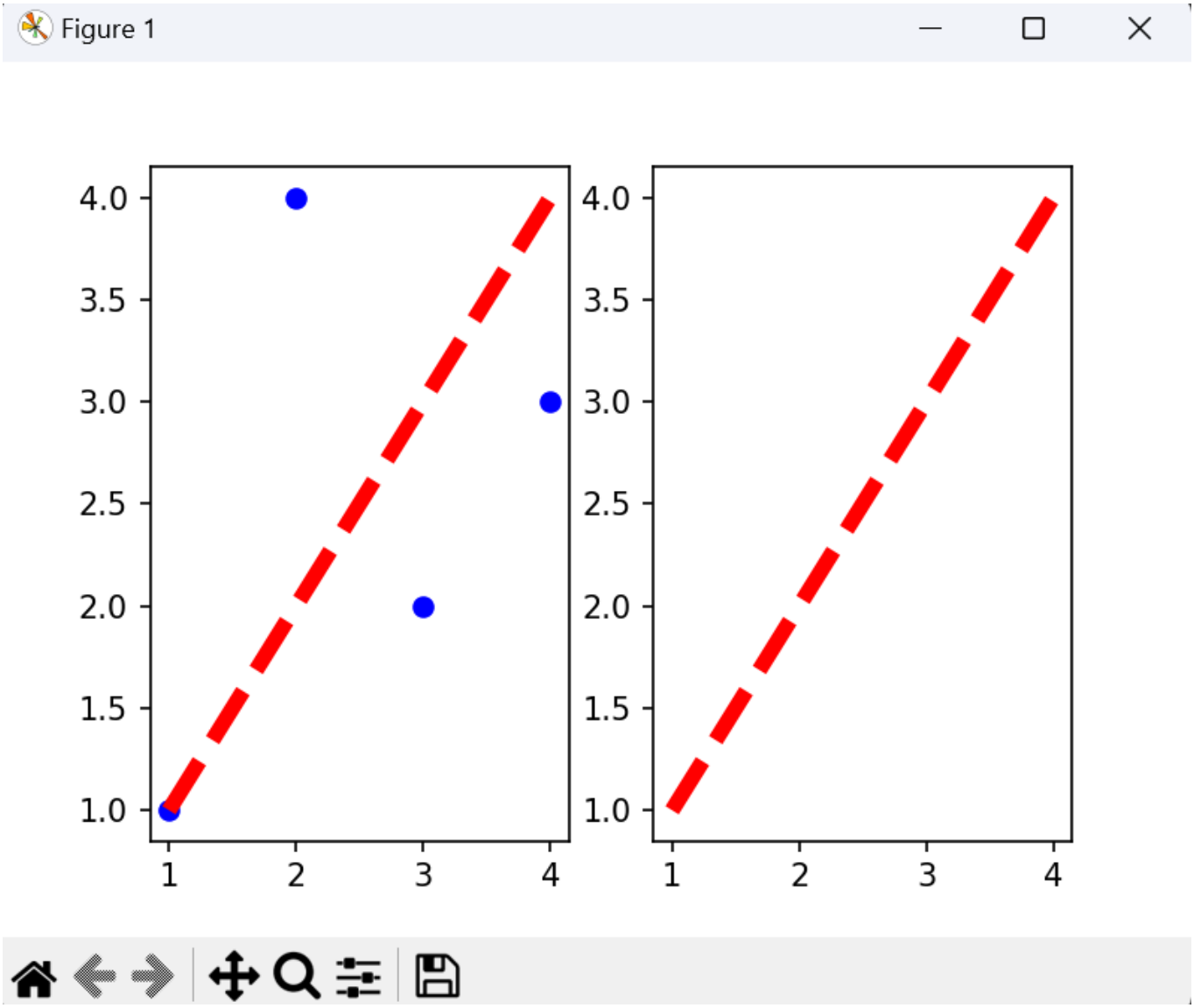
subplot可以在一个figure里创建多个图像，但是需要自行设计分布，方法为：

```
plt.subplot(nrows,ncols,index) # 创建nrows行，ncols列的图像分布，指定index
默认为 plt.subplot(1,1,1)
```

subplot()方法本质上是用来定位图像位置的，在宣布完index后，后续plot函数产生的图像都将展示在指定的图像上，直到再次宣布subplot的index

```
# 一个简单的例子
plt.subplot(1,2,1) # 宣布一个一行两列的图像分布，并指定当前绘制在第一个上
plt.plot([1,2,3,4],[1,4,2,3],"ob",color="blue")
plt.plot([1,2,3,4],[1,2,3,4],color="red",linewidth=5,linestyle='--')
plt.subplot(1,2,2) # 重新宣布绘制在第二个上
plt.plot([1,2,3,4],[1,2,3,4],color="red",linewidth=5,linestyle='--')
plt.show()
```

得到的图像为



一些调整用的方法

针对单个图像的调整

就如subplot章节所言，我们调整某个图像或者往某个图像上画图时，都要指定是哪个图像，否则就会指定到默认图像上（虽然也无所谓）

标题

```
plt.title("标题", loc="位置，支持英文")
# 设置plot的标题，如果你使用了subplot,那么你可以为每个子图设置一个标题
# 并且可以在指定subplot前设置一次标题，这个标题可以视为figure的大标题
```

坐标轴调整

```
plt.xlim(left,right) # 设置横轴的上下限
plt.xticks(Array) # 设置横轴记号,所有在列表中的点都会被标出
plt.xlabel("string",loc="位置") #设置x轴名称和位置
plt.ylim(left,right) # 设置纵轴的上下限
plt.yticks(Array) # 设置纵轴记号,同上
plt.ylabel("string",loc="位置") #设置y轴名称和位置
```

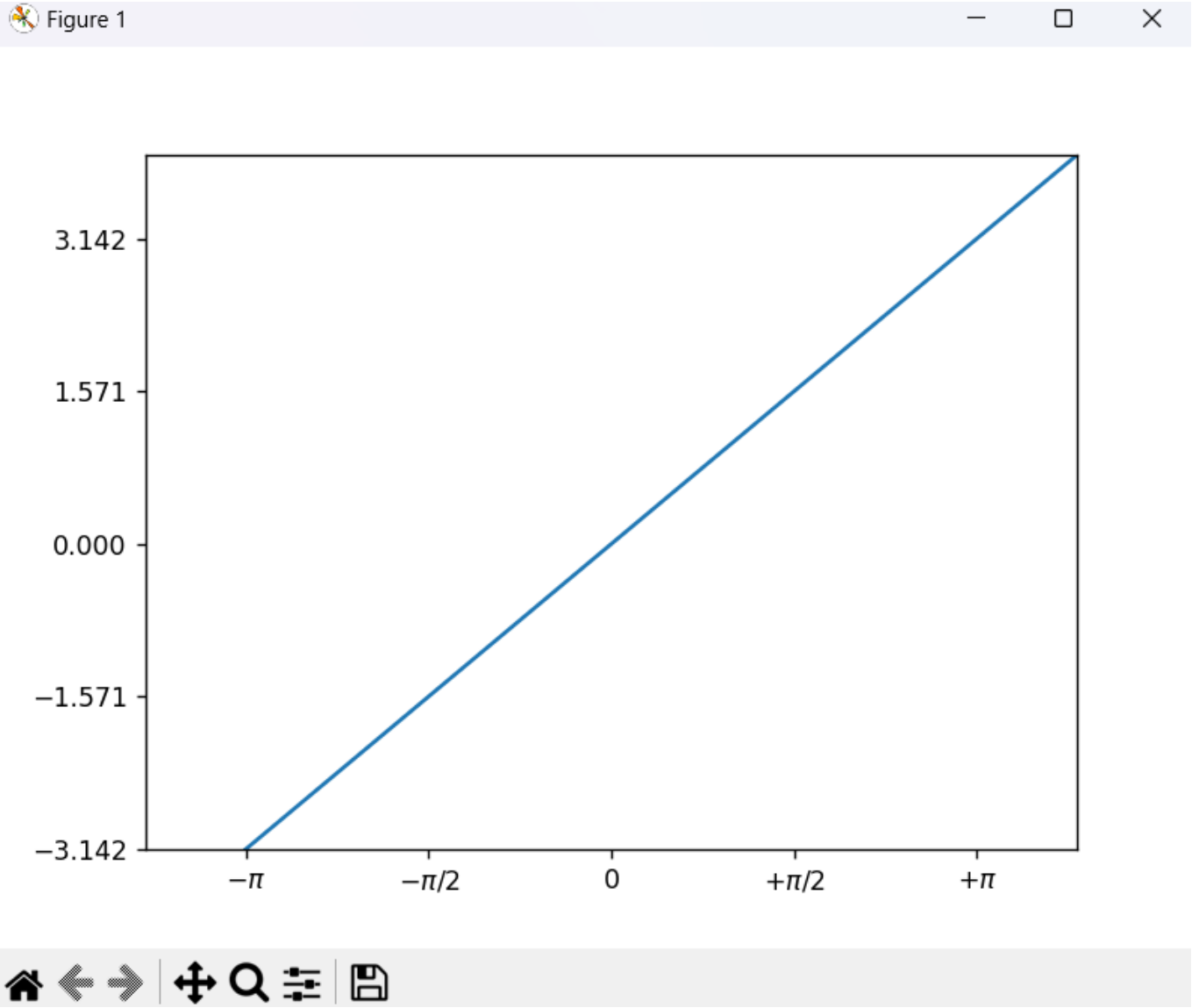
注意：当你设置了上下限时，plot的xscale和yscale将不再工作
这意味着如果你输入的数据点超出了你自己设置的范围，它将不再被显示

加强版坐标轴记号

在使用的时候你会难受的发现记号方法用的是浮点数，连分数都是转化成小数显示的，还有精度这实在是无法接受，标个pi都成了3.142
于是我们可以使用LaTex的语法实现给每个点取名字(只是取名字，位置还是小数的位置)

```
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
       [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])
plt.yticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
```

对比一下坐标轴的显示



这下舒服了

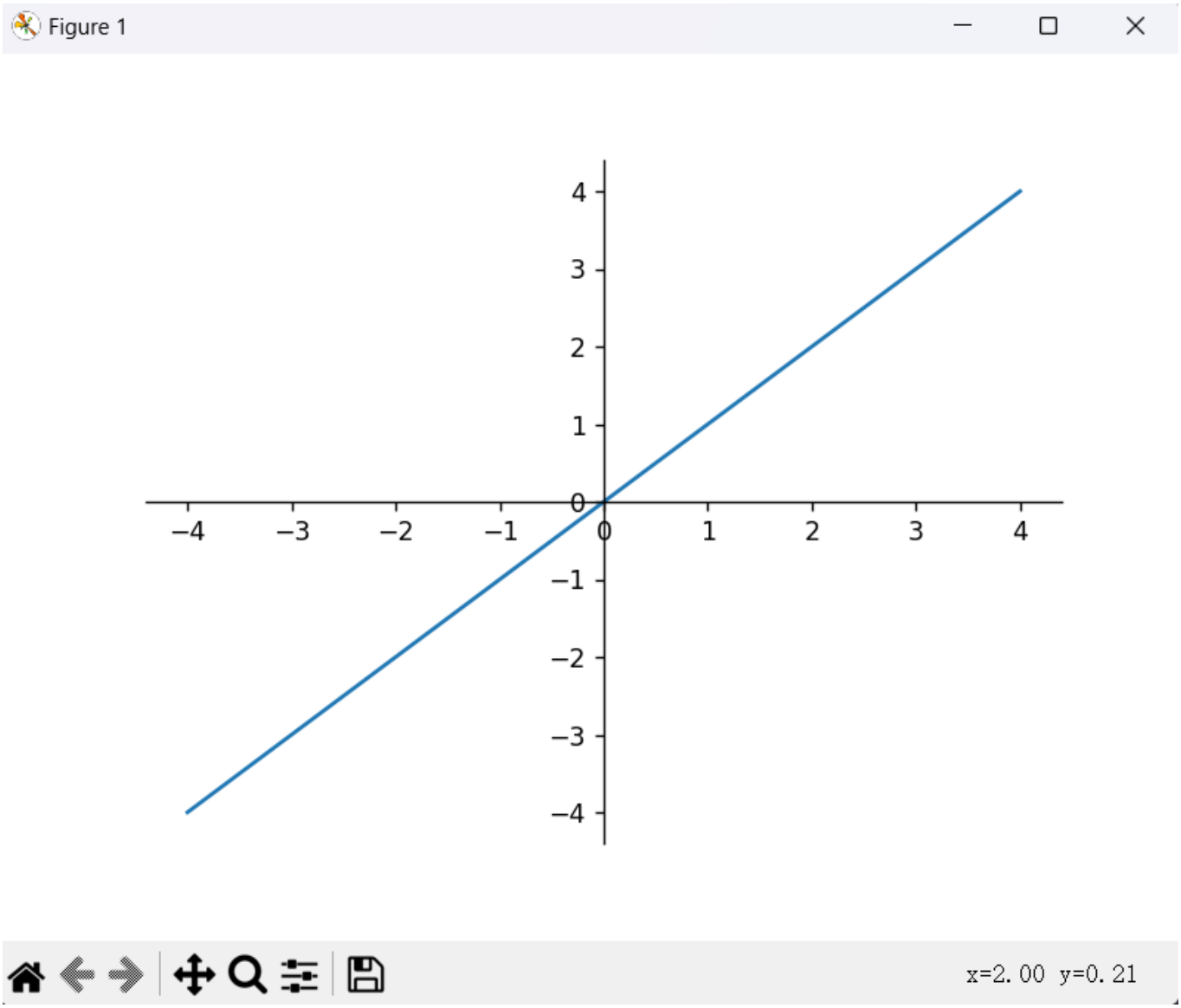
移动脊柱（Spines）——创建平面直角坐标系

- **坐标轴线**和上面的记号连在一起就形成了脊柱（Spines），它记录了数据区域的范围。它们可以放在任意位置，默认情况下它们被放在图的四边。
- 实际上每幅图有四条脊柱（上下左右），如果要将脊柱放在图的中间，我们必须将其中的两条（上和右）设置为无色，然后调整剩下的两条到合适的位置——数据空间的 0 点。

```
# 步骤很简单
[ax = plt.gca()](<plt.plot([-4,-3,-2,-1,1,2,3,4],[-4,-3,-2,-1,1,2,3,4])
ax = plt.gca()  # 获取脊柱对象
ax.spines['right'].set_color('none') #右脊柱设为无色
ax.spines['top'].set_color('none')  #上脊柱设为无色
ax.xaxis.set_ticks_position('bottom')   # 调整x轴的记号位置
ax.spines['bottom'].set_position(('data',0))    # 调整下脊柱的位置
```

```
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',0))
plt.show()>)
# 位置参数与四个脊柱的代号相同
```

得到图像为



实际上通过gca()获取到的对象可以对坐标轴线进行大幅度的改造

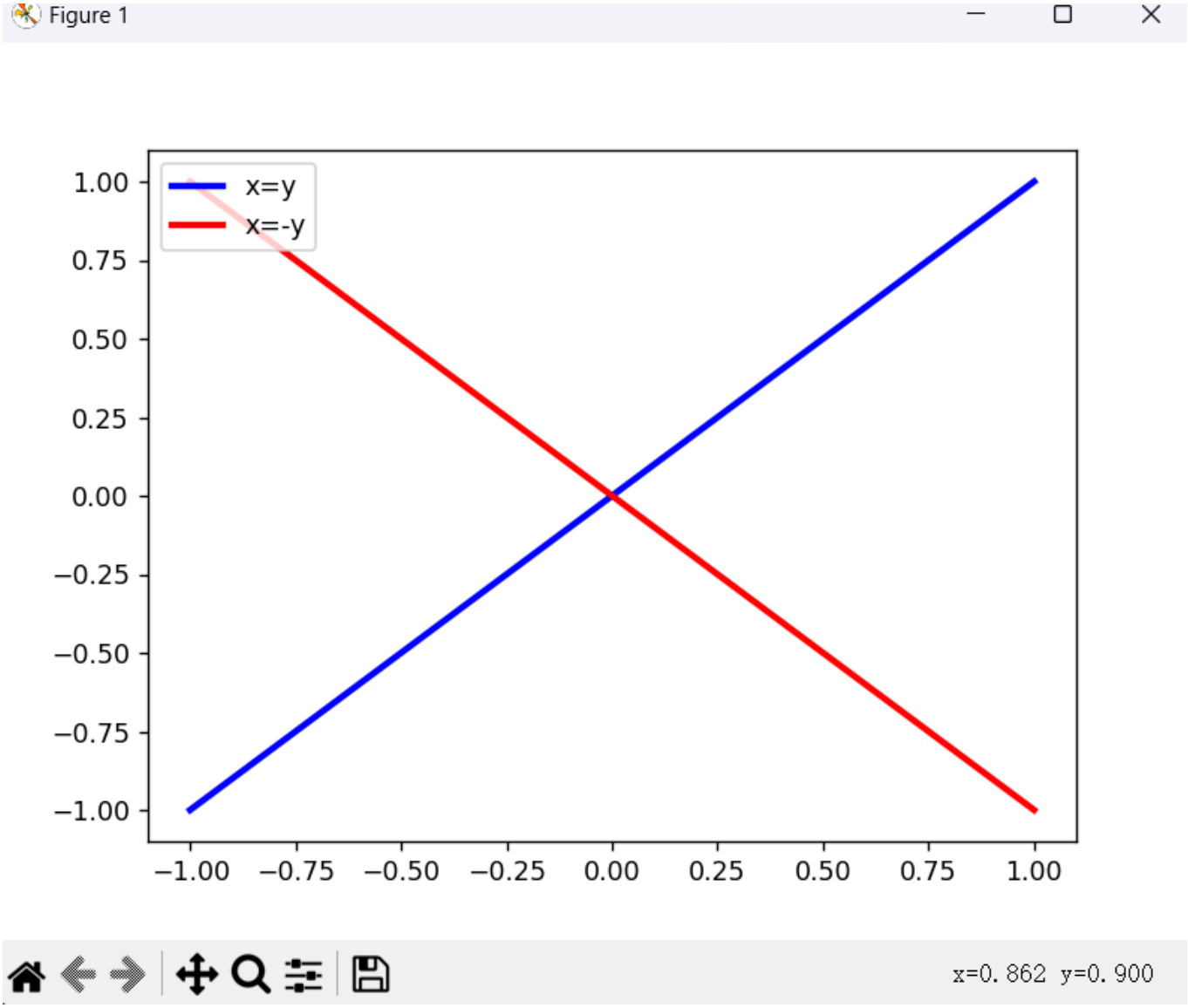
添加图例

如果你在使用plot方法时设置了label参数，那么你就可以通过legend方法创建一个图例来展示label

```
# 一个简单的例子
plt.plot([-1,0,1], [-1,0,1], color="blue", linewidth=2.5, linestyle="-",
label="x=y")
```

```
plt.plot([-1,0,1], [1,0,-1], color="red",  linewidth=2.5, linestyle="-", label="x=-y")
plt.legend(loc='upper left') # loc(ation)参数可指定位置，可以用best参数自动抉择
plt.show()
```

得到图像为



其他图样类型

matpltlib支持非常多种类的图样，可以自行探索