# 表格绘制基本操作

## 环境配置

**导包和配置**

```
import matplotlib.pyplot as plt #导包

%matplotlib notebook  # 魔法配置,可以让表格显示在jupyter里面
```

如果提示如下错误：

```
ModuleNotFoundError: No module named 'matplotlib'
```

安装此模块即可：**(安装前确认自己的环境是否和开发环境一致)**

```
pip install matplotlib
```

**中文字体支持 SimHei.ttf**

1）下载上述字体文件（或从自己电脑搜索SimHei.ttf）

2）到SimHei.ttf所在文件夹，鼠标右键`Open in Terminal`, 执行以下命令

```
# 拷贝当前目录下的simhei.ttf文件到指定目录
sudo cp SimHei.ttf /usr/share/fonts/
# 删除matplotlib配置缓存 （这里的路径中的/home/itcast/注意用自己的用户名）
sudo rm -r /home/itcast/.cache/matplotlib/
# 添加配置内容（这里的路径中的/home/itcast/注意用自己的用户名）
echo -e "font.family : sans-serif \nfont.sans-serif : SimHei
\naxes.unicode_minus : False" >>
/home/itcast/anaconda3/envs/notebook/lib/python3.7/site-packages/matplotlib/mpl-
data/matplotlibrc
```

3）在代码声明plt后添加以下配置

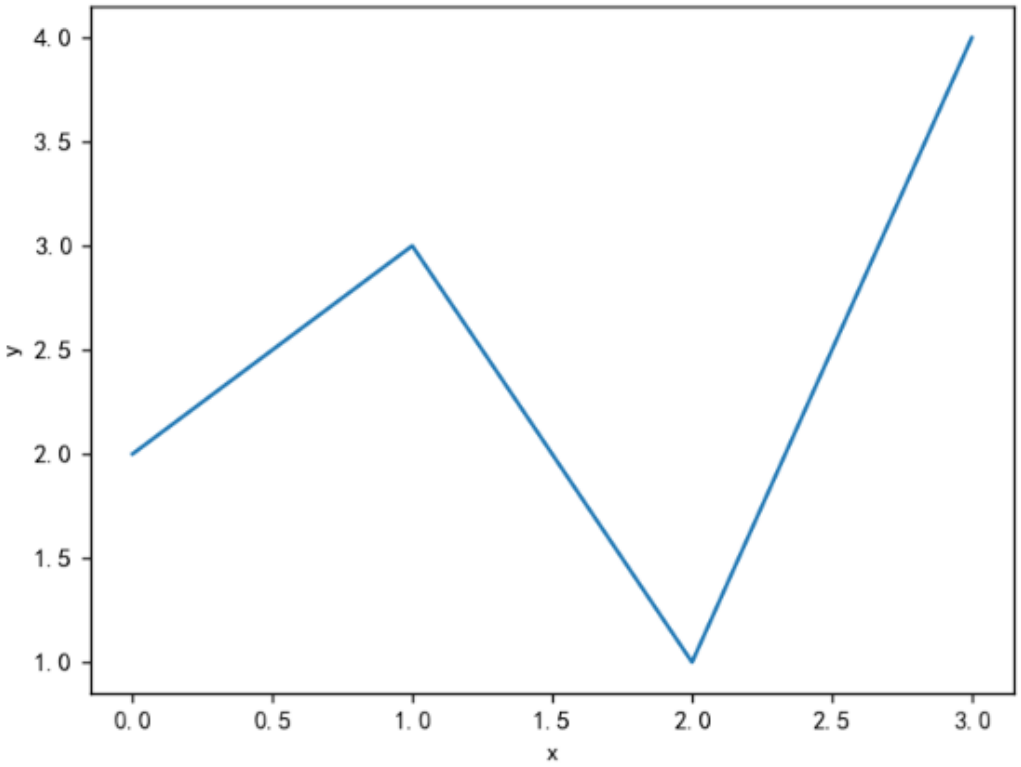
```
plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号
```

## 绘图操作

- 绘制折线图

```
fig= plt.figure()
plt.plot([2, 3, 1, 4]) # 默认x是0开始, (0,2), (1,3), (2,1), (3,4)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

输出如下:

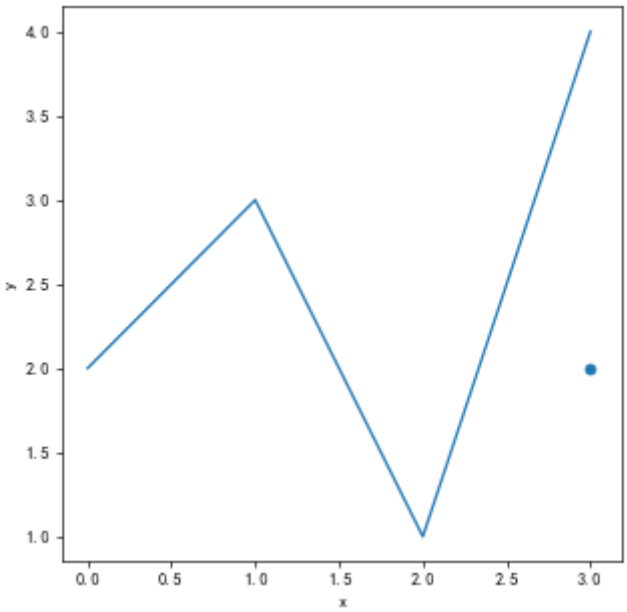



- 指定绘制图形的大小

  宽6英寸，高6英寸，每英寸占60像素

```
fig= plt.figure(figsize=(6,6), dpi=60)
plt.plot([2, 3, 1, 4]) # 默认x是0开始, (0,2), (1,3), (2,1), (3,4)
plt.scatter(3, 2)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

输出如下:

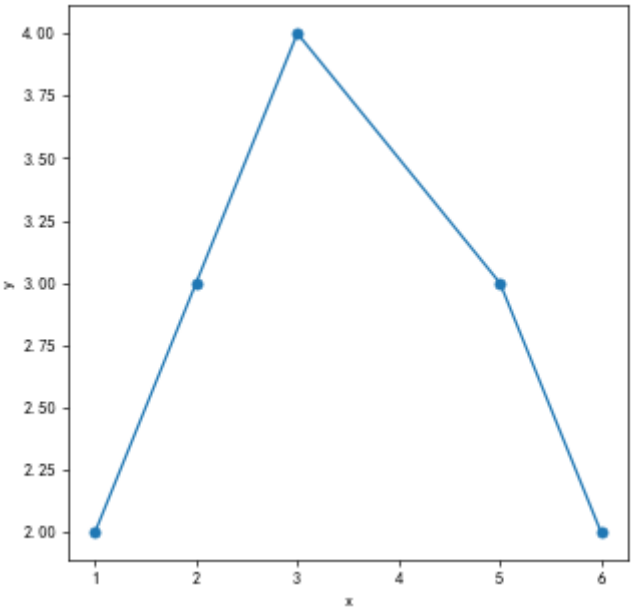



- 包含路径点的折线图

```
x_data = np.array([1, 2, 3, 5, 6])
y_data = np.array([2, 3, 4, 3, 2])

fig= plt.figure(figsize=(6,6), dpi=60)
plt.scatter(x_data, y_data)
plt.plot(x_data, y_data)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

输出如下:




- 指定线宽和颜色


- 指定x和y轴的范围,和标签内容

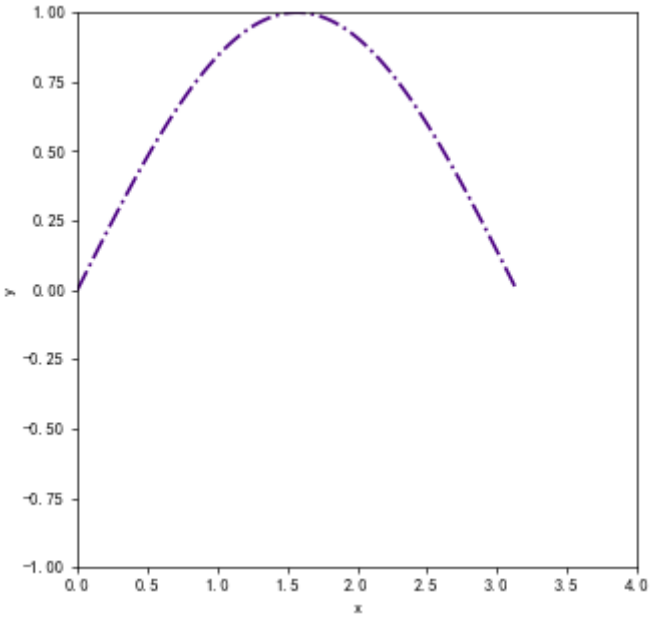
```
# x_data = np.array([0, np.pi / 4, np.pi / 2])
x_data = np.linspace(0, np.pi, 100)
y_data = np.sin(x_data)

fig= plt.figure(figsize=(6,6), dpi=60)
# plt.scatter(x_data, y_data)
plt.plot(x_data, y_data, color="indigo", linewidth=2, linestyle="-.")
plt.xlabel('x')
plt.ylabel('y')

# 指定绘图坐标范围
plt.xlim(0.0, 4.0)
plt.ylim(-1.0, 1.0)

plt.show()
```

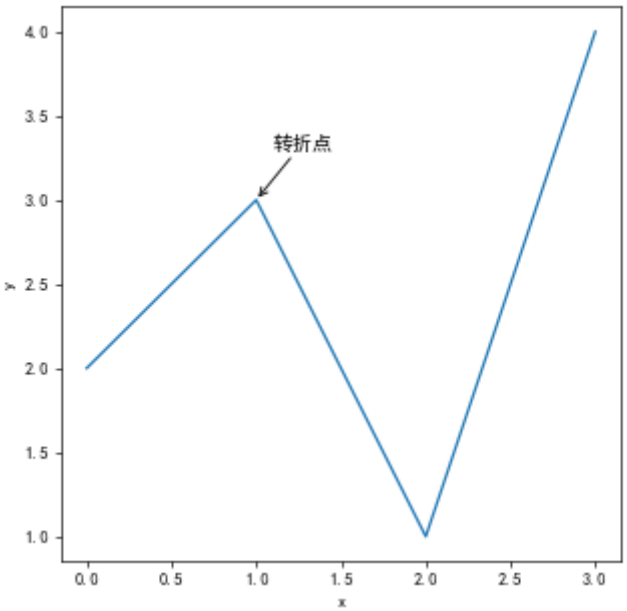
输出内容:




- 设置关键点标注

```
fig = plt.figure(figsize=(6, 6), dpi=60) #
plt.plot([2, 3, 1, 4])
plt.xlabel('x')
plt.ylabel('y')
plt.annotate('转折点',xy=(1, 3),  # 箭头指向的位置
             xytext=(+10, +30),   # 文字偏移的位置
             textcoords='offset points',
             fontsize=12, arrowprops=dict(arrowstyle="->"))
plt.show()
```

输出结果：




# 绘制子表格

- 准备数据

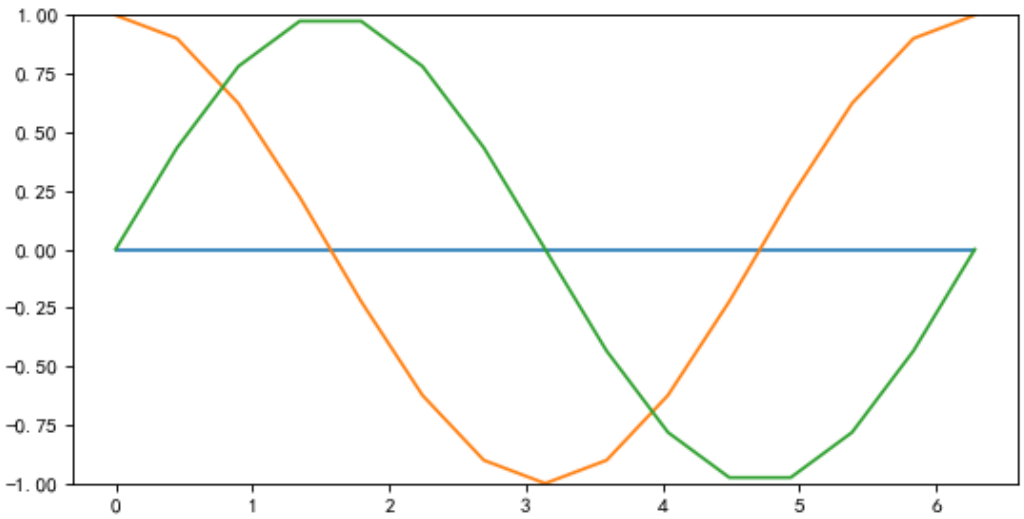
```
# 构建余弦函数和正弦函数
X = np.linspace(0, 2 * np.pi, 15)
C = np.cos(X)
S = np.sin(X)
```

- 同一个表格绘制多个数据

```
# 显示余弦函数和正弦函数
fig = plt.figure(figsize=(8, 4), dpi=80)
plt.plot(X, np.zeros(len(X)))
# plt.subplot(2, 1, 1)
plt.plot(X, C)
# plt.subplot(2, 1, 2)
plt.plot(X, S)

plt.ylim(-1.0, 1.0)
plt.show()
```

输出结果:



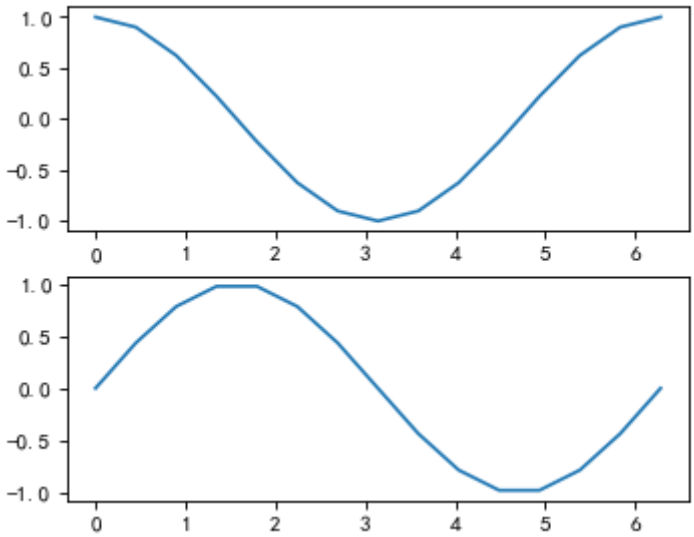

- 同时绘制两个表格

```
# 分成多个表格显示
# 显示余弦函数和正弦函数
fig = plt.figure(figsize=(5, 4), dpi=80)
plt.ylim(-1.0, 1.0)

plt.subplot(2, 1, 1)
plt.plot(X, C)
plt.subplot(2, 1, 2)
plt.plot(X, S)

plt.show()
```

输出结果:


1.0
0.5
0.0
−0.5
−1.0
0
1
2
3
4
5
6
1.0
0.5
0.0
−0.5
−1.0
0
1
2
3
4
5
6