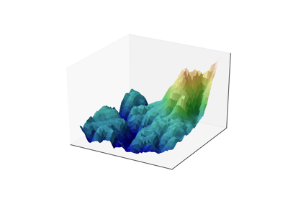
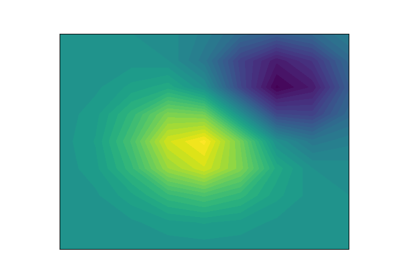
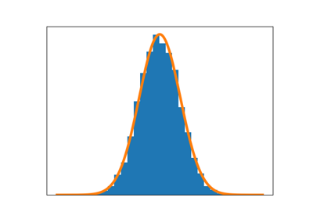
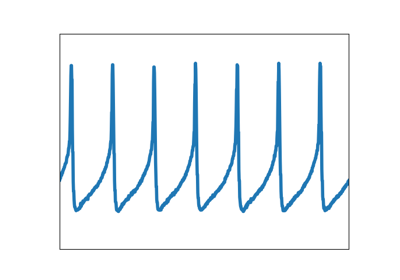
实验3：绘图库matplotlib的使用

**什么是matplotlib？**

Matplotlib是一个Python 2D绘图库，可以生成各种硬拷贝格式和跨平台交互式环境的出版物质量数据。Matplotlib可用于Python脚本，Python和IPython shell，Jupyter笔记本，Web应用程序服务器和四个图形用户界面工具包。

[](https://matplotlib.org/tutorials/introductory/sample_plots.html)

Matplotlib试图让简单易事的事情成为可能。你只需几行代码即可生成绘图，直方图，功率谱，条形图，误差图，散点图等。有关示例，请参阅示例图库和缩略图库。

对于简单的绘图，pyplot模块提供类似MATLAB的接口，特别是与IPython结合使用时。 对于高级用户，你可以通过面向对象的界面或通过MATLAB用户熟悉的一组函数完全控制线型，字体属性，轴属性等。

# 1、plt的使用

ipython notebook中有一个相当方便的语句: %matplotlib inline，可以实现运行cell即出现结果图像。

%matplotlib inline

## 1.1 基本折线图

使用import导入模块matplotlib.pyplot，并简写成plt 使用import导入模块numpy，并简写成np

import matplotlib.pyplot as plt

import numpy as np

使用np.linspace定义x：范围是(-1,1);个数是50. 仿真一维数据组(x ,y)表示曲线1.

x = np.linspace(-1, 1, 50)

y = 2\*x + 1

使用plt.figure定义一个图像窗口. 使用plt.plot画(x ,y)曲线. 使用plt.show显示图像.

plt.figure()

plt.plot(x, y)

plt.show()

**示例：**

%matplotlib inline

import matplotlib.pyplot as plt

import numpy as np

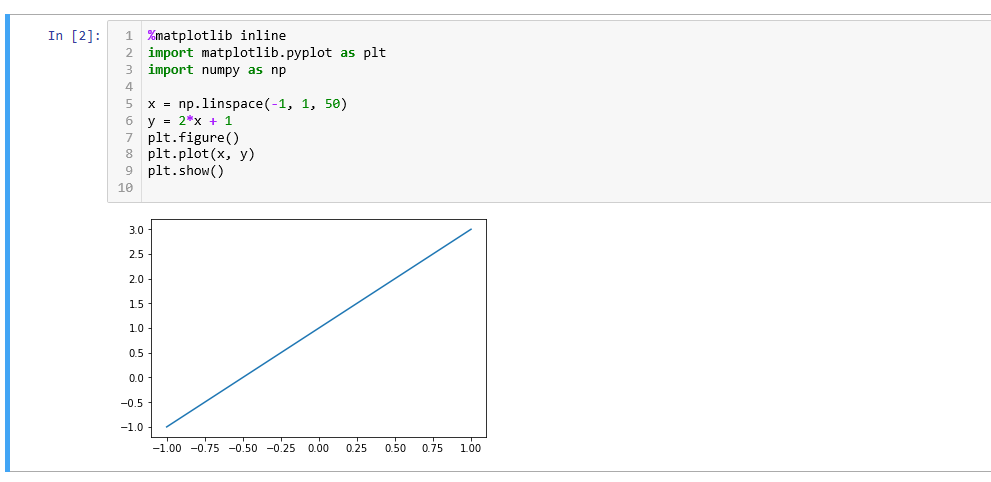
x = np.linspace(-1, 1, 50)

y = 2\*x + 1

plt.figure()

plt.plot(x, y)

plt.show()



**示例：**

import matplotlib.pyplot as plt

import numpy as np

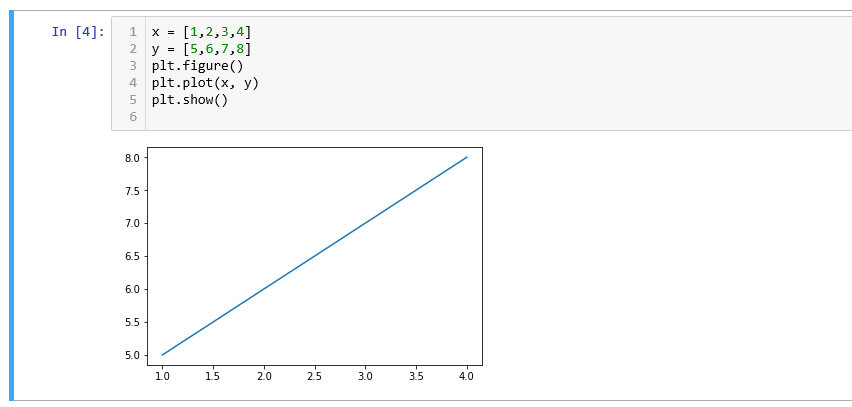
x = [1,2,3,4]

y = [5,6,7,8]

plt.figure()

plt.plot(x, y)

plt.show()



**示例：**

import matplotlib.pyplot as plt

import numpy as np

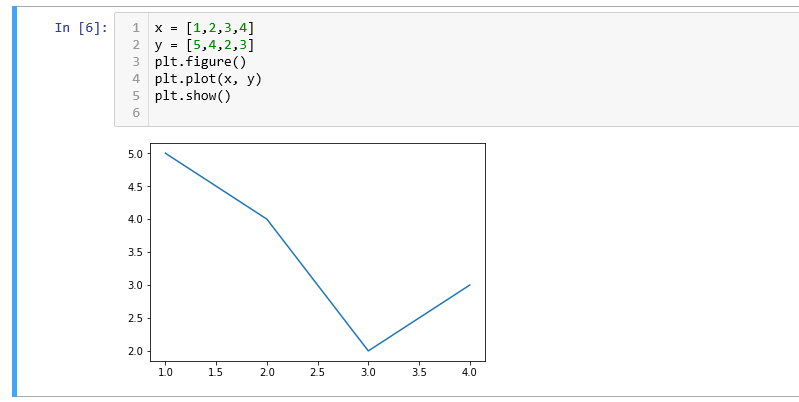
x = [1,2,3,4]

y = [5,4,2,3]

plt.figure()

plt.plot(x, y)

plt.show()

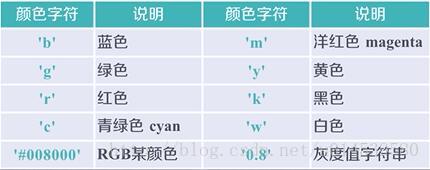


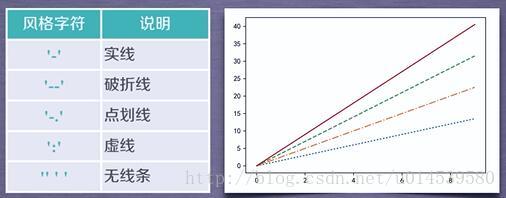
## 1.2 基本样式

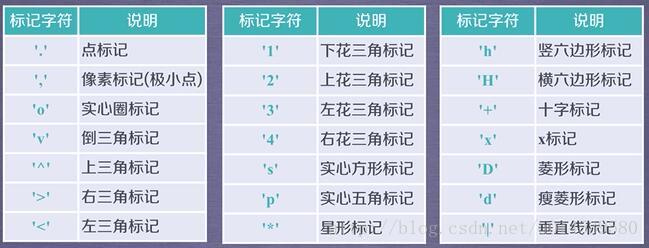
plt.plot(x,y,format\_string,\*\*kwargs)

x轴数据，y轴数据，format\_string控制曲线的格式字串；format\_string 由颜色字符，风格字符，和标记字符等构成

* color 颜色
* linestyle 线条样式
* linewidth 线条粗细
* marker 标记风格
* markerfacecolor 标记颜色
* markersize 标记大小
* markeredgecolor 标记边的颜色
* markeredgewidth 标记边的粗细







**示例：**

%matplotlib inline

import matplotlib.pyplot as plt

import numpy as np

x = [1,2,3,4]

y = [5,4,2,3]

plt.figure()

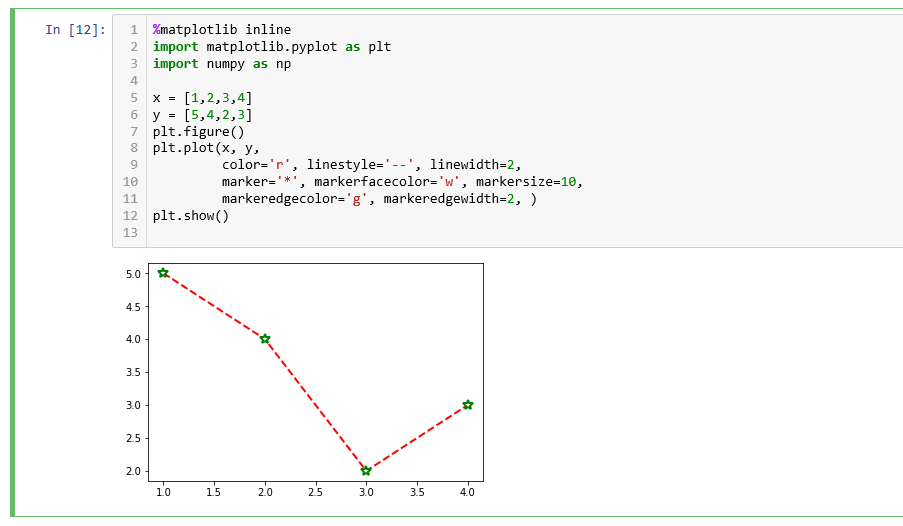
plt.plot(x, y,

color='r', linestyle='--', linewidth=2,

marker='\*', markerfacecolor='w', markersize=10,

markeredgecolor='g', markeredgewidth=2, )

plt.show()



## 1.3 多条折线

可以多次plot，画多条折线

**示例**

%matplotlib inline

import matplotlib.pyplot as plt

import numpy as np

x = [1,2,3,4]

y = [5,4,2,3]

plt.figure()

plt.plot(x, y,

color='r', linestyle='--', linewidth=2,

marker='\*', markerfacecolor='w', markersize=9,

markeredgecolor='g', markeredgewidth=2, )

x = [1,2,3,4]

y = [3,2,3,4]

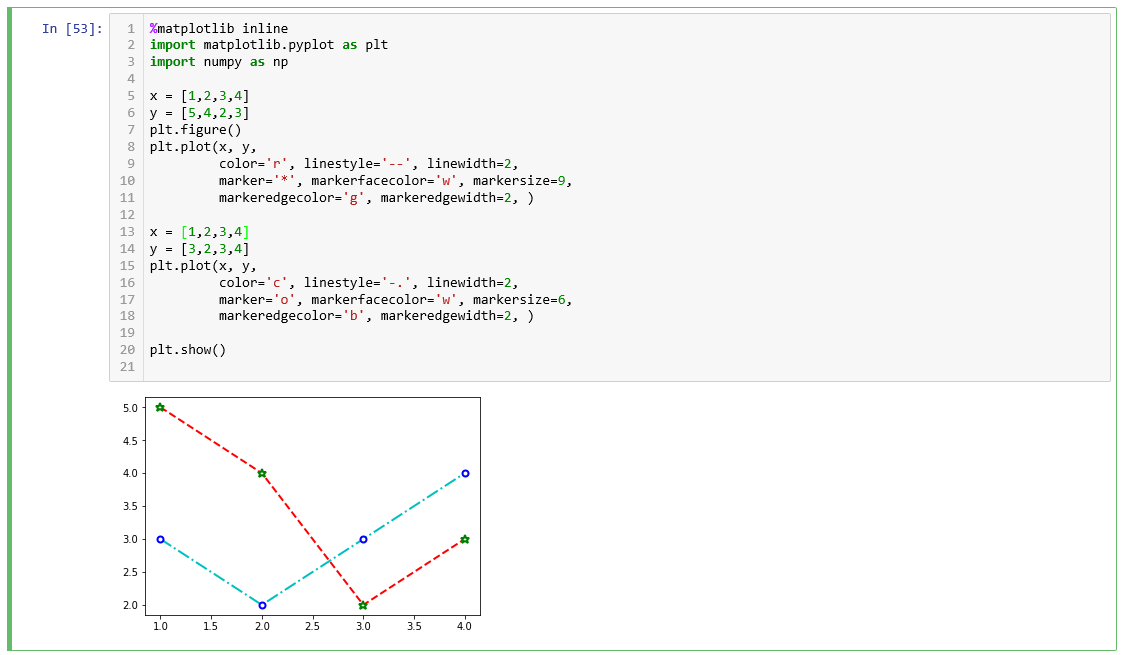
plt.plot(x, y,

color='c', linestyle='-.', linewidth=2,

marker='o', markerfacecolor='w', markersize=6,

markeredgecolor='b', markeredgewidth=2, )

plt.show()



## 1.4 坐标轴

**使用plt.xlim设置x坐标轴范围**：(0, 5)；

**使用plt.ylim设置y坐标轴范围**：(1, 6)；

**使用plt.xlabel设置x坐标轴名称**：’I am x’；

**使用plt.ylabel设置y坐标轴名称**：’I am y’；

**对x轴使用font\_axis字体**

**示例**

%matplotlib inline

import matplotlib.pyplot as plt

import numpy as np

x = [1,2,3,4]

y = [5,4,2,3]

plt.figure()

plt.plot(x, y,

color='r', linestyle='--', linewidth=2,

marker='\*', markerfacecolor='w', markersize=9,

markeredgecolor='g', markeredgewidth=2, )

x = [1,2,3,4]

y = [3,2,3,4]

plt.plot(x, y,

color='c', linestyle='-.', linewidth=2,

marker='o', markerfacecolor='w', markersize=6,

markeredgecolor='b', markeredgewidth=2, )

plt.xlim((0, 5))

plt.ylim((1, 6))

font\_axis = {

'family' : 'times new roman',

'weight' : 'normal',

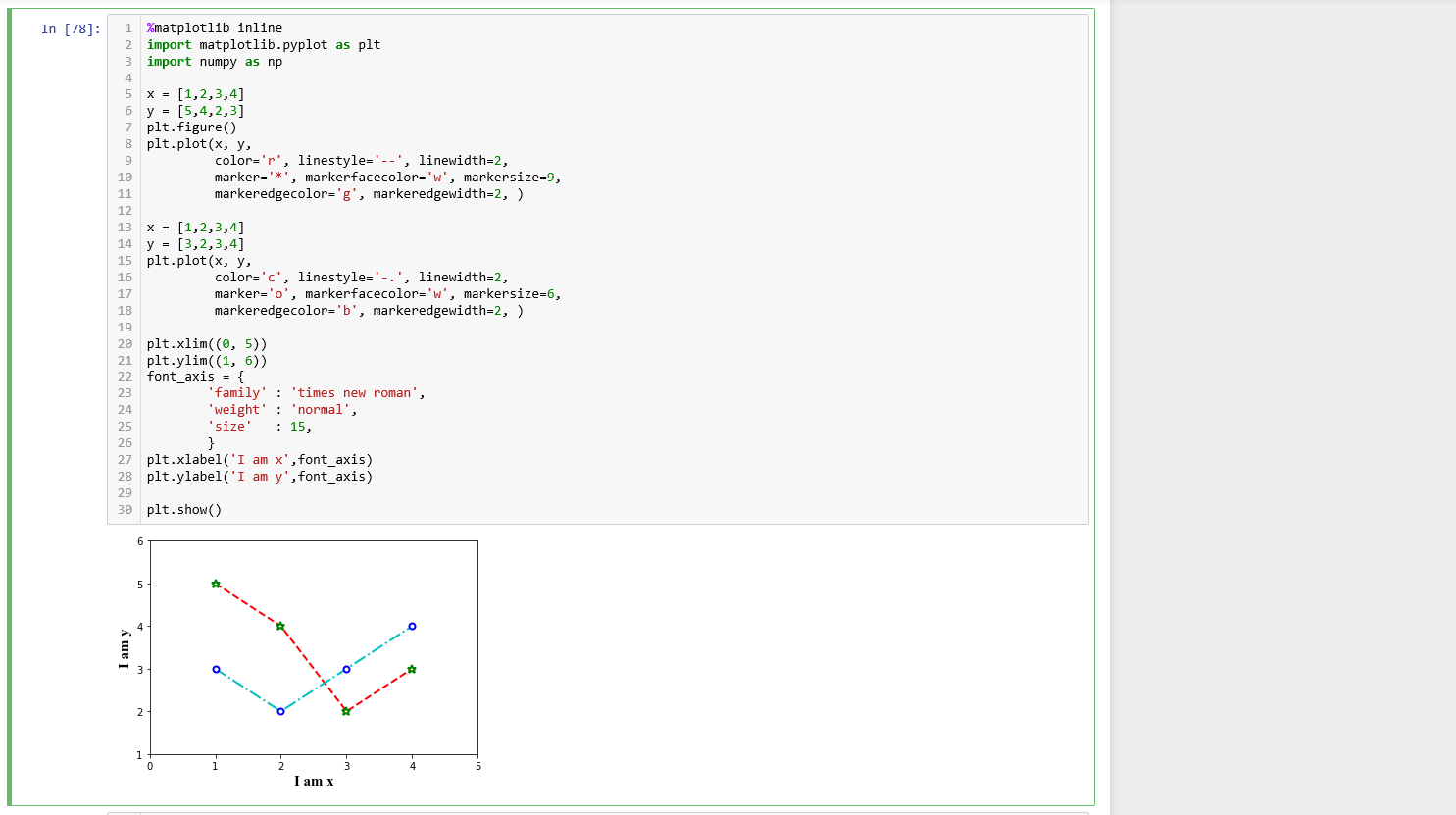
'size' : 15,

}

plt.xlabel('I am x',font\_axis)

plt.ylabel('I am y',font\_axis)

plt.show()

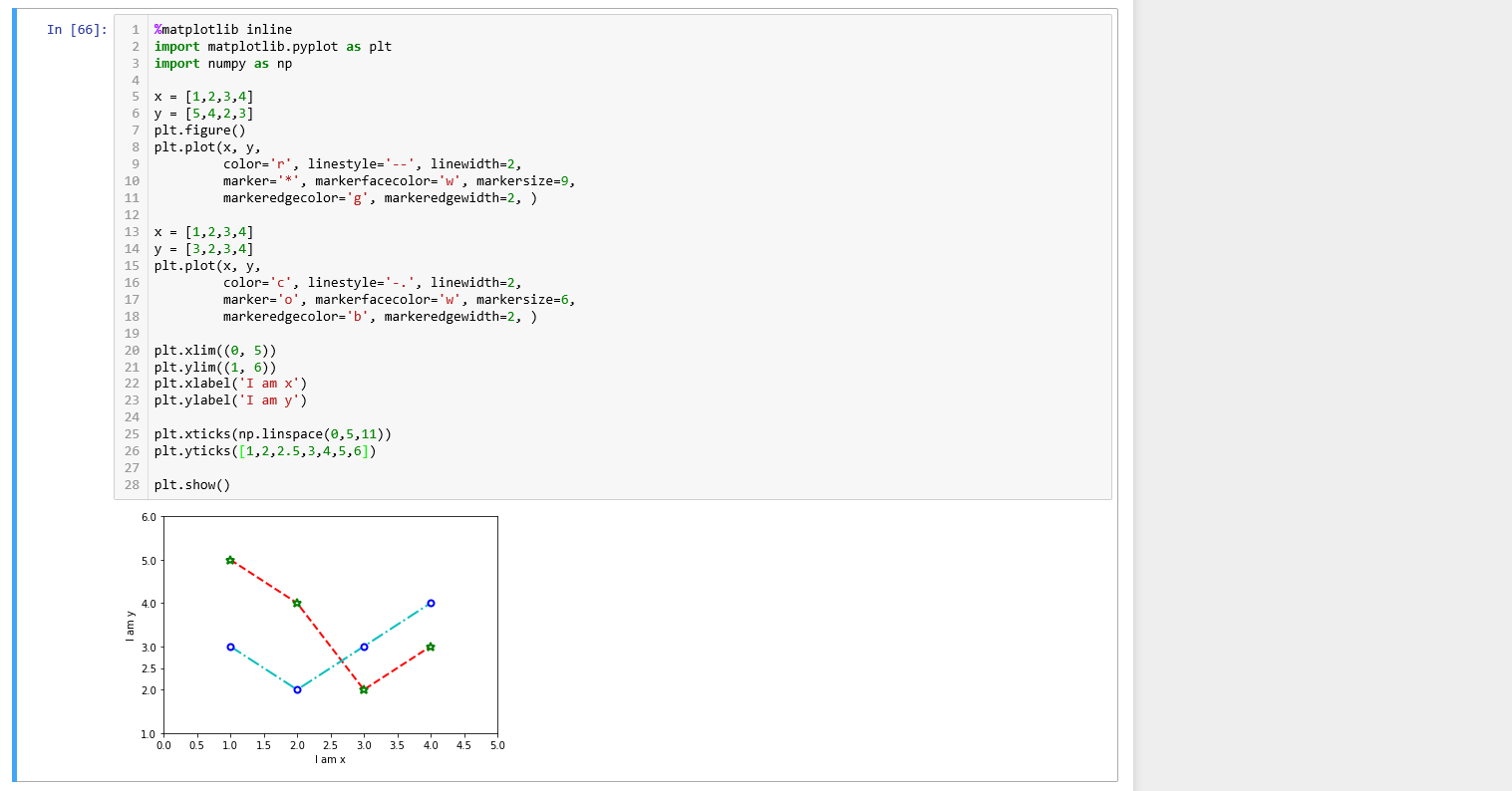


**使用plt.xticks设置x轴刻度**：范围是(0,5);个数是11.

**使用plt.yticks设置y轴刻度**：刻度为[1, 2, 2.5, 3, 4, 5, 6]；

plt.xticks(np.linspace(0,5,11))

plt.yticks([1,2,2.5,3,4,5,6])



**使用plt.gca获取当前坐标轴信息.**

**使用.spines设置边框：右侧边框；**

**使用.set\_color设置边框颜色**：默认白色；

**使用.spines设置边框**：上边框；

**使用.set\_color设置边框颜色**：默认白色；

%matplotlib inline

import matplotlib.pyplot as plt

import numpy as np

x = [1,2,3,4]

y = [5,4,2,3]

plt.figure()

plt.plot(x, y,

color='r', linestyle='--', linewidth=2,

marker='\*', markerfacecolor='w', markersize=9,

markeredgecolor='g', markeredgewidth=2, )

x = [1,2,3,4]

y = [3,2,3,4]

plt.plot(x, y,

color='c', linestyle='-.', linewidth=2,

marker='o', markerfacecolor='w', markersize=6,

markeredgecolor='b', markeredgewidth=2, )

plt.xlim((0, 5))

plt.ylim((1, 6))

plt.xlabel('I am x')

plt.ylabel('I am y')

plt.xticks(np.linspace(0,5,11))

plt.yticks([1,2,2.5,3,4,5,6])

ax = plt.gca()

ax.spines['right'].set\_color('none')

ax.spines['top'].set\_color('none')

plt.show()

## 1.5 设置grid

使用ax.grid(True, linestyle='-.')来设置grid线

使用ax.tick\_params(labelcolor='r', labelsize='medium', width=3)来设置grid顶端和标签

%matplotlib inline

import matplotlib.pyplot as plt

import numpy as np

x = [1,2,3,4]

y = [5,4,2,3]

plt.figure()

plt.plot(x, y,

color='r', linestyle='--', linewidth=2,

marker='\*', markerfacecolor='w', markersize=9,

markeredgecolor='g', markeredgewidth=2, )

x = [1,2,3,4]

y = [3,2,3,4]

plt.plot(x, y,

color='c', linestyle='-.', linewidth=2,

marker='o', markerfacecolor='w', markersize=6,

markeredgecolor='b', markeredgewidth=2, )

plt.xlabel('I am x')

plt.ylabel('I am y')

ax = plt.gca()

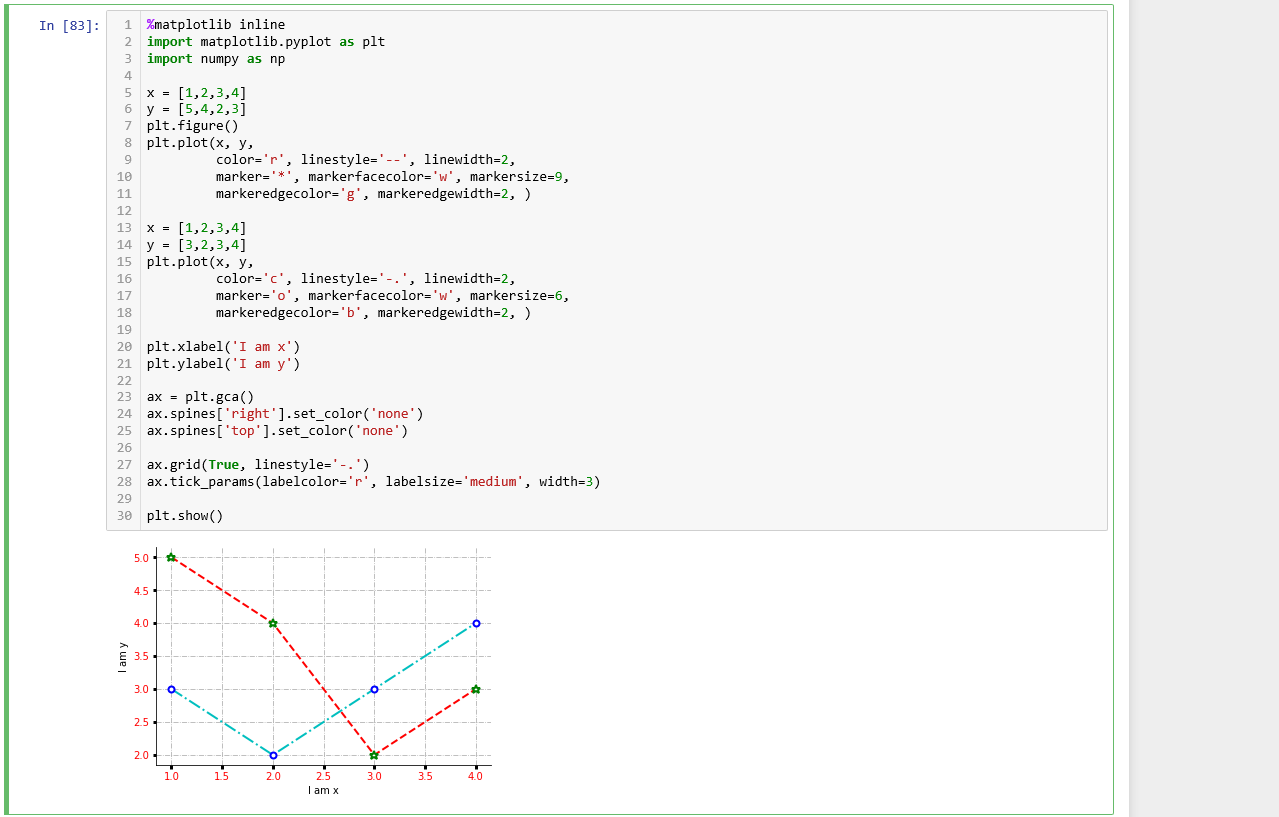
ax.spines['right'].set\_color('none')

ax.spines['top'].set\_color('none')

ax.grid(True, linestyle='-.')

ax.tick\_params(labelcolor='r', labelsize='medium', width=3)

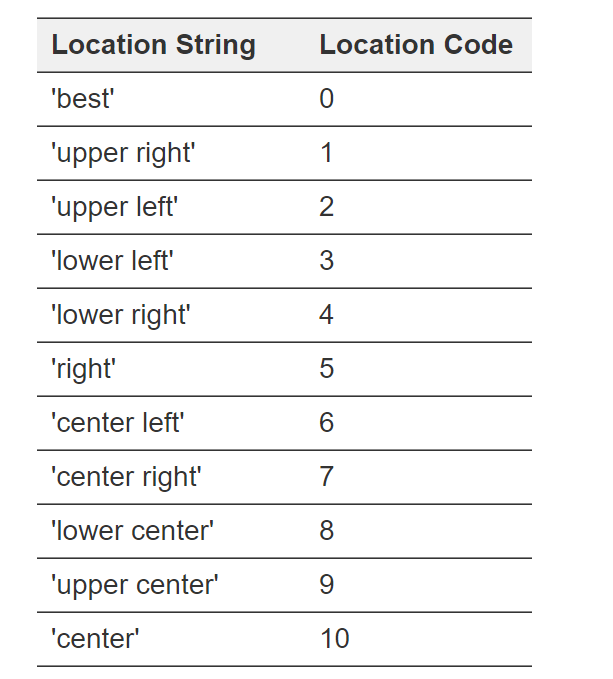
plt.show()



## 1.6 设置图例

使用plt.legend函数来设置图例。

参数loc来控制图例位置



%matplotlib inline

import matplotlib.pyplot as plt

import numpy as np

x = [1,2,3,4]

y = [5,4,2,3]

plt.figure()

plt.plot(x, y,

color='r', linestyle='--', linewidth=2,

marker='\*', markerfacecolor='w', markersize=9,

markeredgecolor='g', markeredgewidth=2, label='line 1')

x = [1,2,3,4]

y = [3,2,3,4]

plt.plot(x, y,

color='c', linestyle='-.', linewidth=2,

marker='o', markerfacecolor='w', markersize=6,

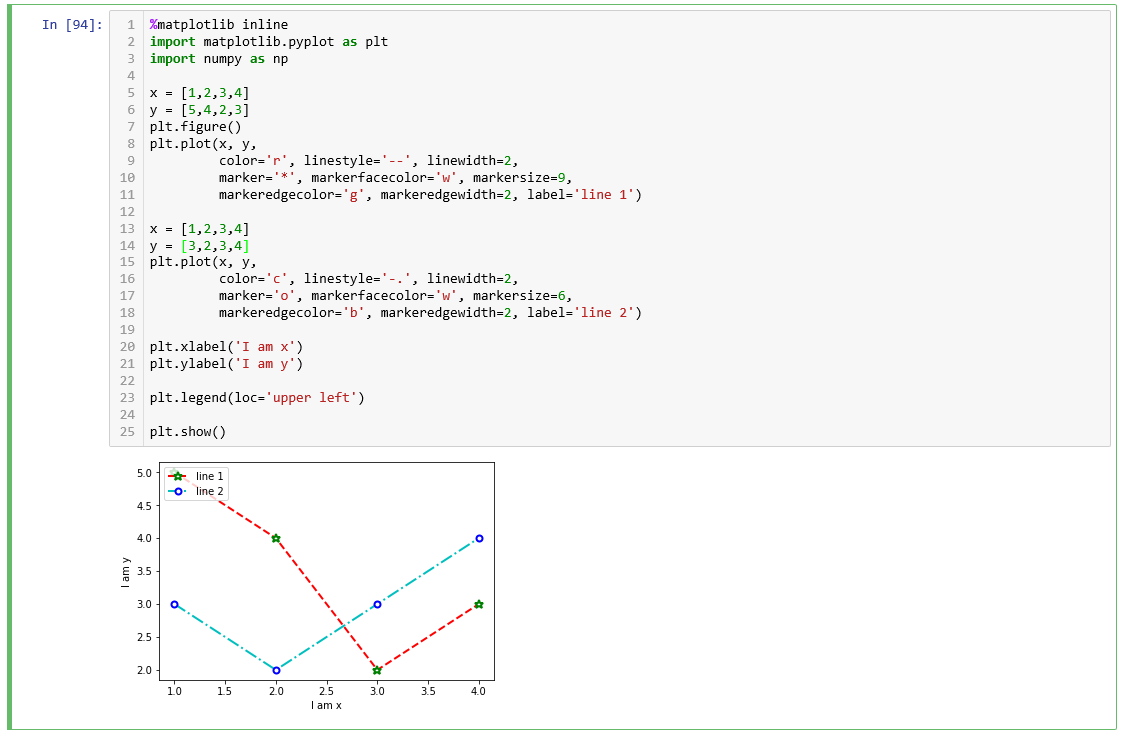
markeredgecolor='b', markeredgewidth=2, label='line 2')

plt.xlabel('I am x')

plt.ylabel('I am y')

plt.legend(loc='upper left')

plt.show()



参数handle来收集线段的列表。

参数labels指定显示的名称。

参数prop来指定字体。  
参数ncol指定列数

更多参数，可以参考<https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.legend.html#matplotlib.pyplot.legend>

%matplotlib inline

import matplotlib.pyplot as plt

import numpy as np

x = [1,2,3,4]

y = [5,4,2,3]

plt.figure()

l1, = plt.plot(x, y,

color='r', linestyle='--', linewidth=2,

marker='\*', markerfacecolor='w', markersize=9,

markeredgecolor='g', markeredgewidth=2, label='line 1')

x = [1,2,3,4]

y = [3,2,3,4]

l2, = plt.plot(x, y,

color='c', linestyle='-.', linewidth=2,

marker='o', markerfacecolor='w', markersize=6,

markeredgecolor='b', markeredgewidth=2, label='line 2')

plt.xlabel('I am x')

plt.ylabel('I am y')

font\_leg = {

'family' : 'times new roman',

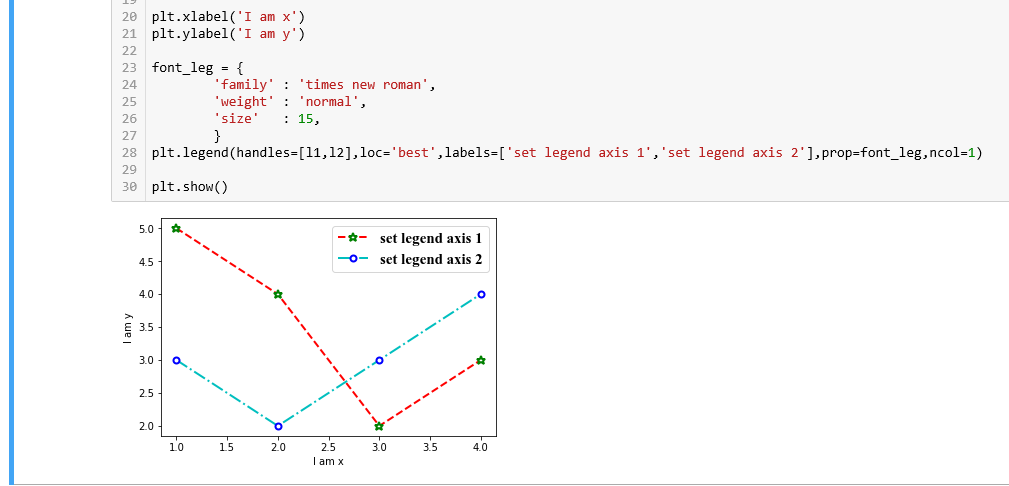
'weight' : 'normal',

'size' : 15,

}

plt.legend(handles=[l1,l2],loc='best',labels=['set legend axis 1','set legend axis 2'],prop=font\_leg,ncol=1)

plt.show()



## 1.7 设置窗口大小

1、使用plt.figure(figsize=(width, height))来设置。

%matplotlib inline

import matplotlib.pyplot as plt

import numpy as np

x = [1,2,3,4]

y = [5,4,2,3]

plt.figure(figsize=(3, 2))

plt.plot(x, y,

color='r', linestyle='--', linewidth=2,

marker='\*', markerfacecolor='w', markersize=9,

markeredgecolor='g', markeredgewidth=2, )

x = [1,2,3,4]

y = [3,2,3,4]

plt.plot(x, y,

color='c', linestyle='-.', linewidth=2,

marker='o', markerfacecolor='w', markersize=6,

markeredgecolor='b', markeredgewidth=2, )

plt.xlim((0, 5))

plt.ylim((1, 6))

plt.xlabel('I am x')

plt.ylabel('I am y')

plt.xticks(np.linspace(0,5,11))

plt.yticks([1,2,2.5,3,4,5,6])

ax = plt.gca()

ax.spines['right'].set\_color('none')

ax.spines['top'].set\_color('none')

plt.show()



# 2、画图种类

## 2.1 散点图

散点图常用于一些无规律的数据点展示，或者尚不明确数据之间的关系

\*\*函数：matplotlib.pyplot.scatter(x, y, s=None, c=None, marker=None, cmap=None,linewidth=None)，

主要参数：

x,y: 数据点的位置

s: 数据点标记的大小，可为常数或者一个列表

c: 为数据点标记的颜色，可以为单个颜色或者一个颜色列表

marker: 数据点标记的形状，markers参数候选项

cmap: colormap列表，当c为浮点型时才奏效

linewidth: 控制数据点边界线

更多参数，可以参考：<https://matplotlib.org/api/_as_gen/matplotlib.pyplot.scatter.html>

%matplotlib inline

import matplotlib.pyplot as plt

import numpy as np

n = 1024 # data size

X = np.random.normal(0, 1, n) # 每一个点的X值

Y = np.random.normal(0, 1, n) # 每一个点的Y值

T = np.arctan2(Y,X) # for color value

#T = np.arctan(X+Y)

plt.scatter(X, Y, s=75, c=T, alpha=.5)

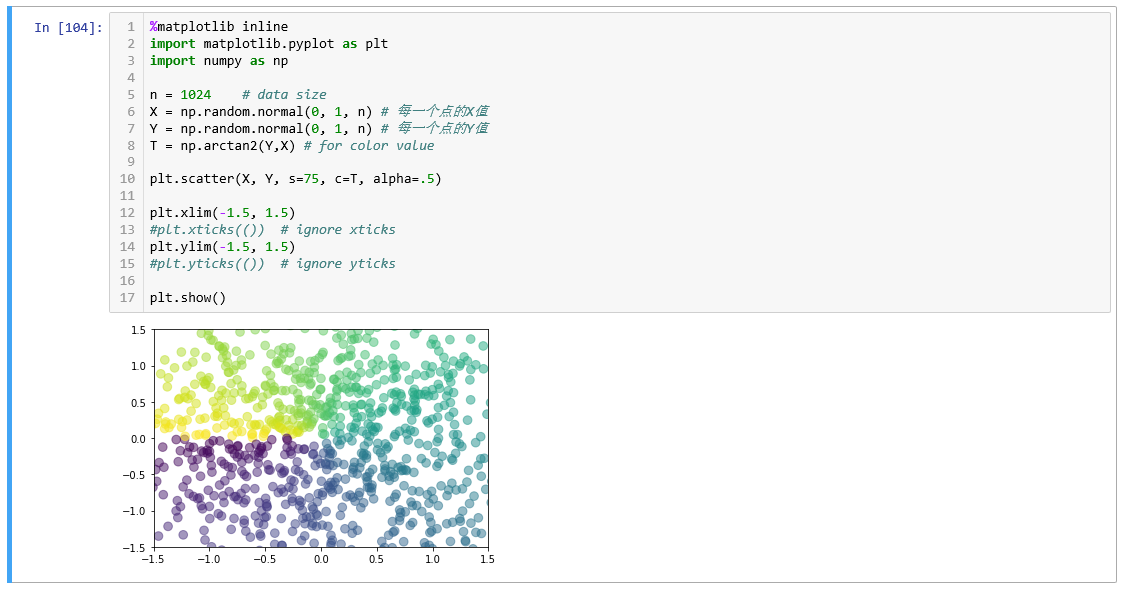
plt.xlim(-1.5, 1.5)

#plt.xticks(()) # ignore xticks

plt.ylim(-1.5, 1.5)

#plt.yticks(()) # ignore yticks

plt.show()



## 2.2 柱状图

绘制柱状图，我们主要用到bar()函数。只要将该函数理解透彻，我们就能绘制各种类型的柱状图。

我们先看下bar()的构造函数：bar(x，height， width，\*，align='center'，\*\*kwargs)

**x**包含所有柱子的下标的列表

**height**包含所有柱子的高度值的列表

**width**每个柱子的宽度。可以指定一个固定值，那么所有的柱子都是一样的宽。或者设置一个列表，这样可以分别对每个柱子设定不同的宽度。

**align**柱子对齐方式，有两个可选值：center和edge。center表示每根柱子是根据下标来对齐, edge则表示每根柱子全部以下标为起点，然后显示到下标的右边。如果不指定该参数，默认值是center。

其他可选参数有：

**color**每根柱子呈现的颜色。同样可指定一个颜色值，让所有柱子呈现同样颜色；或者指定带有不同颜色的列表，让不同柱子显示不同颜色。

**edgecolor**每根柱子边框的颜色。同样可指定一个颜色值，让所有柱子边框呈现同样颜色；或者指定带有不同颜色的列表，让不同柱子的边框显示不同颜色。

**linewidth**每根柱子的边框宽度。如果没有设置该参数，将使用默认宽度，默认是没有边框。

**tick\_label**每根柱子上显示的标签，默认是没有内容。

**xerr**每根柱子顶部在横轴方向的线段。如果指定一个固定值，所有柱子的线段将一直长；如果指定一个带有不同长度值的列表，那么柱子顶部的线段将呈现不同长度。

**yerr**每根柱子顶端在纵轴方向的线段。如果指定一个固定值，所有柱子的线段将一直长；如果指定一个带有不同长度值的列表，那么柱子顶部的线段将呈现不同长度。

**ecolor**设置 xerr 和 yerr 的线段的颜色。同样可以指定一个固定值或者一个列表。

**capsize**这个参数很有趣, 对xerr或者yerr的补充说明。一般为其设置一个整数，例如 10。如果你已经设置了  
yerr 参数，那么设置 capsize 参数，会在每跟柱子顶部线段上面的首尾部分增加两条垂直原来线段的线段。对 xerr 参数也是同样道理。可能看说明会觉得绕，如果你看下图就一目了然了。

import matplotlib.pyplot as plt

import numpy as np

# 创建一个点数为 5 x 4 的窗口, 并设置分辨率为 80像素/每英寸

plt.figure(figsize=(5, 4), dpi=80)

# 柱子总数

N = 6

# 包含每个柱子对应值的序列

values = [5, 32, 34, 20, 41, 50]

# 包含每个柱子下标的序列

index = np.arange(N)

# 柱子的宽度

width = 0.35

# 绘制柱状图, 每根柱子的颜色为紫罗兰色

p2 = plt.bar(index, values, width, label="rainfall", color="#87CEFA")

# 设置横轴标签

plt.xlabel('Months')

# 设置纵轴标签

plt.ylabel('rainfall (mm)')

# 添加标题

plt.title('Monthly average rainfall')

# 添加纵横轴的刻度

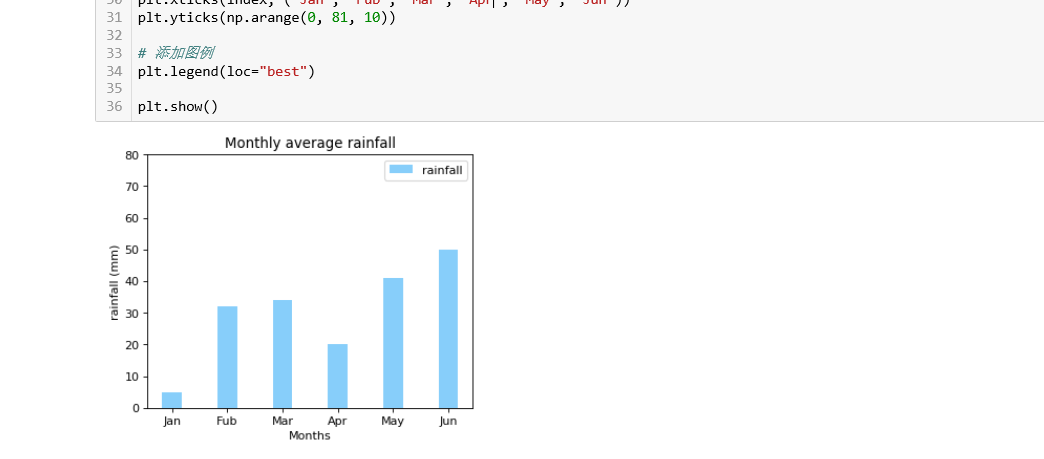
plt.xticks(index, ('Jan', 'Fub', 'Mar', 'Apr', 'May', 'Jun'))

plt.yticks(np.arange(0, 81, 10))

# 添加图例

plt.legend(loc="best")

plt.show()



**两个系列柱子：**

import matplotlib.pyplot as plt

import numpy as np

# 创建一个点数为 5 x 4 的窗口, 并设置分辨率为 80像素/每英寸

plt.figure(figsize=(5, 4), dpi=80)

# 柱子总数

N = 6

# 包含每个柱子对应值的序列

values = [5, 32, 34, 20, 41, 50]

values2 = [12, 15, 30, 46, 19, 26]

# 包含每个柱子下标的序列

index = np.arange(N)

# 柱子的宽度

width = 0.35

# 绘制柱状图, 每根柱子的颜色为紫罗兰色

p2 = plt.bar(index-0.175, values, width, label="rainfall", color="#87CEFA")

p2 = plt.bar(index+0.175, values2, width, label="snowfall", color="#ffCEFA")

# 设置横轴标签

plt.xlabel('Months')

# 设置纵轴标签

plt.ylabel('rainfall (mm)')

# 添加标题

plt.title('Monthly average rainfall')

# 添加纵横轴的刻度

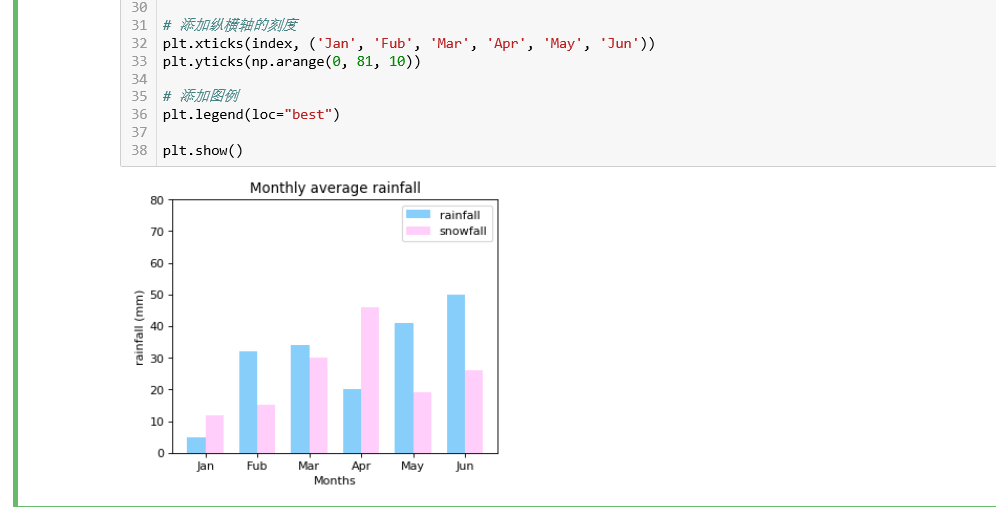
plt.xticks(index, ('Jan', 'Fub', 'Mar', 'Apr', 'May', 'Jun'))

plt.yticks(np.arange(0, 81, 10))

# 添加图例

plt.legend(loc="best")

plt.show()



**加入柱值**

import matplotlib.pyplot as plt

import numpy as np

# 创建一个点数为 5 x 4 的窗口, 并设置分辨率为 80像素/每英寸

plt.figure(figsize=(5, 4), dpi=80)

# 柱子总数

N = 6

# 包含每个柱子对应值的序列

values = [5, 32, 34, 20, 41, 50]

values2 = [12, 15, 30, 46, 19, 26]

# 包含每个柱子下标的序列

index = np.arange(N)

# 柱子的宽度

width = 0.35

# 绘制柱状图, 每根柱子的颜色为紫罗兰色

p1 = plt.bar(index-0.175, values, width, label="rainfall", color="#87CEFA")

p2 = plt.bar(index+0.175, values2, width, label="snowfall", color="#ffCEFA")

# 设置横轴标签

plt.xlabel('Months')

# 设置纵轴标签

plt.ylabel('rainfall (mm)')

# 添加标题

plt.title('Monthly average rainfall')

# 添加纵横轴的刻度

plt.xticks(index, ('Jan', 'Fub', 'Mar', 'Apr', 'May', 'Jun'))

plt.yticks(np.arange(0, 81, 10))

def autolabel(rects):

"""Attach a text label above each bar in \*rects\*, displaying its height."""

for rect in rects:

height = rect.get\_height()

plt.annotate('{}'.format(height),

xy=(rect.get\_x() + rect.get\_width() / 2, height),

xytext=(0, 3), # 3 points vertical offset

textcoords="offset points",

ha='center', va='bottom')

autolabel(p1)

autolabel(p2)

# 添加图例

plt.legend(loc="best")

plt.show()



**堆形柱状图**

import matplotlib.pyplot as plt

import numpy as np

# 创建一个点数为 5 x 4 的窗口, 并设置分辨率为 80像素/每英寸

plt.figure(figsize=(5, 4), dpi=80)

# 柱子总数

N = 6

# 包含每个柱子对应值的序列

values = [5, 32, 34, 20, 41, 50]

values2 = [12, 15, 30, 46, 19, 26]

# 包含每个柱子下标的序列

index = np.arange(N)

# 柱子的宽度

width = 0.35

# 绘制柱状图, 每根柱子的颜色为紫罗兰色

p1 = plt.bar(index, values, width, label="rainfall", color="#87CEFA")

p2 = plt.bar(index, values2, width, label="snowfall", color="#ffCEFA", bottom=values)

# 设置横轴标签

plt.xlabel('Months')

# 设置纵轴标签

plt.ylabel('rainfall (mm)')

# 添加标题

plt.title('Monthly average rainfall')

# 添加纵横轴的刻度

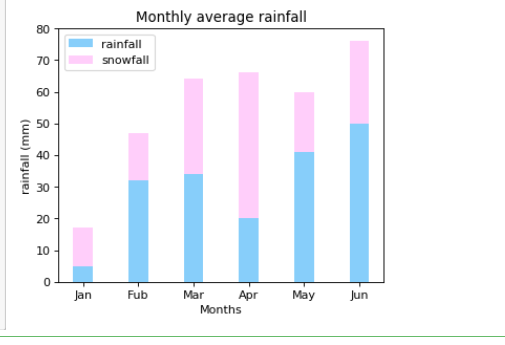
plt.xticks(index, ('Jan', 'Fub', 'Mar', 'Apr', 'May', 'Jun'))

plt.yticks(np.arange(0, 81, 10))

# 添加图例

plt.legend(loc="best")

plt.show()



## 2.3 饼状图

饼状图可以使用plt.pie()方法创建。

pie(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,

hold=None, data=None

)

**x** - 创建饼状图的数组

**explode** - array-like, optional, default: None；偏离每个楔形的半径的分数。若非None，则 数组长度 = len(x)，

**labels** - list, optional, default: None；为每个楔形添加标签

**color** - array-like, optional, default: None；若无，则用currently active cycle中的颜色添加

**autopct** - None (default), string, or function, optional；用于数值标记楔形的数值 / 字符串 / 函数，其中标签置于楔形内部；若None，则不现实楔形块占比数值，可以是整数( ' %d%%' )、浮点数( ' %2.3f %%' )、字符串( ' %s %%' )、函数。

**pctdistance** - float, optional, default: 0.6 ；描述autopct 值在楔形块的位置；当pctdistance = 0，所有autopct值均位于楔形块的 ‘ 尖端 ’ 位置，当为1，则位于楔形块的弧线位置，当大于1时，则会成比例远离尖端位置，

当 pctdistance = 0时，一个饼状图特例。

**shadow**- bool, optional, default: False；为饼状图画阴影(True)

**labeldistance** : float, optional, default: 1.1；描述label标签的在楔形块上的位置，确切地说，与圆心无关，而相对于楔形块的 “尖端 ” 的位置关系

**startangle** : float, optional, default: None；若不是None，则饼图的起点为 x 轴并逆时针旋转 angle 度

**radius** : float, optional, default: None；饼图的半径，若为None时，则默认为1

**counterclock** : bool, optional, default: True；指定分数方向，逆时针 (True) 或顺时针

**wedgeprops** : dict, optional, default: None；描述楔形边界线宽度值，参数形式``wedgeprops = {'linewidth': 3}``楔形边界线宽度为3

**textprops** : dict, optional, default: None；传递给文本对象的字典参数

**center** :  list of float, optional, default: (0, 0)；图标的中心为，默认(0,0)，也可以是两个标量的序列(sequence  of 2 scalars)

**frame** : bool, optional, default: False；若True，则绘制轴框架

**texts** : list；标签列表，class:`matplotlib.text.Text` instances实例.

**autotexts** : list；基于数字标签(numeric labels)的class:`~matplotlib.text.Text`实例列表，仅当参数\* autopct \*不是\* None \*时才会返回。

**示例**

import matplotlib.pyplot as plt

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

labels = ['娱乐','育儿','饮食','房贷','交通','其它']

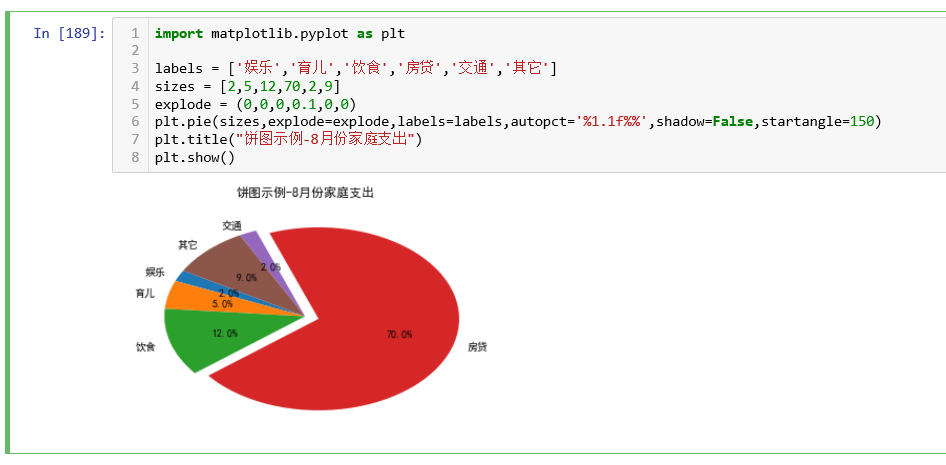
sizes = [2,5,12,70,2,9]

explode = (0,0,0,0.1,0,0)

plt.pie(sizes,explode=explode,labels=labels,autopct='%1.1f%%',shadow=False,startangle=150)

plt.title("饼图示例-8月份家庭支出")

plt.show()



拉伸比例成长宽相等

import matplotlib.pyplot as plt

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

labels = ['娱乐','育儿','饮食','房贷','交通','其它']

sizes = [2,5,12,70,2,9]

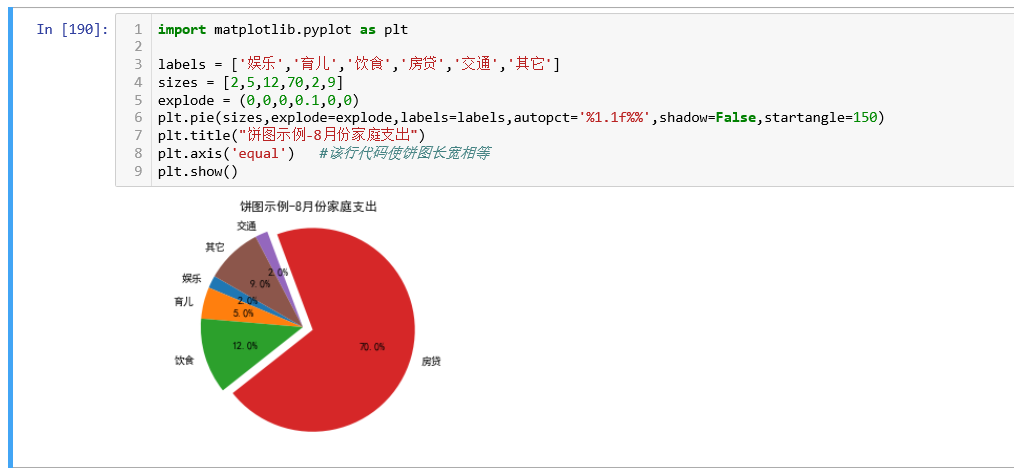
explode = (0,0,0,0.1,0,0)

plt.pie(sizes,explode=explode,labels=labels,autopct='%1.1f%%',shadow=False,startangle=150)

plt.title("饼图示例-8月份家庭支出")

plt.axis('equal') #该行代码使饼图长宽相等

plt.show()



# 3、子图subplot的使用

## 1、指定划分方式和位置进行绘图。

示例

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6), dpi=80)

t=np.arange(0.0,2.01,0.01)

s=np.round(np.sin(t\*np.pi),2)

#s=np.sin(t\*np.pi)

plt.subplot(2,2,1) #要生成两行两列，这是第一个图plt.subplot('行','列','编号')

plt.plot(t,s,'b--')

plt.ylabel('y1')

plt.subplot(2,2,2) #两行两列,这是第二个图

plt.plot(2\*t,s,'r--')

plt.ylabel('y2')

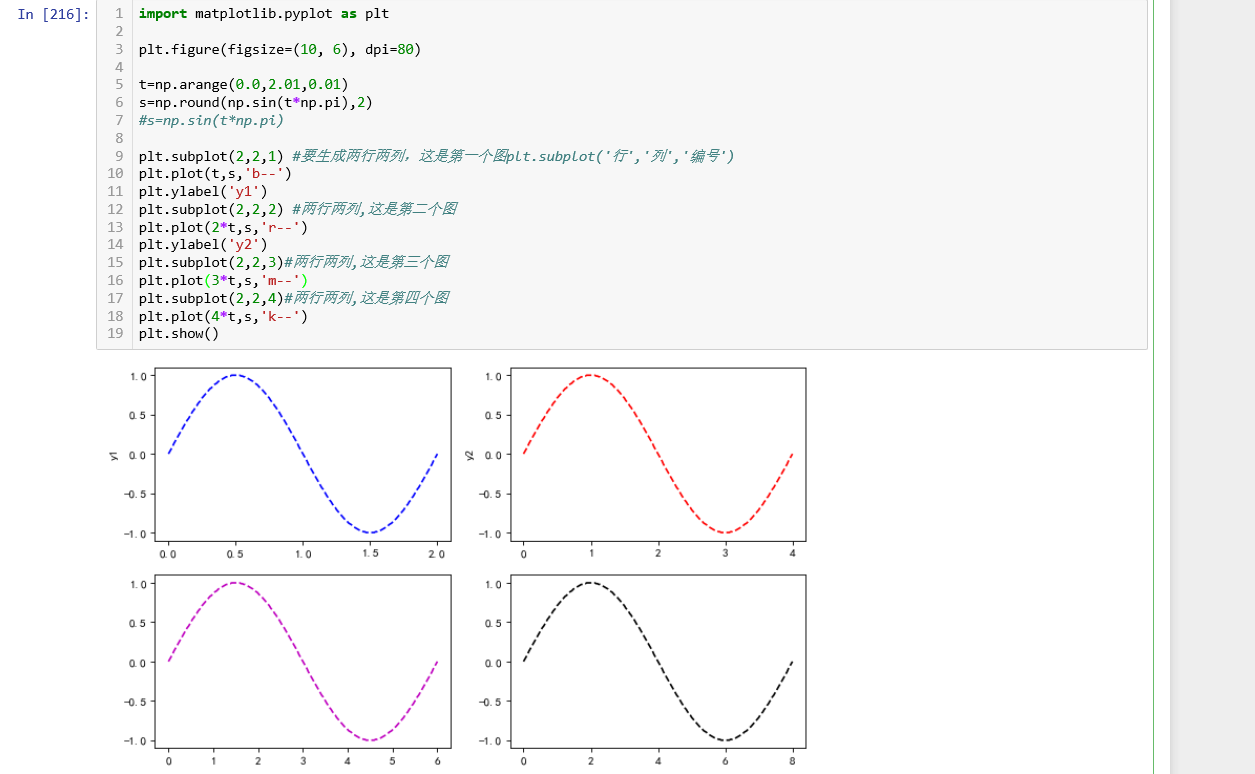
plt.subplot(2,2,3)#两行两列,这是第三个图

plt.plot(3\*t,s,'m--')

plt.subplot(2,2,4)#两行两列,这是第四个图

plt.plot(4\*t,s,'k--')

plt.show()



## 2、使用编号

import matplotlib.pyplot as plt

import numpy as np

t=np.arange(0.0,2.0,0.1)

s=np.sin(t\*np.pi)

figure,ax=plt.subplots(2,2)

ax[0][0].plot(t,s,'b--')

ax[0][1].plot(t\*2,s,'b--')

ax[0][0].plot(t,s,'b--')

ax[0][0].set\_ylabel('y1')

ax[0][1].plot(2\*t,s,'r--')

ax[0][1].set\_ylabel('y2')

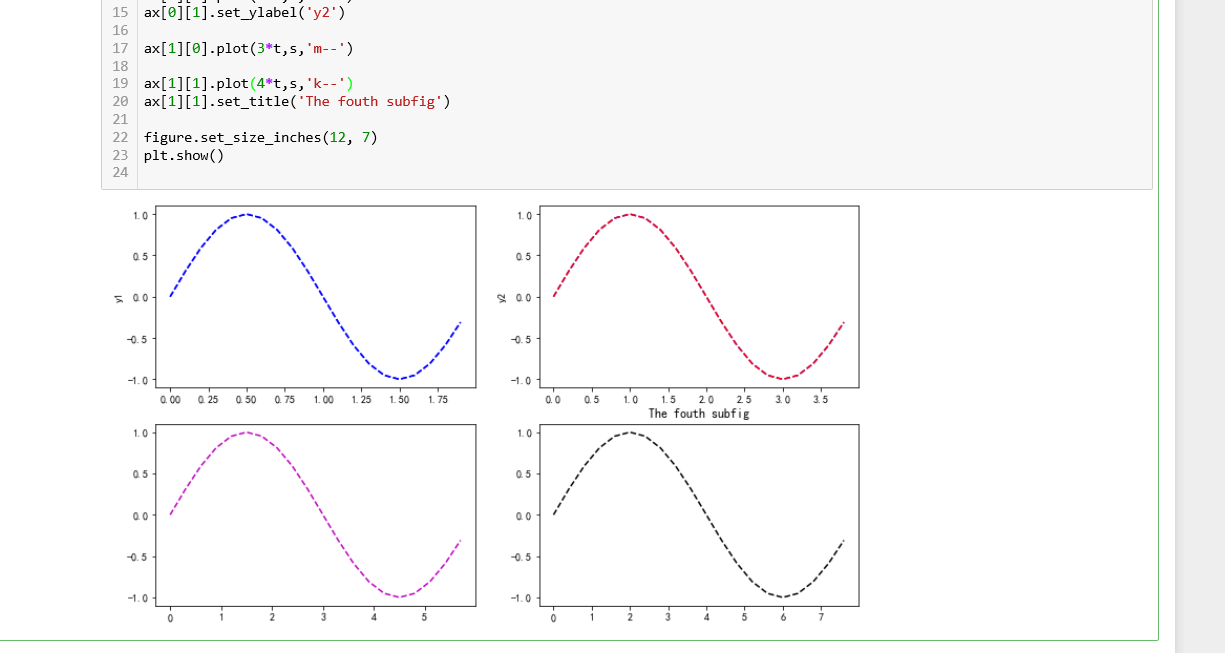
ax[1][0].plot(3\*t,s,'m--')

ax[1][1].plot(4\*t,s,'k--')

ax[1][1].set\_title('The fouth subfig')

figure.set\_size\_inches(12, 7)

plt.show()



# 4、使用Pandas数据绘图

## 4.1 简单绘图

import pandas as pd

import matplotlib.pyplot as plt

pdata = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',

'Kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],

'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],

'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],

'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}

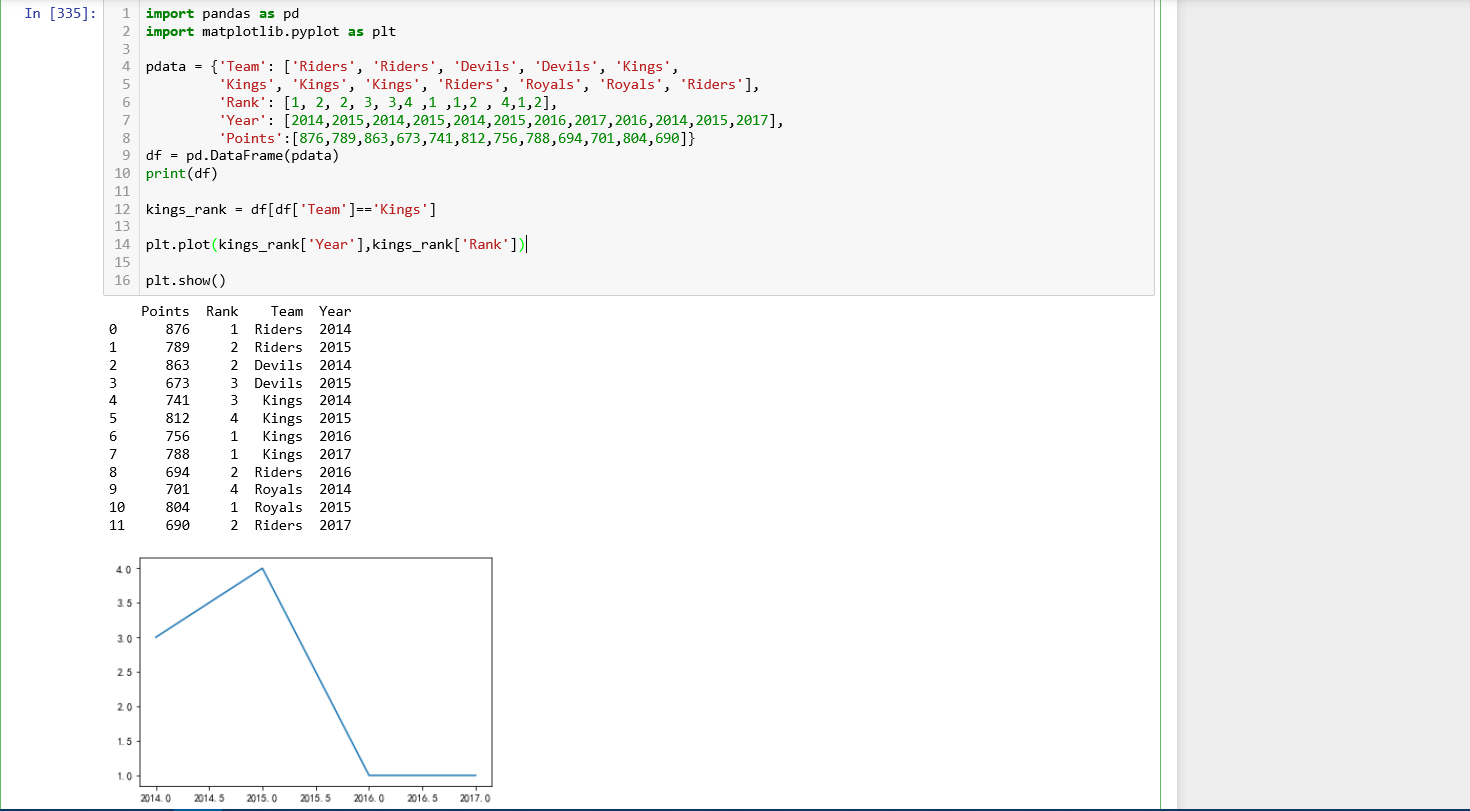
df = pd.DataFrame(pdata)

print(df)

kings\_rank = df[df['Team']=='Kings']

plt.plot(kings\_rank['Year'],kings\_rank['Rank'])

plt.show()



## 4.2 分组绘图

示例1：

plt.figure(figsize=(6, 4), dpi=80)

pdata = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',

'Kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders', 'Devils', 'Devils', 'Royals', 'Royals'],

'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2, 4, 3,3,4],

'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017,2016,2017,2016,2017],

'Points':[876,789,863,673,741,812,756,788,694,701,804,690,645,679,678,669]}

df = pd.DataFrame(pdata)

print(df)

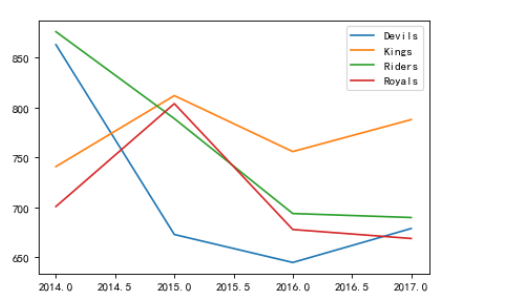
for key, grp in df.groupby(['Team']):

plt.plot(grp['Year'], grp['Points'], label=key)

#plt.plot(grp['Year'], grp['Points'], label=key, markersize=10, markerfacecolor='r',marker='\*')

plt.legend()

plt.show()



示例2：

plt.figure(figsize=(6, 4), dpi=80)

pdata = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',

'Kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders', 'Devils', 'Devils', 'Royals', 'Royals'],

'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2, 4, 3,3,4],

'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017,2016,2017,2016,2017],

'Points':[876,789,863,673,741,812,756,788,694,701,804,690,645,679,678,669]}

df = pd.DataFrame(pdata)

print(df)

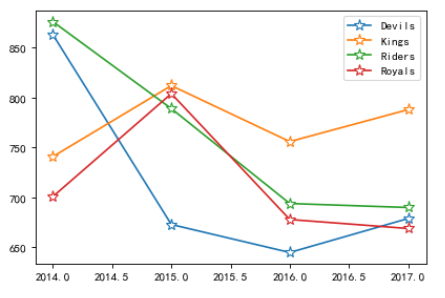
for key, grp in df.groupby(['Team']):

#plt.plot(grp['Year'], grp['Points'], label=key)

plt.plot(grp['Year'], grp['Points'], label=key, markersize=10, markerfacecolor='w',marker='\*')

plt.legend()

plt.show()



## 4.3 多子图分组绘图

示例

plt.figure(figsize=(12, 8), dpi=80)

pdata = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',

'Kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders', 'Devils', 'Devils', 'Royals', 'Royals'],

'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2, 4, 3,3,4],

'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017,2016,2017,2016,2017],

'Points':[876,789,863,673,741,812,756,788,694,701,804,690,645,679,678,669]}

df = pd.DataFrame(pdata)

print(df)

marker = ['\*','^','<','>']

x\_axis = ['Year','Year','Points','Rank']

y\_axis = ['Points','Rank','Rank','Year']

for i in range(4):

plt.subplot(2,2,i+1)

inx = 0

for key, grp in df.groupby(['Team']):

#plt.plot(grp['Year'], grp['Points'], label=key)

grp = grp.sort\_values(by=[x\_axis[i]])

plt.plot(grp[x\_axis[i]], grp[y\_axis[i]], label=key, markersize=5, markerfacecolor='w',marker=marker[inx])

inx+=1

plt.legend()

plt.show()

