## 基础知识

Artist类是所有图像元件的抽象父类。

figure对象是图底板，一个图底板上可以有多张图。

axes…对象是每张图纸，通过figure.add\_subplot可以添加图纸，获得axessubplot对象。

axe=fg.add\_subplot(312,[参数])（表示这个图板共3行1列，共3幅画。现在获取的是第2幅对象。）

axis为坐标轴对象

AxesSubplot对象可以调用所有关画图的一些方法和对象，如plot、set\_title、set\_ylabels、set\_title等等方法。plt可以调用的有关画图和设置的方法，他都可以调用（猜想）。注意plt的方法在axes..类中很多是set\_...，如plt.title在axes..类中是set\_title。

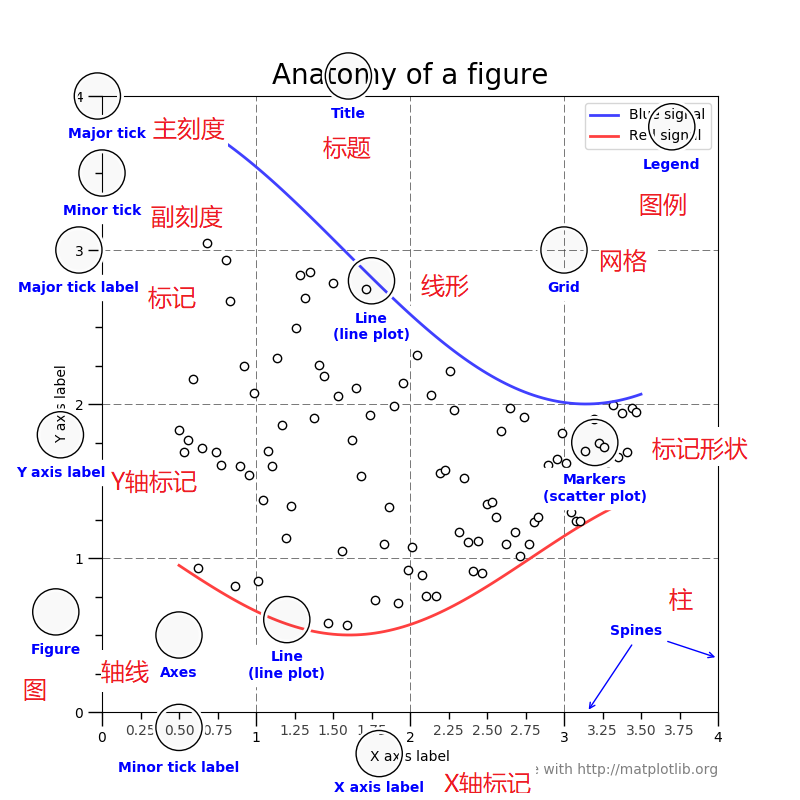
plt.gcf()：获取当前figure对象。

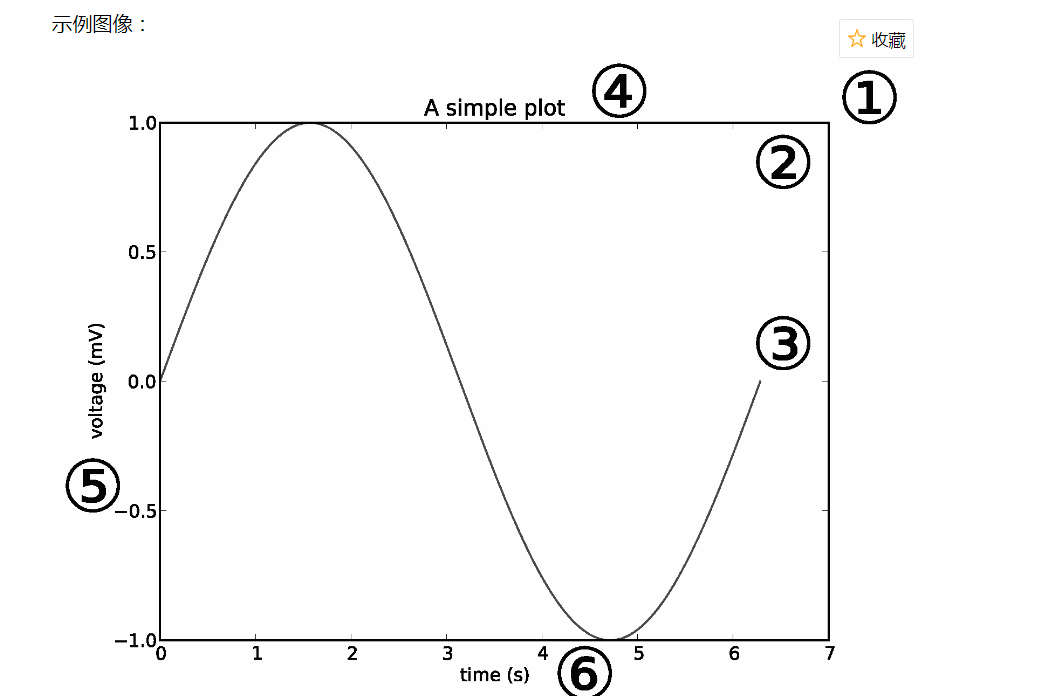
plt.gca():获取当前axes对象。

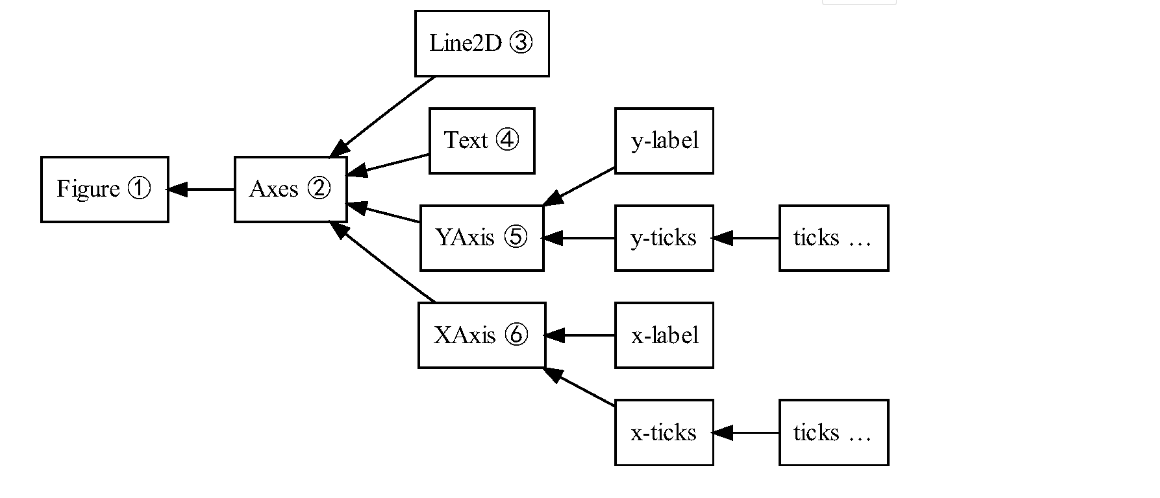
plt.clf(): 清理当前figure  
plt.cla(): 清理当前axes  
plt.close(): 一副figure直到显式的调用close()时才会释放她所占用的资源;

当我们使用plot时，会自动调用gca来获得当前axes，gca会自动调用gcf来获得当前figure，如果没有创建figure，系统会自动为我们创建一个figure，并建立一个subsplot（111）

scale是坐标，有直线坐标、对数坐标、symlog坐标







上图为figure与axes和

1：线图　　plot（）单线段图

　　2：多个线图　　subplot（）Multiple axes (i.e. subplots) are created with the [subplot()](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.subplot.html#matplotlib.pyplot.subplot) function。

　　3：图像　　imshow（）显示图像尺寸

　　4：轮廓和颜色　　pcolormesh（）函数可以对二维数组进行着色表示，即使水平维度是不均匀间隔的　　contour（）函数是表示相同数据的另一种方式

　　5：直方图　　hist（）函数自动生成直方图并返回bin计数或概率。

　　6：路径　　matplotlib.path添加任意路径

　　7：三维绘图　　mplot3d支持简单的3D图形，包括表面、线框、散布和条形图。

　　8：流线图　　streamplot()流线图（）函数绘制了矢量场的流线。除了简单地绘制流线之外，它还可以将流线的颜色和/或线宽映射到单独的参数，例如矢量场的速度或局部强度。

　　9：Ellipses　　地面追踪显示

　　10：条形图　　bar（）函数来创建条形图，其中包括诸如错误条的自定义

　　11：饼图　　Pie（）函数允许您创建饼图。可选的特征包括自动标记区域的百分比，从馅饼的中心爆炸一个或多个楔子，以及阴影效果。

　　12：桌表　　Table（）函数将文本表添加到轴

　　13：散点图　　scatter（）函数用可选的大小和颜色参数进行散点图

　　14：图形用户界面控件　　MatPuttLB具有基本的GUI小部件，这些图形小部件与您正在使用的图形用户界面无关，允许您编写跨GUI图形和小部件

15：填充曲线　　fill（）函数可以绘制填充曲线和多边形

## 1.导入

from matplotlib import pyplot as plt

## 通用方法：

#### 设置各轴名称xlabel/ylabel

plt.ylabel(‘yaxis’，\*\*kwargs) 设置y轴名称

plt.xlabel(‘xaxis’)设置x轴名称

由于默认不支持中文，可以通过

字典，里面存放着参数，其实和\*\*kwargs一样。

override = {

'fontsize' : 'small',

'verticalalignment' : 'center',

'horizontalalignment' : 'right',

'rotation'='vertical' : }

字典里面的键值可以参考text中的参数。

import matplotlib

# fname 为 你下载的字体库路径，注意 SimHei.ttf 字体的路径

zhfont1 = matplotlib.font\_manager.FontProperties(fname="SimHei.ttf")

来加载中文字体。

也可以使用系统自带字体。获取系统自带字体：

a=[font **for** font **in** matplotlib.font\_manager.fontManager.ttflist]

并在方法中指定：

plt.ylabel(‘y轴’，fontsize=..，fontproperties=zhfont1)

#### 设置画图标题title

plt.title("菜鸟教程 - 测试", fontproperties=zhfont1)

#### 设置坐标轴显示范围axis

如果不设置，默认为所画数值的最小值和最大值。

plt.axis([xmin,xmax,ymin,ymax])

或者

plt.xlim(xmin,xmax) :参数不必填完。可以指定关键字参数。xmax=

plt.ylim(ymin,ymax)

如果想获得当前的xin和xmax

xmin,xmax=plt.xlim()

#### 在指定位置显示文字text

plt.text(x,y,’文字’) 在坐标x、y处显示文字。

输入特殊符号支持使用 laTex 语法，用 $<some Tex code>$ 隔开。

plt.text(60, .025, r'$\mu=100,\ \sigma=15$') 分别是μ和σ

 LaTeX语法：

#### 注释annotate

被axes对象来调用。

使用 annotate 函数进行注释，annotate 主要有两个参数：

-第一个参数为要注释的文字。

- xy ：注释位置

- xytext ：注释文字位置

ax = plt.subplot(111)

t = np.arange(0.0, 5.0, 0.01)

s = np.cos(2\*np.pi\*t)

line, = plt.plot(t, s, lw=2)

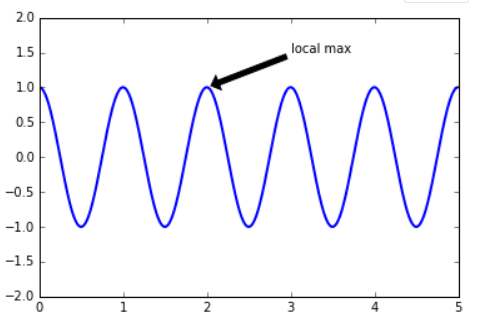
plt.annotate('local max', xy=(2, 1), xytext=(3, 1.5),

arrowprops=dict(facecolor='black', shrink=0.05),

)

plt.ylim(-2,2)

plt.show()



##### 参数：

###### **s：**

**注释的内容，str**

###### **xy:**

**需要注释的点的位置，(x,y)**

###### **xytext:**

**注释的位置（x,y）坐标点**

###### **xycoords** ：

注释点坐标的参考坐标系，可以为下表的字符

###### **textcoords：**

**注释位置坐标的参考坐标系可以为下表的字符**

| **Property** | **Description** |
| --- | --- |
| ‘figure points’ | points from the lower left of the figure  图版（figure）的左下角为原点 |
| ‘figure pixels’ | pixels from the lower left of the figure  图板（figure）的左下角像素为原点 |
| ‘figure fraction’ | fraction of figure from lower left |
| ‘axes points’ | points from lower left corner of axes  图纸（axes）的左下角为原点 |
| ‘axes pixels’ | pixels from lower left corner of axes |
| ‘axes fraction’ | fraction of axes from lower left |
| ‘data’ | 使用坐标系 |
| ‘polar’ | *(theta,r)* if not native ‘data’ coordinates |

textcoords还可以设置相对坐标，相对于x、y标记点的偏移量。和xytext结合使用。

此时，xytext表示的就是相对于xy的偏移量。

|  |  |
| --- | --- |
| ‘offset points’ | offset (in points) from the *xy* value  以坐标点为偏移 |
| ‘offset pixels’ | offset (in pixels) from the *xy* value  以像素为偏移 |

###### **arrowprops：**

**箭头道具**

**字典格式，里面为箭头的一些设置参数。**

**绘制从xytext到xy之间的箭头**

**如果省略，绘制默认箭头。**

如果字典中不包含’arrowstyle’键，则可以使用下列的键来定制：

| **Key** | **Description** |
| --- | --- |
| width | the width of the arrow in points  箭头宽度 |
| headwidth | the width of the base of the arrow head in points  箭头头部宽度 |
| headlength | the length of the arrow head in points  箭头头部长度 |
| shrink | fraction of total length to ‘shrink’ from both ends，0-1的小数  ‘缩水’的意思，即箭头两端都向内缩进（缩短）多少，注意，如果缩的过短，小于（箭头的头）的长度后，箭头的头部会不变，并顶点（顶着xy点）显示，只对箭头尾部进行缩进。 |
| ? | any key to [matplotlib.patches.FancyArrowPatch](https://matplotlib.org/api/patches_api.html#matplotlib.patches.FancyArrowPatch)  该类的其他允许的key |

如果字典包含‘arrowstyle’键，设置箭头格式。则上面的键是被隐藏的，arrowstyle对应的值为：

| **value** | **默认设置** |
| --- | --- |
| '-' | None |
| '->' | head\_length=0.4,head\_width=0.2 |
| '-[' | widthB=1.0,lengthB=0.2,angleB=None |
| '|-|' | widthA=1.0,widthB=1.0 |
| '-|>' | head\_length=0.4,head\_width=0.2 |
| '<-' | head\_length=0.4,head\_width=0.2 |
| '<->' | head\_length=0.4,head\_width=0.2 |
| '<|-' | head\_length=0.4,head\_width=0.2 |
| '<|-|>' | head\_length=0.4,head\_width=0.2 |
| 'fancy' | head\_length=0.4,head\_width=0.4,tail\_width=0.4 |
| 'simple' | head\_length=0.5,head\_width=0.5,tail\_width=0.2 |
| 'wedge' | tail\_width=0.3,shrink\_factor=0.5 |

[FancyArrowPatch](https://matplotlib.org/api/patches_api.html#matplotlib.patches.FancyArrowPatch)一些有效的key:

| **Key** | **Description** |
| --- | --- |
| arrowstyle | the arrow style |
| connectionstyle | the connection style |
| relpos | default is (0.5, 0.5) |
| patchA | default is bounding box of the text |
| patchB | default is None |
| shrinkA | default is 2 points |
| shrinkB | default is 2 points |
| mutation\_scale | default is text size (in points) |
| mutation\_aspect | default is 1. |
| ? | any key for [matplotlib.patches.PathPatch](https://matplotlib.org/api/patches_api.html#matplotlib.patches.PathPatch) |

###### **annotation\_clip** :

布尔值，控制注释在图像外部的可见性。

如果为True，则注释在图纸内部才显示。

默认为None，即：当xycoords为data时，其被设置为True，其余情况设置为False。

#### 指定x、y轴坐标点ticks

plt.sticks()/plt.yticks()

xticks( [-np.pi, -np.pi/2, 0, np.pi/2, np.pi])

yticks([-1, 0, +1])

#使用latex语法，将pi换为π

xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],

[r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

yticks([-1, 0, +1],

[r'$-1$', r'$0$', r'$+1$'])

loc,labels=xticks()

获取坐标的设置：loc为坐标点的位置，labels为该点显示的文本的列表形式

#### 将坐标轴移至中间：

通过splines来设置。

坐标轴线和上面的记号连在一起就形成了脊柱（Spines，一条线段上有一系列的凸起，是不是很像脊柱骨啊~），它记录了数据区域的范围。它们可以放在任意位置，不过至今为止，我们都把它放在图的四边。

实际上每幅图有四条脊柱（上下左右），为了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。

ax = plt.gca() #获取当前axes对象，即图纸对象。

ax.spines['right'].set\_color('none') #将右侧轴颜色设为无色

ax.spines['top'].set\_color('none') #上侧轴颜色设为无色

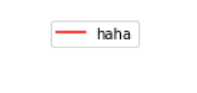
ax.xaxis.set\_ticks\_position('bottom') #设置x轴显示的坐标点在下轴上

ax.spines['bottom'].set\_position(('data',0))

ax.yaxis.set\_ticks\_position('left')#设置y轴显示的坐标点在左轴上

ax.spines['left'].set\_position(('data',0))

#### legend添加图例：



在一个axe对象上添加图例。

三种调用形式：

legend()

legend(labels)

legend(handles, labels)

1.legend()

不使用任何参数，会自动检测并添加图例。

需要在绘图时，设置label属性。即plot(x,y,label=’’)

显示图例时，就是以此label为该图形命名。如果没有label，则不会绘制图例。

或者通过plot方法的返回值matplotlib.lines.Line2D object实例，通过该实例set\_label方法来补充设置label，然后再调用legend，即可生成图例。

2.legend(labels)

ax.plot([1, 2, 3])

ax.legend(['A simple line'])

如果ax实例中有多个点或者多条线，可以使用一个列表存放str，按照顺序，一个点/线对应一个str。

3.legend(handles,labels)

legend((line1, line2, line3), ('label1', 'label2', 'label3'))

通过line实例来指定。

plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="cosine")

plt.plot(X, S, color="red", linewidth=2.5, linestyle="-", label="sine")

plt.legend(loc='upper left', frameon=False)

第一个图例：x、c为图例的坐标点

其他参数：

**loc** : int or string or pair of floats, default: [rcParams["legend.loc"]](https://matplotlib.org/tutorials/introductory/customizing.html#matplotlib-rcparams) ('best' for axes, 'upper right' for figures)

| **Location String** | **Location Code** |
| --- | --- |
| 'best' | 0 |
| 'upper right' | 1 |
| 'upper left' | 2 |
| 'lower left' | 3 |
| 'lower right' | 4 |
| 'right' | 5 |
| 'center left' | 6 |
| 'center right' | 7 |
| 'lower center' | 8 |
| 'upper center' | 9 |
| 'center' | 10 |

Alternatively can be a 2-tuple giving x, y of the lower-left corner of the legend in axes coordinates (in which case bbox\_to\_anchor will be ignored).

如果绘图时数据量很大，则‘best’选项会导致速度变慢，最好指定位置。

其他参数如字体、背景色、偏移、阴影、title、边框、行距、列距、正面色、边缘颜色、透明度、文字与线条的先后顺序、图例标记与原始标记的相对大小等等设置请看：

<https://matplotlib.org/api/_as_gen/matplotlib.pyplot.legend.html#matplotlib.pyplot.legend>

#### 细节的完善

1.设置x、y ticks的背景色及字体

for label in ax.get\_xticklabels() + ax.get\_yticklabels():

label.set\_fontsize(16)

label.set\_bbox(dict(facecolor='white', edgecolor='None', alpha=0.65 ))

bbox指的是字体周围的方框。

## figures

figure相当于一个图板，编号从1开始。

### 1.参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 默认值 | 参数描述 |
| num | 1 | figure的编号，可省略 |
| figsize | figure.figsize | 2元元组，figure的大小（英寸），宽和高 |
| dpi | figure.dpi | 分辨率，以每英寸多少点为单位 |
| facecolor | figure.facecolor | 画板中的背景色 |
| edgecolor | figure.edgecolor | 画板边框的颜色 |
| frameon | True | 布尔值，是否绘制画板边框 |

### 2.关闭figure

plt.close():关闭当前figure

plt.close(5):关闭编号为5的figure

plt.close(fig):关闭实例fig

plt.close([1,5,3]):批量关闭（是否为列表，不肯定，但是可以批量关闭。）

### 3.set\_....

可以通过 set\_性质 来设置一些figure的文本等性质

## pyplot

### 2.plt.show()

显示做出的图形。

在jupyter中不需要，

使用

%matplotlib notebook 插入到notebook

%matplotlib inline 自动在行内显示

### 3.plt.plot

根据所给出x,y的点或者表达式做出图形

在axes上绘制线或者标志。使用plt.plot则默认为当前axes对象。

plt.plot(x,y,linewidth,str\_format)

返回值是一个存放着所画线或者标记的对象matplotlib.lines.Line2D object，是个列表形式，如果使用一次plot，画多个图形，返回多个实例。

plot(x, y, color='green', linestyle='dashed', marker='o',

markerfacecolor='blue', markersize=12)

（x,y）（x、y都为列表或数组，一一对应）点用蓝色（markerfacecolor），12号大小（markersize）的圆点（marker）标出。用线型（linestyle）为dashed的绿色（color）的线连起来。

#### 参数：

x：可省略，省略后默认为是 x=0 至n-1 其中n为y数组的个数。即确保每一个y都有一个x对应。

y:列表、数组、向量、单个数

linewidth:整数，表示线宽

str\_format：为画图的格式：

颜色线型等。

可以组合在一起:’r-’，也可以单独设置：color=’r’

和 \*\*MATLAB\*\* 中类似，我们还可以用字符来指定绘图的格式：

表示颜色的字符参数有：

字符 | 颜色

‘b’| 蓝色，blue

‘g’| 绿色，green

‘r’| 红色，red

‘c’| 青色，cyan

‘m’| 品红，magenta

‘y’| 黄色，yellow

‘k’| 黑色，black

‘w’| 白色，white

此外，还可以以很多方式来指定颜色，包括全名(“green”)、十六进制字符串(“#008000”)、RGB或RGBA元组（(0,1,0,1)）或灰度强度作为字符串(‘0.8’)。其中，字符串规范可以代替FMT组，但元组表单只能用作kwargs。

表示类型的字符参数有：

字符|类型 | 字符|类型

'-' 实线样式

'--' 短横线样式

'-.' 点划线样式

':' 虚线样式

'.' 点标记

',' 像素标记

'o' 圆标记

'v' 倒三角标记

'^' 正三角标记

'<' 左三角标记

'>' 右三角标记

'1' 下箭头标记

'2' 上箭头标记

'3' 左箭头标记

'4' 右箭头标记

's' 正方形标记

'p' 五边形标记

'\*' 星形标记

'h' 六边形标记 1

'H' 六边形标记 2

'+' 加号标记

'x' X 标记

'D' 菱形标记

'd' 窄菱形标记

'&#124;' 竖直线标记

'\_' 水平线标记

#### 作图

例如我们要画出红色圆点：

plt.plot([1,2,3,4], [1,4,9,16], 'ro')

plt.show()

同时做多个图：

t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green triangles

plt.plot(t, t, 'r--',

t, t\*\*2, 'bs',

t, t\*\*3, 'g^')

plt.show()

#### 修改设置线条属性

##### 使用 plt.plot() 的返回值来设置线条属性

plot 函数返回一个 Line2D 对象组成的列表，每个对象代表输入的一对组合，例如：

- line1, line2 为两个 Line2D 对象

line1, line2 = plt.plot(x1, y1, x2, y2)

- 返回 3 个 Line2D 对象组成的列表

lines = plt.plot(x1, y1, x2, y2, x3, y3)

我们可以使用这个返回值来对线条属性进行设置：

# 加逗号 line 中得到的是 line2D 对象，不加逗号得到的是只有一个 line2D 对象的列表

line, = plt.plot(x, y, 'r-')

# 将抗锯齿关闭

line.set\_antialiased(False)

plt.show()

##### plt.setp() 修改线条性质

lines = plt.plot(x, y)

# 使用键值对

plt.setp(lines, color='r', linewidth=2.0)

# 或者使用 MATLAB 风格的字符串对

plt.setp(lines, 'color', 'r', 'linewidth', 2.0)

plt.show()

可以设置的属性有很多，可以使用 plt.setp(lines) 查看 lines 可以设置的属性，各属性的含义可参考 matplotlib 的文档。

### 4.子图figure、subplot

figure() 函数会产生一个指定编号为 num 的图：

plt.figure(num)

这里，figure(1) 其实是可以省略的，因为默认情况下 plt 会自动产生一幅图像。

或者更详细的设置：

plt.figure(figsize=(8,8),dpi=80) # 画图之前首先设置figure对象，此函数相当于设置一块自定义大小的画布，使得后面的图形输出在这块规定了大小的画布上，其中参数figsize设置画布大小

figsize设置输出的大小

a=plt.figure(figsize=(10,10))

ax=a.subplot(211) :获取画图的实例，可以进行画图等一些列操作。

ax.plot()….

使用 subplot 可以在一副图中生成多个子图，其参数为：

plt.subplot(numrows, numcols, fignum)

numrows为有几行图

numcols为每行有有几个图

fignum为接下来将画第几张图。

在画该图前进行设置并指明。

当 numrows \* numcols < 10 时，中间的逗号可以省略，因此 plt.subplot(211) 就相当于 plt.subplot(2,1,1)。

使用subplot后还需要用plot进行绘制。

def f(t):

return np.exp(-t) \* np.cos(2\*np.pi\*t)

t1 = np.arange(0.0, 5.0, 0.1)

t2 = np.arange(0.0, 5.0, 0.02)

plt.figure(1)

plt.subplot(211)

plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')

plt.subplot(212)

plt.plot(t2, np.cos(2\*np.pi\*t2), 'r--')

plt.show()

### 直方图plt.hist()

import matplotlib.pyplot as plt

import numpy as np

import matplotlib

# 设置matplotlib正常显示中文和负号

matplotlib.rcParams['font.sans-serif']=['SimHei'] # 用黑体显示中文

matplotlib.rcParams['axes.unicode\_minus']=False # 正常显示负号

# 随机生成（10000,）服从正态分布的数据

data = np.random.randn(10000)

"""

绘制直方图

data:必选参数，绘图数据

bins:直方图的长条形数目，可选项，默认为10

density:是否将得到的直方图向量归一化，可选项，默认为0，代表不归一化，显示频数。normed=1，表示归一化，显示频率。

facecolor:长条形的颜色

edgecolor:长条形边框的颜色

alpha:透明度

"""

plt.hist(data, bins=40, density=0, facecolor="blue", edgecolor="black", alpha=0.7)

# 显示横轴标签

plt.xlabel("区间")

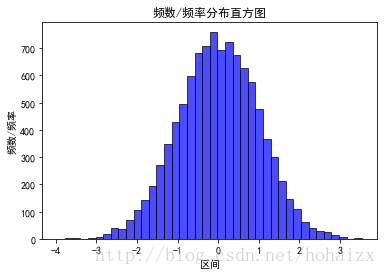
# 显示纵轴标签

plt.ylabel("频数/频率")

# 显示图标题

plt.title("频数/频率分布直方图")

plt.show()



### 条形图plt.bar(x,y)

#### 1.普通条形图bar

import matplotlib.pyplot as plt

import matplotlib

# 设置中文字体和负号正常显示

matplotlib.rcParams['font.sans-serif'] = ['SimHei']

matplotlib.rcParams['axes.unicode\_minus'] = False

label\_list = ['2014', '2015', '2016', '2017'] # 横坐标刻度显示值

num\_list1 = [20, 30, 15, 35] # 纵坐标值1

num\_list2 = [15, 30, 40, 20] # 纵坐标值2

x = range(len(num\_list1))

"""

绘制条形图

left:长条形中点横坐标

height:长条形高度

width:长条形宽度，默认值0.8

label:为后面设置legend准备

"""

rects1 = plt.bar(left=x, height=num\_list1, width=0.4, alpha=0.8, color='red', label="一部门")

rects2 = plt.bar(left=[i + 0.4 for i in x], height=num\_list2, width=0.4, color='green', label="二部门")

plt.ylim(0, 50) # y轴取值范围

plt.ylabel("数量")

"""

设置x轴刻度显示值

参数一：中点坐标

参数二：显示值

"""

plt.xticks([index + 0.2 for index in x], label\_list)

plt.xlabel("年份")

plt.title("某某公司")

plt.legend() # 设置题注

# 编辑文本

for rect in rects1:

height = rect.get\_height()

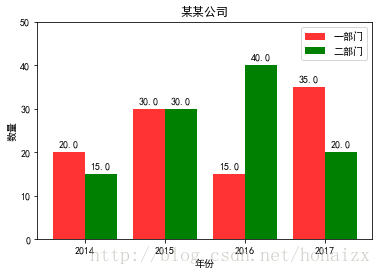
plt.text(rect.get\_x() + rect.get\_width() / 2, height+1, str(height), ha="center", va="bottom")

for rect in rects2:

height = rect.get\_height()

plt.text(rect.get\_x() + rect.get\_width() / 2, height+1, str(height), ha="center", va="bottom")

plt.show()



#### 2.水平条形图barh

import matplotlib.pyplot as plt

import matplotlib

matplotlib.rcParams['font.sans-serif'] = ['SimHei']

matplotlib.rcParams['axes.unicode\_minus'] = False

price = [39.5, 39.9, 45.4, 38.9, 33.34]

"""

绘制水平条形图方法barh

参数一：y轴

参数二：x轴

"""

plt.barh(range(5), price, height=0.7, color='steelblue', alpha=0.8) # 从下往上画

plt.yticks(range(5), ['亚马逊', '当当网', '中国图书网', '京东', '天猫'])

plt.xlim(30,47)

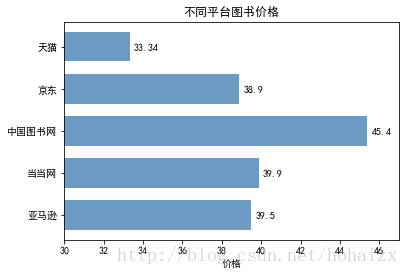
plt.xlabel("价格")

plt.title("不同平台图书价格")

for x, y in enumerate(price):

plt.text(y + 0.2, x - 0.1, '%s' % y)

plt.show()



#### 3. 堆叠条形图bar

需要放上面的条形图加bottom参数，指定为放下面条形图的高度。

使用bottom参数

import matplotlib.pyplot as plt

import matplotlib

matplotlib.rcParams['font.sans-serif'] = ['SimHei']

matplotlib.rcParams['axes.unicode\_minus'] = False

label\_list = ['2014', '2015', '2016', '2017']

num\_list1 = [20, 30, 15, 35]

num\_list2 = [15, 30, 40, 20]

x = range(len(num\_list1))

rects1 = plt.bar(left=x, height=num\_list1, width=0.45, alpha=0.8, color='red', label="一部门")

rects2 = plt.bar(left=x, height=num\_list2, width=0.45, color='green', label="二部门", bottom=num\_list1)

plt.ylim(0, 80)

plt.ylabel("数量")

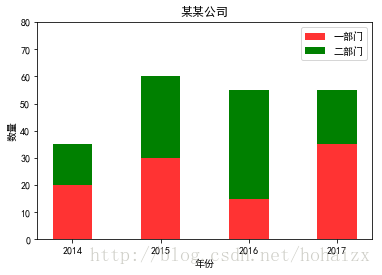
plt.xticks(x, label\_list)

plt.xlabel("年份")

plt.title("某某公司")

plt.legend()

plt.show()



### 点状图 plt.scatter

极坐表绘制跳转图：

import numpy as np

import matplotlib.pyplot as plt

ax = plt.axes([0.025,0.025,0.95,0.95], polar=True) #设置为极坐标

N = 20

theta = np.arange(0.0, 2\*np.pi, 2\*np.pi/N) #取好角度

radii = 10\*np.random.rand(N) #取好半径

width = np.pi/4\*np.random.rand(N)

bars = plt.bar(theta, radii, width=width, bottom=0.0)

for r,bar in zip(radii, bars):

bar.set\_facecolor( plt.cm.jet(r/10.))

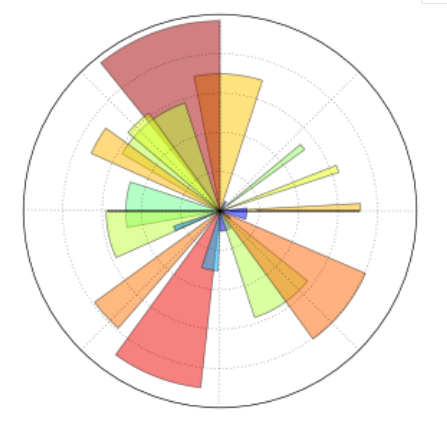
bar.set\_alpha(0.5)

ax.set\_xticklabels([])

ax.set\_yticklabels([])

# savefig('../figures/polar\_ex.png',dpi=48)

plt.show()



### 饼状图

import matplotlib.pyplot as plt

import matplotlib

matplotlib.rcParams['font.sans-serif'] = ['SimHei']

matplotlib.rcParams['axes.unicode\_minus'] = False

label\_list = ["第一部分", "第二部分", "第三部分"] # 各部分标签

size = [55, 35, 10] # 各部分大小

color = ["red", "green", "blue"] # 各部分颜色

explode = [0.05, 0, 0] # 各部分突出值，将某一块突出出来。本例是将第一块突出出来。

"""

绘制饼图

explode：设置各部分突出

label:设置各部分标签

labeldistance:设置标签文本距圆心位置，1.1表示1.1倍半径

autopct：设置圆里面文本

shadow：设置是否有阴影

startangle：起始角度，默认从0开始逆时针转

pctdistance：设置圆内文本距圆心距离

返回值

l\_text：圆内部文本，matplotlib.text.Text object

p\_text：圆外部文本

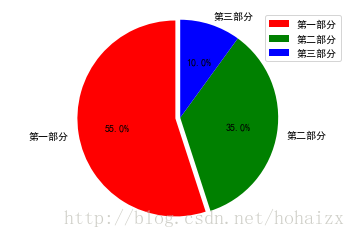
"""

patches, l\_text, p\_text = plt.pie(size, explode=explode, colors=color, labels=label\_list, labeldistance=1.1, autopct="%1.1f%%", shadow=False, startangle=90, pctdistance=0.6)

plt.axis("equal") # 设置横轴和纵轴大小相等，这样饼才是圆的

plt.legend()

plt.show()



import numpy as np

import matplotlib.pyplot as plt

n = 20

Z = np.ones(n)

Z[-1] \*= 2

plt.axes([0.025,0.025,0.95,0.95])

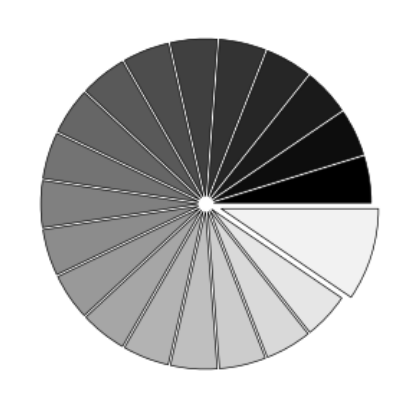
plt.pie(Z, explode=Z\*.05, colors = ['%f' % (i/float(n)) for i in range(n)])

plt.gca().set\_aspect('equal')

plt.xticks([]), plt.yticks([])

# savefig('../figures/pie\_ex.png',dpi=48)

plt.show()



### 极坐标

极坐标表示为(θ，ρ)。

#### 1.创建极坐标图

matplotlib的pyplot子库提供了绘制极坐标图的方法，在调用subplot()创建子图时通过设置projection='polar',便可创建一个极坐标子图，然后调用plot()在极坐标子图中绘图。  
下面就创建一个极坐标子图和一个直角坐标子图进行对比。极坐标为弧度

import matplotlib.pyplot as plt

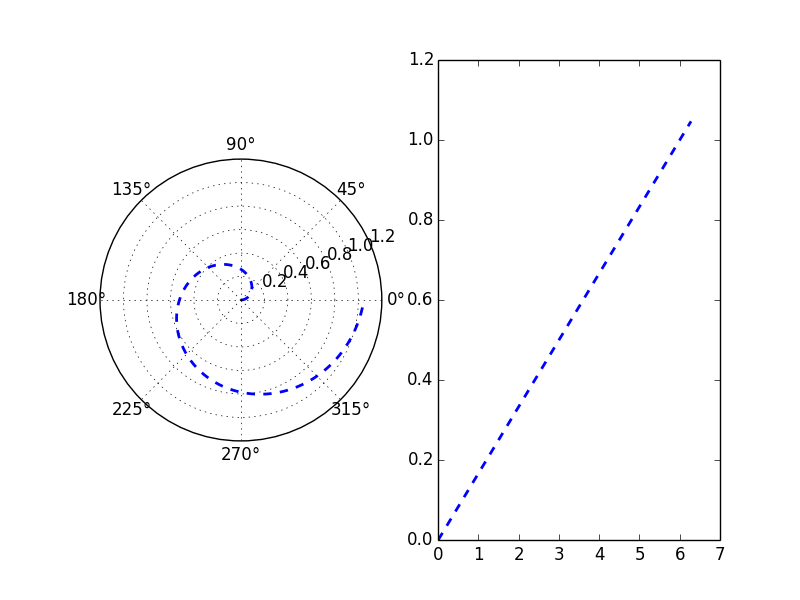
ax1 = plt.subplot(121, projection='polar')

ax2 = plt.subplot(122)

ax1.plot(theta,theta/6,'--',lw=2) #lw为线宽

ax2.plot(theta,theta/6,'--',lw=2)

plt.show()



#### 极坐标正方向

set\_theta\_direction方法用于设置极坐标的正方向

当set\_theta\_direction的参数值为1,'counterclockwise'或者是'anticlockwise'的时候，正方向为逆时针；

当set\_theta\_direction的参数值为-1或者是'clockwise'的时候，正方向为顺时针；

默认情况下正方向为逆时针

import matplotlib.pyplot as plt

import numpy as np

theta=np.arange(0,2\*np.pi,0.02)

ax1= plt.subplot(121, projection='polar') ax2= plt.subplot(122, projection='polar')

ax2.set\_theta\_direction(-1)

ax1.plot(theta,theta/6,'--',lw=2)

ax2.plot(theta,theta/6,'--',lw=2)

plt.show()

#### 极坐标0°位置

set\_theta\_zero\_location方法用于设置极坐标0°位置

0°可设置在八个位置，分别为N, NW, W, SW, S, SE, E, NE

当set\_theta\_zero\_location的参数值为'N','NW','W','SW','S','SE','E','NE'时，0°分别对应的位置为方位N, NW, W, SW, S, SE, E, NE；

默认情况下0°位于E方位

#### 极坐标角度网格线显示

set\_thetagrids方法用于设置极坐标角度网格线显示

参数为所要显示网格线的角度值列表

默认显示0°、45°、90°、135°、180°、225°、270°、315°的网格线

import matplotlib.pyplot as plt

import numpy as np

theta=np.arange(0,2\*np.pi,0.02)

ax1= plt.subplot(121, projection='polar')

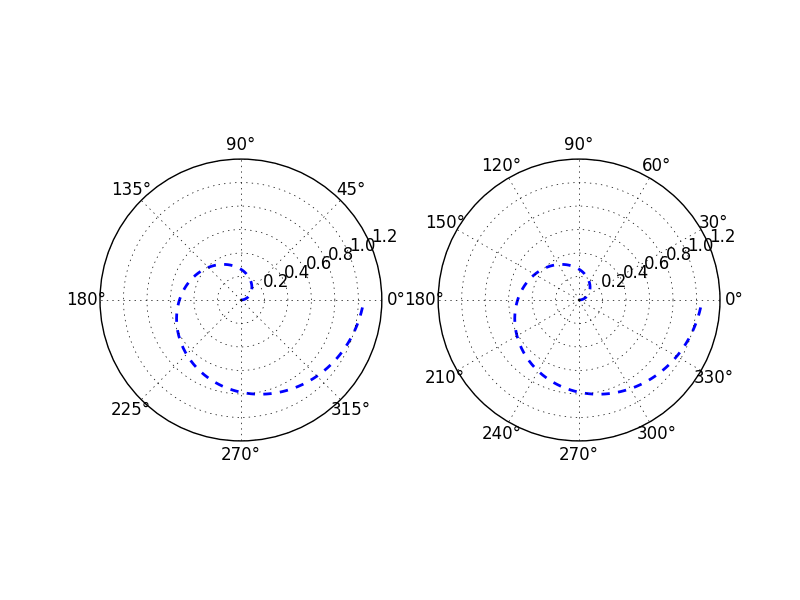
ax2= plt.subplot(122, projection='polar')

ax2.set\_thetagrids(np.arange(0.0, 360.0, 30.0))

ax1.plot(theta,theta/6,'--',lw=2)

ax2.plot(theta,theta/6,'--',lw=2)

plt.show()



#### 极坐标角度偏离

类似于切换0度的位置

set\_theta\_offset方法用于设置角度偏离

参数值为弧度值数值

import matplotlib.pyplot as plt

import numpy as np

theta=np.arange(0,2\*np.pi,0.02)

ax1= plt.subplot(121, projection='polar')

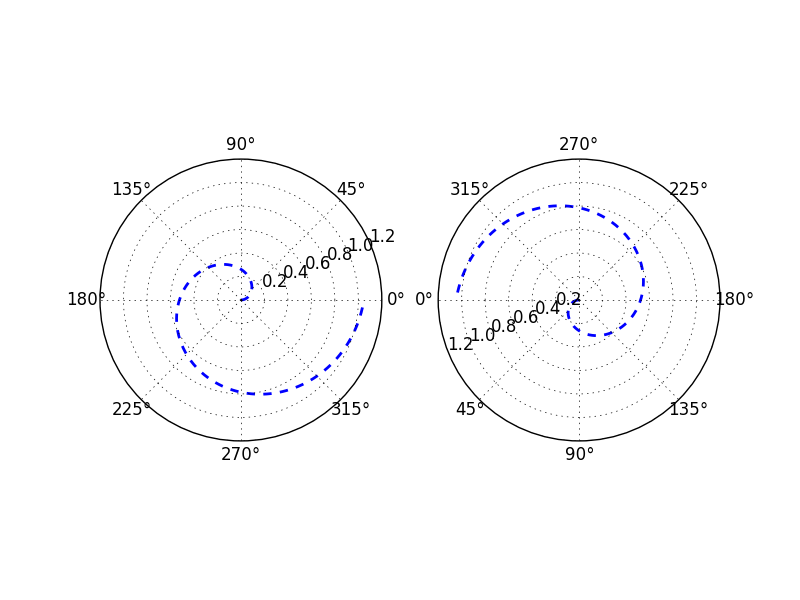
ax2= plt.subplot(122, projection='polar')

ax2.set\_theta\_offset(np.pi)

ax1.plot(theta,theta/6,'--',lw=2)

ax2.plot(theta,theta/6,'--',lw=2)

plt.show()



#### 极坐标极径网格线显示

set\_rgrids方法用于设置极径网格线显示

参数值为所要显示网格线的极径值列表，最小值不能小于等于0

import matplotlib.pyplot as plt

import numpy as np

theta=np.arange(0,2\*np.pi,0.02)

ax1= plt.subplot(121, projection='polar')

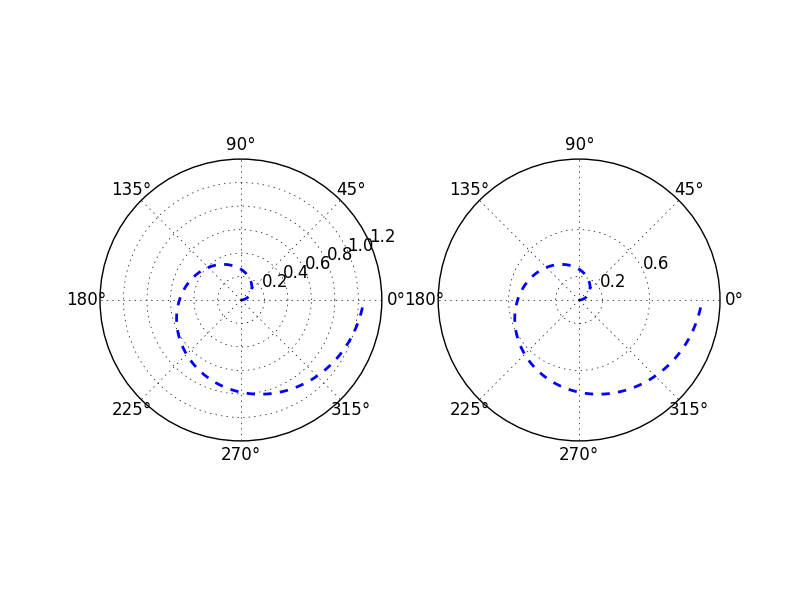
ax2= plt.subplot(122, projection='polar')

ax2.set\_rgrids(np.arange(0.2,1.0,0.4))

ax1.plot(theta,theta/6,'--',lw=2)

ax2.plot(theta,theta/6,'--',lw=2)

plt.show()



#### 极坐标极径标签位置

set\_rlabel\_position方法用于设置极径标签显示位置

参数为标签所要显示在的角度

import matplotlib.pyplot as plt

import numpy as np

theta=np.arange(0,2\*np.pi,0.02)

ax1= plt.subplot(121, projection='polar')

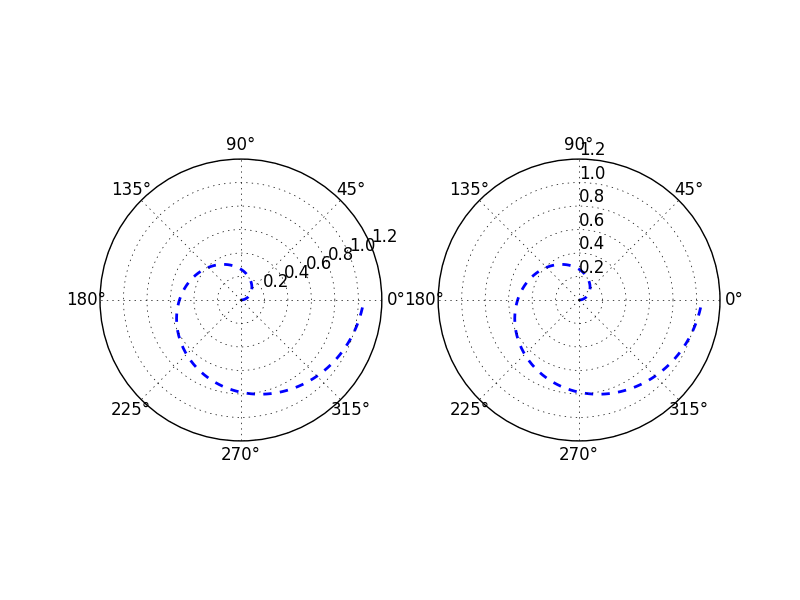
ax2= plt.subplot(122, projection='polar')

ax2.set\_rlabel\_position('90')

ax1.plot(theta,theta/6,'--',lw=2)

ax2.plot(theta,theta/6,'--',lw=2)

plt.show()



#### 极坐标极径范围

set\_rlim方法用于设置显示的极径范围

参数为极径最小值，最大值

import matplotlib.pyplot as plt

import numpy as np

theta=np.arange(0,2\*np.pi,0.02)

ax1= plt.subplot(121, projection='polar')

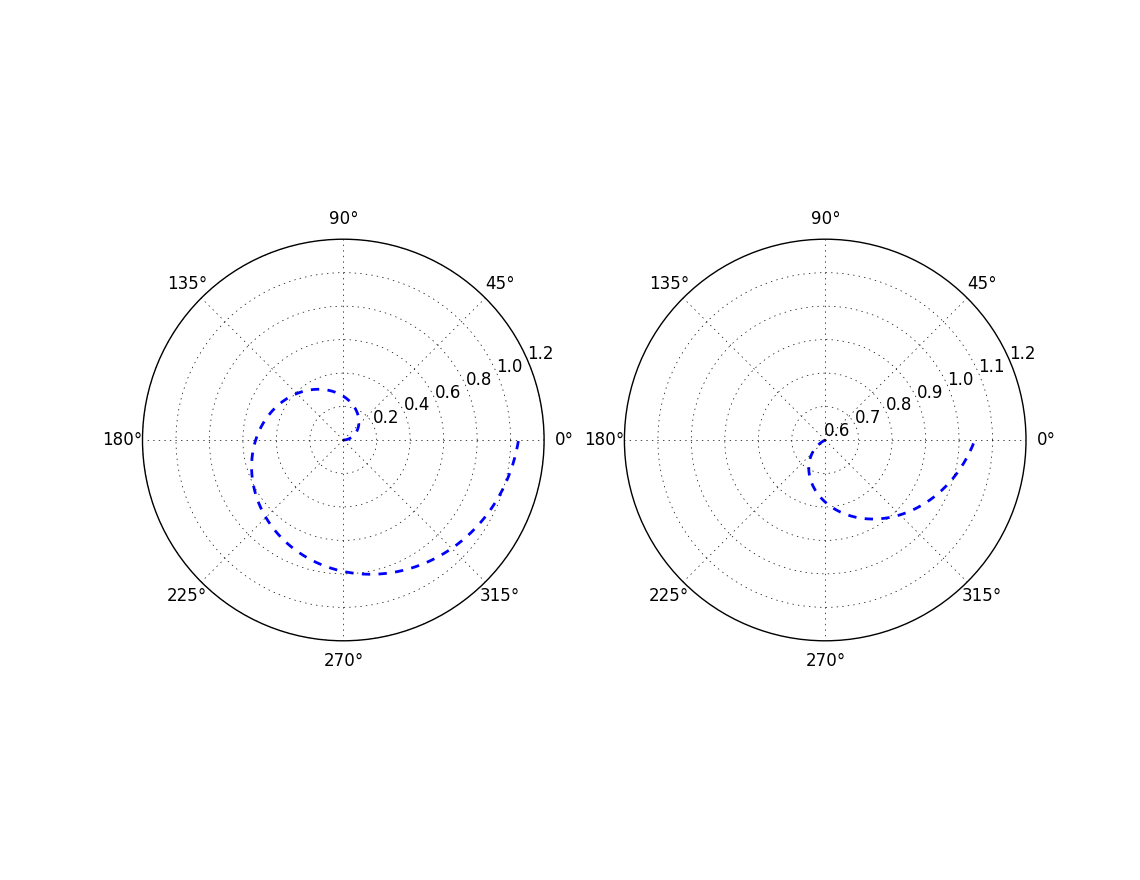
ax2= plt.subplot(122, projection='polar')

ax2.set\_rlim(0.6,1.2)

ax1.plot(theta,theta/6,'--',lw=2)

ax2.plot(theta,theta/6,'--',lw=2)

plt.show()



#### 极坐标极径最大值

set\_rmax方法用于设置显示的极径最大值

该方法要在绘制完图像后使用才有效

import matplotlib.pyplot as plt

import numpy as np

theta=np.arange(0,2\*np.pi,0.02)

ax1= plt.subplot(121, projection='polar')

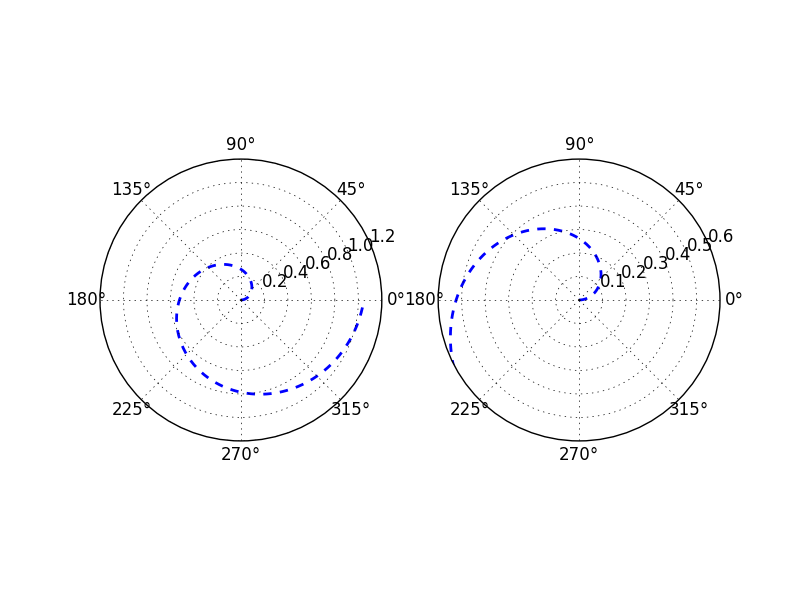
ax2= plt.subplot(122, projection='polar')

ax1.plot(theta,theta/6,'--',lw=2)

ax2.plot(theta,theta/6,'--',lw=2)

ax2.set\_rmax(0.6)

plt.show()



#### 极坐标极径最小值

set\_rmin方法用于设置显示的极径最小值

该方法要在绘制完图像后使用才有效

import matplotlib.pyplot as plt

import numpy as np

theta=np.arange(0,2\*np.pi,0.02)

ax1= plt.subplot(121, projection='polar')

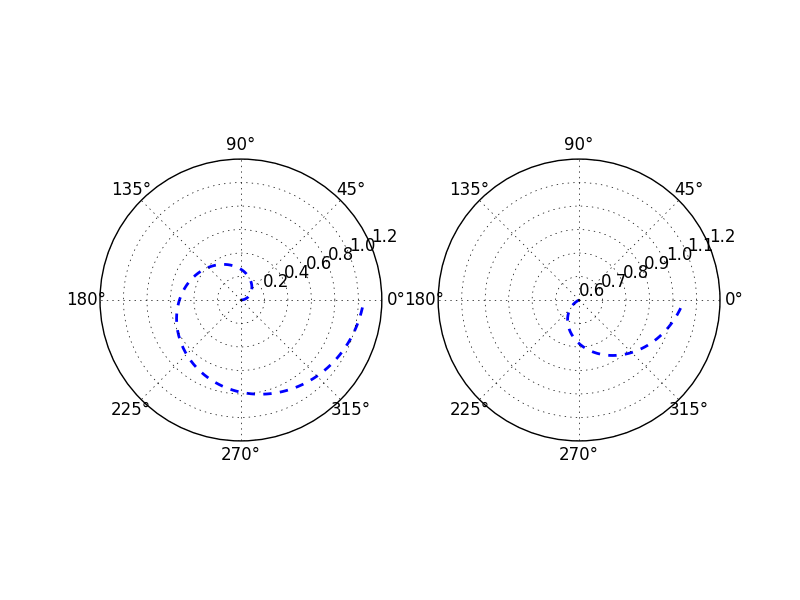
ax2= plt.subplot(122, projection='polar')

ax1.plot(theta,theta/6,'--',lw=2)

ax2.plot(theta,theta/6,'--',lw=2)

ax2.set\_rmin(0.6)

plt.show()



#### 极径对数坐标

set\_rscale方法用于设置极径对数坐标

参数值为'linear','log','symlog'

默认值为'linear'

该方法要在绘制完图像后使用才有效

import matplotlib.pyplot as plt

import numpy as np

theta=np.arange(0,2\*np.pi,0.02)

ax1= plt.subplot(121, projection='polar')

ax2= plt.subplot(122, projection='polar')

ax1.plot(theta,theta/6,'--',lw=2)

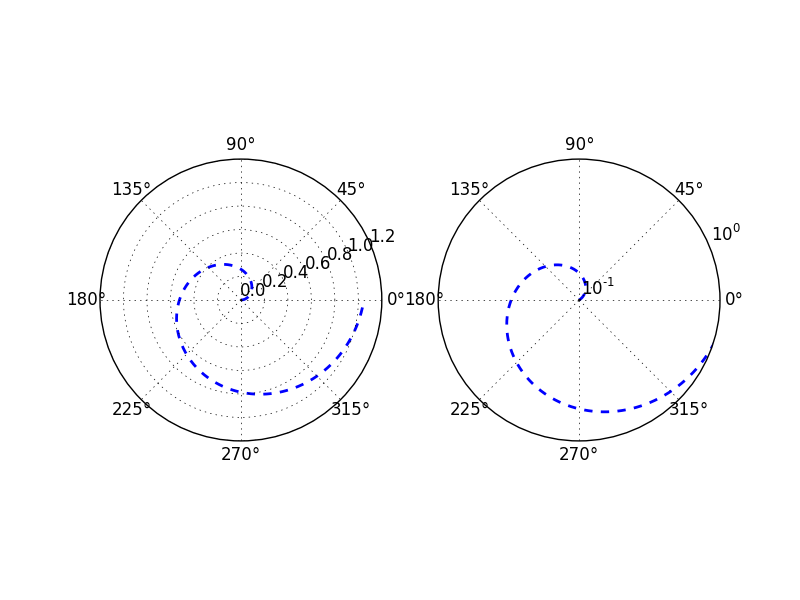
ax2.plot(theta,theta/6,'--',lw=2)

ax2.set\_rlim(math.pow(10,-1),math.pow(10,0))

ax1.set\_rscale('linear')

ax2.set\_rscale('symlog')

plt.show()



#### 极坐标极径网格线显示范围

set\_rticks方法用于设置极径网格线的显示范围

import matplotlib.pyplot as plt

import numpy as np

theta=np.arange(0,2\*np.pi,0.02)

ax1= plt.subplot(121, projection='polar')

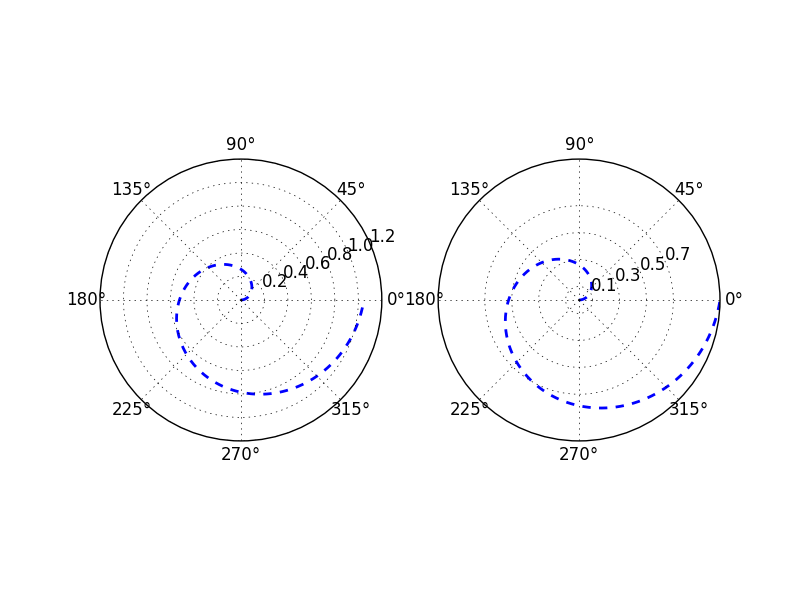
ax2= plt.subplot(122, projection='polar')

ax2.set\_rticks(np.arange(0.1, 0.9, 0.2))

ax1.plot(theta,theta/6,'--',lw=2)

ax2.plot(theta,theta/6,'--',lw=2)

plt.show()



### 使用style配置pyplot

#### style

style 是 pyplot 的一个子模块，方便进行风格转换， pyplot 有很多的预设风格，可以使用 plt.style.available 来查看：

plt.style.available

x = np.linspace(0, 2 \* np.pi)

y = np.sin(x)

plt.style.use('ggplot') #ggplot风格

plt.plot(x, y)

plt.show()

有时候，我们不希望改变全局的风格，只是想暂时改变一下分隔，则可以使用with和 context 将风格改变限制在某一个代码块内：

with plt.style.context(('dark\_background')):

plt.plot(x, y, 'r-o')

plt.show()

还可以混搭使用多种风格，不过最右边的一种风格会将最左边的覆盖：

plt.style.use(['dark\_background', 'ggplot'])

plt.plot(x, y, 'r-o')

plt.show()

#### 自定义style

事实上，我们还可以自定义风格文件。

自定义文件需要放在 matplotlib 的配置文件夹 mpl\_configdir 的子文件夹 mpl\_configdir/stylelib/ 下，以 .mplstyle 结尾。

mpl\_configdir 的位置可以这样查看：

import matplotlib

matplotlib.get\_configdir()

获取配置文件的地址。

里面的内容以 属性：值 的形式保存：

axes.titlesize : 24

axes.labelsize : 20

lines.linewidth : 3

lines.markersize : 10

xtick.labelsize : 16

ytick.labelsize : 16

假设我们将其保存为 mpl\_configdir/stylelib/presentation.mplstyle，那么使用这个风格的时候只需要调用：

plt.style.use('presentation')

### 处理文本

matplotlib 对文本的支持十分完善，包括数学公式，Unicode 文字，栅格和向量化输出，文字换行，文字旋转等一系列操作。

在 matplotlib.pyplot 中，基础的文本函数如下：

#### 基础函数

- text() 在 Axes 对象的任意位置添加文本

- xlabel() 添加 x 轴标题

- ylabel() 添加 y 轴标题

- title() 给 Axes 对象添加标题

- figtext() 在 Figure 对象的任意位置添加文本

- suptitle() 给 Figure 对象添加标题

- anotate() 给 Axes 对象添加注释（可选择是否添加箭头标记）

# -\*- coding: utf-8 -\*-

import matplotlib.pyplot as plt

%matplotlib inline

# plt.figure() 返回一个 Figure() 对象

fig = plt.figure(figsize=(12, 9))

# 设置这个 Figure 对象的标题

# 事实上，如果我们直接调用 plt.suptitle() 函数，它会自动找到当前的 Figure 对象

fig.suptitle('bold figure suptitle', fontsize=14, fontweight='bold')

# Axes 对象表示 Figure 对象中的子图

# 这里只有一幅图像，所以使用 add\_subplot(111)

ax = fig.add\_subplot(111) #获取子图对象

fig.subplots\_adjust(top=0.85)

# 可以直接使用 set\_xxx 的方法来设置标题

ax.set\_title('axes title')

# 也可以直接调用 title()，因为会自动定位到当前的 Axes 对象

# plt.title('axes title')

ax.set\_xlabel('xlabel')

ax.set\_ylabel('ylabel')

# 添加文本，斜体加文本框

ax.text(3, 8, 'boxed italics text in data coords', style='italic',

bbox={'facecolor':'red', 'alpha':0.5, 'pad':10})

# 数学公式，用 $$ 输入 Tex 公式

ax.text(2, 6, r'an equation: $E=mc^2$', fontsize=15)

# Unicode 支持

ax.text(3, 2, unicode('unicode: Institut f\374r Festk\366rperphysik', 'latin-1'))

# 颜色，对齐方式

ax.text(0.95, 0.01, 'colored text in axes coords',

verticalalignment='bottom', horizontalalignment='right',

transform=ax.transAxes,

color='green', fontsize=15)

# 注释文本和箭头

ax.plot([2], [1], 'o')

ax.annotate('annotate', xy=(2, 1), xytext=(3, 4),

arrowprops=dict(facecolor='black', shrink=0.05))

# 设置显示范围

ax.axis([0, 10, 0, 10])

plt.show()

#### 文本属性：

各属性不仅仅是text的属性，对于用图像、文字呈现出来的都具有这些属性。如做出的图、标题、轴、图例等等。

各属性都有对应的 set\_属性名 方法。对需要设置的对象进行直接调用。不仅仅包括文本对象。

alpha 浮点值 透明度 也可以使用set\_alpha方法来设置

animated [True / False] 是否动态

antialiased or aa [True / False]

clip\_box matplotlib.transform.Bbox 实例 用于绘制背景框

clip\_on [True / False] 是否绘制边框

clip\_path Path实例， Transform，以及Patch实例

color or c 任何 matplotlib 颜色

contains 命中测试函数

dash\_capstyle ['butt' / 'round' / 'projecting']

dash\_joinstyle ['miter' / 'round' / 'bevel']

dashes 以点为单位的连接/断开墨水序列

data (np.array xdata, np.array ydata)

figure matplotlib.figure.Figure 实例

label 任何字符串

linestyle or ls 线型[ '-' / '--' / '-.' / ':' / 'steps' / ...]

linewidth or lw 以点为单位的浮点值

lod [True / False]

marker [ '+' / ',' / '.' / '1' / '2' / '3' / '4' ]

markeredgecolor or mec 任何 matplotlib 颜色

markeredgewidth or mew 以点为单位的浮点值

markerfacecolor or mfc 任何 matplotlib 颜色

markersize or ms 浮点值

markevery [ None / 整数值 / (startind, stride) ]

picker 用于交互式线条选择

pickradius 线条的拾取选择半径

solid\_capstyle ['butt' / 'round' / 'projecting']

solid\_joinstyle ['miter' / 'round' / 'bevel']

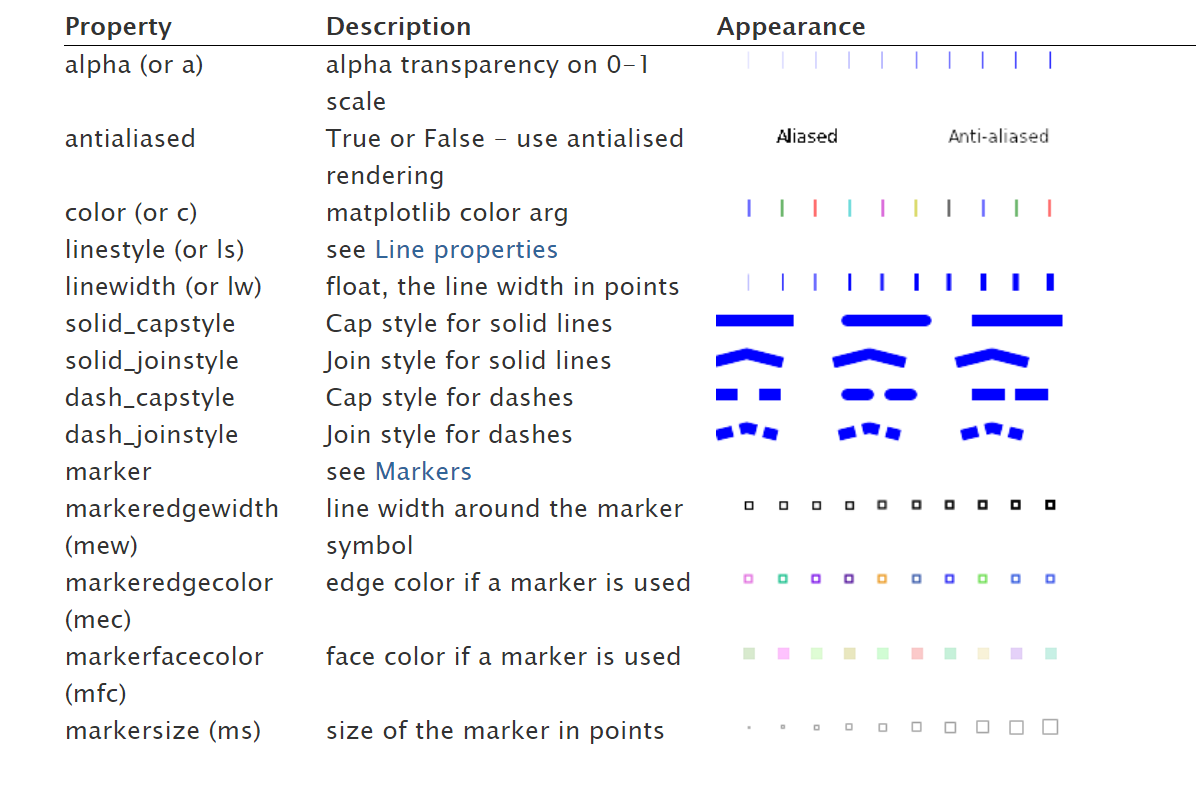
transform matplotlib.transforms.Transform 实例

visible [True / False]

xdata np.array

ydata np.array

zorder 任何数值



其中 `va`, `ha`, `multialignment` 可以用来控制布局。

- `horizontalalignment` or `ha` ：x 位置参数表示的位置 左对齐、右对齐、居中

- `verticalalignment` or `va`：y 位置参数表示的位置 上对齐、下对齐、居中

- `multialignment`：多行位置控制

import matplotlib.pyplot as plt

import matplotlib.patches as patches

# build a rectangle in axes coords

left, width = .25, .5

bottom, height = .25, .5

right = left + width

top = bottom + height

fig = plt.figure(figsize=(10,7))

ax = fig.add\_axes([0,0,1,1])

# axes coordinates are 0,0 is bottom left and 1,1 is upper right

p = patches.Rectangle(

(left, bottom), width, height,

fill=False, transform=ax.transAxes, clip\_on=False

)

ax.add\_patch(p)

ax.text(left, bottom, 'left top',

horizontalalignment='left',

verticalalignment='top',

transform=ax.transAxes,

size='xx-large')

ax.text(left, bottom, 'left bottom',

horizontalalignment='left',

verticalalignment='bottom',

transform=ax.transAxes,

size='xx-large')

ax.text(right, top, 'right bottom',

horizontalalignment='right',

verticalalignment='bottom',

transform=ax.transAxes,

size='xx-large')

ax.text(right, top, 'right top',

horizontalalignment='right',

verticalalignment='top',

transform=ax.transAxes,

size='xx-large')

ax.text(right, bottom, 'center top',

horizontalalignment='center',

verticalalignment='top',

transform=ax.transAxes,

size='xx-large')

ax.text(left, 0.5\*(bottom+top), 'right center',

horizontalalignment='right',

verticalalignment='center',

rotation='vertical',

transform=ax.transAxes,

size='xx-large')

ax.text(left, 0.5\*(bottom+top), 'left center',

horizontalalignment='left',

verticalalignment='center',

rotation='vertical',

transform=ax.transAxes,

size='xx-large')

ax.text(0.5\*(left+right), 0.5\*(bottom+top), 'middle',

horizontalalignment='center',

verticalalignment='center',

fontsize=20, color='red',

transform=ax.transAxes)

ax.text(right, 0.5\*(bottom+top), 'centered',

horizontalalignment='center',

verticalalignment='center',

rotation='vertical',

transform=ax.transAxes,

size='xx-large')

ax.text(left, top, 'rotated\nwith newlines',

horizontalalignment='center',

verticalalignment='center',

rotation=45,

transform=ax.transAxes,

size='xx-large')

ax.set\_axis\_off()

plt.show()

## spline类 边柱

### 基本情况

class matplotlib.spines.Spine(axes, spine\_type, path, \*\*kwargs)

可以通过axes对象来调用，axes.spines[‘left’].method：只能调用方形柱

ax = plt.gca()

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

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

ax.xaxis.set\_ticks\_position('bottom')

ax.spines['bottom'].set\_position(('data',0))

ax.yaxis.set\_ticks\_position('left')

ax.spines['left'].set\_position(('data',0))

即图像四周的四个边框坐标。left、right、bottom、top，可以使上和右坐标不显示，下和左坐标移到中间，以0点为原点。

matplotlib.spines

class matplotlib.spines.Spine(axes, spine\_type, path, \*\*kwargs)[source]

Bases: matplotlib.patches.Patch

an axis spine -- the line noting the data area boundaries

Spines are the lines connecting the axis tick marks and noting the boundaries of the data area. They can be placed at arbitrary positions.

See function: [set\_position](https://matplotlib.org/api/spines_api.html#matplotlib.spines.Spine.set_position)  for more information.

默认位置是 ('outward',0)

Spines 是 [Patch](https://matplotlib.org/api/_as_gen/matplotlib.patches.Patch.html#matplotlib.patches.Patch)的子类，并继承了很多方法。

使用类方法set\_patch\_line/set\_patch\_circle/set\_patch\_arc来设置直线、圆形、弧形。

如果直接实例化，sp=spine.Spine(….),则为直线。

spine\_arc弧度

sp=spine.Spine.arc\_spine(*axes*, *spine\_type*, *center*, *radius*, *theta1*, *theta2*, *\*\*kwargs*)

spine\_circle为圆形，sp=spine.Spine.circular\_spine(*axes*, *center*, *radius*, *\*\*kwargs*)

参数：

axes : the Axes instance containing the spine。需要绘制spine的axes的实例。

spine\_type : 一个字符串，指定对哪条边界进行设置。left、right、top、bottom

path : 用来绘制spline的path实例。

属性：

agg\_filter a filter function, which takes a (m, n, 3) float array and a dpi value, and returns a (m, n, 3) array

alpha float or None 透明度

animated bool 是否动态

antialiased unknown

capstyle {'butt', 'round', 'projecting'}

clip\_box Bbox

clip\_on bool

clip\_path [(Path, Transform) | Patch | None]

color color

contains callable

edgecolor color or None or 'auto'，边框颜色。

facecolor color or None 填充色

figure Figure

fill bool

gid str 背景网络

hatch {'/', '\', '|', '-', '+', 'x', 'o', 'O', '.', '\*'}

in\_layout bool

joinstyle {'miter', 'round', 'bevel'}

label object

linestyle {'-', '--', '-.', ':', '', (offset, on-off-seq), ...}

linewidth float or None for default

path\_effects AbstractPathEffect

picker None or bool or float or callable

rasterized bool or None

sketch\_params (scale: float, length: float, randomness: float)

snap bool or None

transform Transform

url str

visible bool

zorder float

### 方法

#### 1.arc\_spine(..)

类方法，返回一个弧状边框柱

classmethod arc\_spine(axes, spine\_type, center, radius, theta1, theta2, \*\*kwargs)[source]

(classmethod) Returns an arc Spine.

#### 2.circular\_spine(…)

类方法，返回一个圆形边框柱

classmethod circular\_spine(axes, center, radius, \*\*kwargs)[source]

(staticmethod) Returns a circular Spine.

#### 3. linear\_spine

classmethod linear\_spine(axes, spine\_type, \*\*kwargs)[source]

(staticmethod) Returns a linear Spine.

#### 4.set\_position(tumple)

set the position of the spine

Spine position is specified by a 2 tuple of (position type, amount). The position types are:

* 'outward' : place the spine out from the data area by the specified number of points. (Negative values specify placing the spine inward.)
* 'axes' : place the spine at the specified Axes coordinate (from 0.0-1.0).
* 'data' : place the spine at the specified data coordinate.

Additionally, shorthand notations define a special positions:

* 'center' -> ('axes',0.5)
* 'zero' -> ('data', 0.0)

#### 3.cla()

清除当前的spine

cla()

Clear the current spine

#### 4.draw(renderer)

draw(renderer)[source]

Draw the Patch to the given renderer.

#### 5.get\_bounds()

get\_bounds()[source]

Get the bounds of the spine.

#### 6.get\_patch\_transform()

Return the Transform instance which takes patch coordinates to data coordinates.

For example, one may define a patch of a circle which represents a radius of 5 by providing coordinates for a unit circle, and a transform which scales the coordinates (the patch coordinate) by 5.

#### 7.get\_path()

Return the path of this patch

#### 8.get\_position()

get the spine position

#### get\_smart\_bounds()

get whether the spine has smart bounds

#### get\_spine\_transform()

get the spine transform

#### get\_window\_extent(renderer=None)

Get the axes bounding box in display space. Subclasses should override for inclusion in the bounding box "tight" calculation. Default is to return an empty bounding box at 0, 0.

Be careful when using this function, the results will not update if the artist window extent of the artist changes. The extent can change due to any changes in the transform stack, such as changing the axes limits, the figure size, or the canvas used (as is done when saving a figure). This can lead to unexpected behavior where interactive figures will look fine on the screen, but will save incorrectly.

#### is\_frame\_like()

return True if directly on axes frame

This is useful for determining if a spine is the edge of an old style MPL plot. If so, this function will return True.

#### register\_axis(axis)

register an axis

An axis should be registered with its corresponding spine from the Axes instance. This allows the spine to clear any axis properties when needed.

#### set\_bounds(low, high)

Set the bounds of the spine.

#### set\_color(c)

Set the edgecolor.

‘None’表示不显示

Parameters:

c : color or sequence of rgba tuples

.. seealso::

#### set\_facecolor(), set\_edgecolor()

For setting the edge or face color individually.

设置内部颜色和外部颜色

#### set\_patch\_arc(center, radius, theta1, theta2)

set the spine to be arc-like

#### set\_patch\_circle(center, radius)

set the spine to be circular

#### set\_patch\_line()

set the spine to be linear

#### set\_smart\_bounds(value)

set the spine and associated axis to have smart bounds

## legend()

用于添加图例

## axes画纸

axes即画纸对象，

plt.axes()

plt.axes(rect, projection=**None**, polar=**False**, \*\*kwargs)

plt.axes(ax)

1.plt.axes() 向当前figure添加一个axes对象。

和add\_subplot()不同，添加的axes是可以任意位置的，而subplot只能按照规定的顺序进行排列。

2.plt.axes(ax) 将ax选为当前使用对象。

3.plt.axes(….)

参数：

`subplot` 主要是使用网格排列子图：

subplot(2,1,1)

xticks([]), yticks([])

text(0.5,0.5, 'subplot(2,1,1)',ha='center',va='center',size=24,alpha=.5)

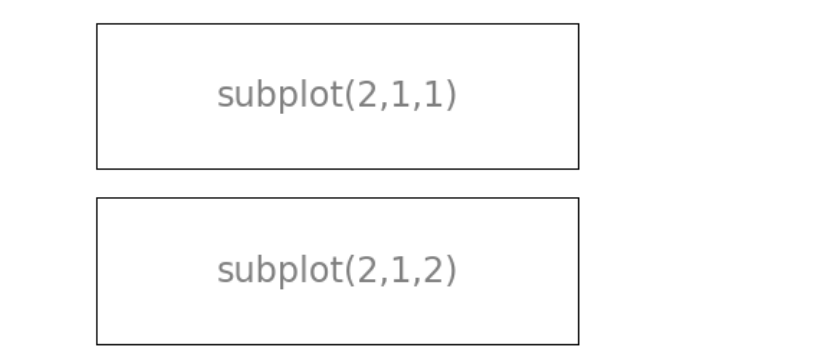
subplot(2,1,2)

xticks([]), yticks([])

text(0.5,0.5, 'subplot(2,1,2)',ha='center',va='center',size=24,alpha=.5)

show()

按照格式一点一点排开：



subplot也可以绘制第一行一个、第二行三个的样式：

ax1=plt.subplot(2,1,1):两行1列的第一个图，即第一行整行，此时将第一行画满。

ax2=plt.subplot(2,3,4)：两行2列的第4个图，即第二行第一个。

ax2=plt.subplot(2,3,5)：两行2列的第5个图，即第二行第2个。

而更高级的可以用 `gridspec` 来绘图：相对自由一些

import matplotlib.gridspec as gridspec

G = gridspec.GridSpec(3, 3)

axes\_1 = subplot(G[0, :])

xticks([]), yticks([])

text(0.5,0.5, 'Axes 1',ha='center',va='center',size=24,alpha=.5)

axes\_2 = subplot(G[1,:-1])

xticks([]), yticks([])

text(0.5,0.5, 'Axes 2',ha='center',va='center',size=24,alpha=.5)

axes\_3 = subplot(G[1:, -1])

xticks([]), yticks([])

text(0.5,0.5, 'Axes 3',ha='center',va='center',size=24,alpha=.5)

axes\_4 = subplot(G[-1,0])

xticks([]), yticks([])

text(0.5,0.5, 'Axes 4',ha='center',va='center',size=24,alpha=.5)

axes\_5 = subplot(G[-1,-2])

xticks([]), yticks([])

text(0.5,0.5, 'Axes 5',ha='center',va='center',size=24,alpha=.5)

show()



最自由的是axes：

`subplot` 返回的是 `Axes` 对象，但是 `Axes` 对象相对于 `subplot` 返回的对象来说要更自由一点。`Axes` 对象可以放置在图像中的任意位置：

axes([0.1,0.1,.8,.8])

xticks([]), yticks([])

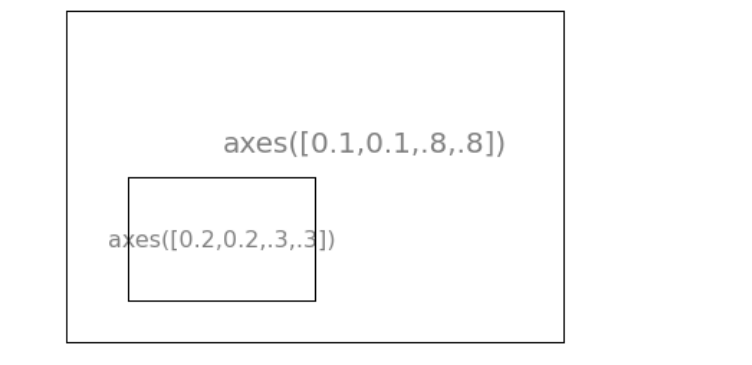
text(0.6,0.6, 'axes([0.1,0.1,.8,.8])',ha='center',va='center',size=20,alpha=.5)

axes([0.2,0.2,.3,.3])

xticks([]), yticks([])

text(0.5,0.5, 'axes([0.2,0.2,.3,.3])',ha='center',va='center',size=16,alpha=.5)

show()



axes([0.1,0.1,.5,.5])

xticks([]), yticks([])

text(0.1,0.1, 'axes([0.1,0.1,.8,.8])',ha='left',va='center',size=16,alpha=.5)

axes([0.2,0.2,.5,.5])

xticks([]), yticks([])

text(0.1,0.1, 'axes([0.2,0.2,.5,.5])',ha='left',va='center',size=16,alpha=.5)

axes([0.3,0.3,.5,.5])

xticks([]), yticks([])

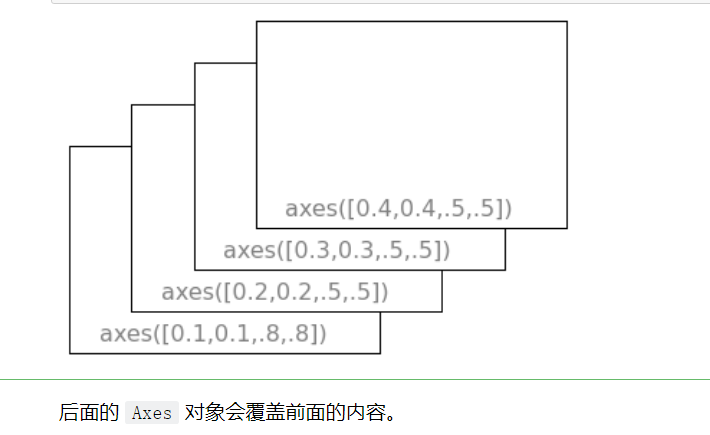
text(0.1,0.1, 'axes([0.3,0.3,.5,.5])',ha='left',va='center',size=16,alpha=.5)

axes([0.4,0.4,.5,.5])

xticks([]), yticks([])

text(0.1,0.1, 'axes([0.4,0.4,.5,.5])',ha='left',va='center',size=16,alpha=.5)

show()



其他获取axes对象的方法：

1.plt.subplots()

*#Creates just a figure and only one subplot*

fig, ax = plt.subplots()

ax.plot(x, y)

*#Creates two subplots and unpacks the output array immediately*

f, (ax1, ax2) = plt.subplots(1, 2, sharey=**True**)

*#Creates four polar axes, and accesses them through the returned array*

fig, axes = plt.subplots(2, 2, subplot\_kw=dict(polar=**True**))

axes[0, 0].plot(x, y)

axes[1, 1].scatter(x, y)

## bbox框

label.set\_bbox(字典)

为标签添加背景框

参数：

如facecolor='white', edgecolor='None', alpha=0.65等

## grid 背景网格

import numpy as np

import matplotlib.pyplot as plt

ax = plt.axes([0.025,0.025,0.95,0.95])

ax.set\_xlim(0,4)

ax.set\_ylim(0,3)

#设置主可读、副刻度

ax.xaxis.set\_major\_locator(plt.MultipleLocator(1.0))#主刻度间隔为1.0

ax.xaxis.set\_minor\_locator(plt.MultipleLocator(0.1))#副刻度间隔为0.1

ax.yaxis.set\_major\_locator(plt.MultipleLocator(1.0))

ax.yaxis.set\_minor\_locator(plt.MultipleLocator(0.1))

#设置网格，区分主刻度和副刻度

ax.grid(which='major', axis='x', linewidth=0.75, linestyle='-', color='0.75')

ax.grid(which='minor', axis='x', linewidth=0.25, linestyle='-', color='0.75')

ax.grid(which='major', axis='y', linewidth=0.75, linestyle='-', color='0.75')

ax.grid(which='minor', axis='y', linewidth=0.25, linestyle='-', color='0.75')

ax.set\_xticklabels([])

ax.set\_yticklabels([])

# savefig('../figures/grid\_ex.png',dpi=48)

plt.show()

## 颜色表示

### 1.RGBA表示法:

color=（0.5,0.2,0.3,0）

为一个4元元组，值介于0-1之间。

第四个元素是透明度，1代表不透明，0代表透明。可以省略。

### 2.灰度表示

color=’0.5’

为0-1之间的字符串，0表示全黑，1表示全白

### 3.内建颜色

蓝色 - 'b'

绿色 - 'g'

红色 - 'r'

青色 - 'c'

品红 - 'm'

黄色 - 'y'

黑色 - 'k'

白色 - 'w'

### 4.html表示

color='#00ff00'

六位16进制数表示，000000表示白色，ffffff表示黑色。