Python数据可视化之matplotlib实践

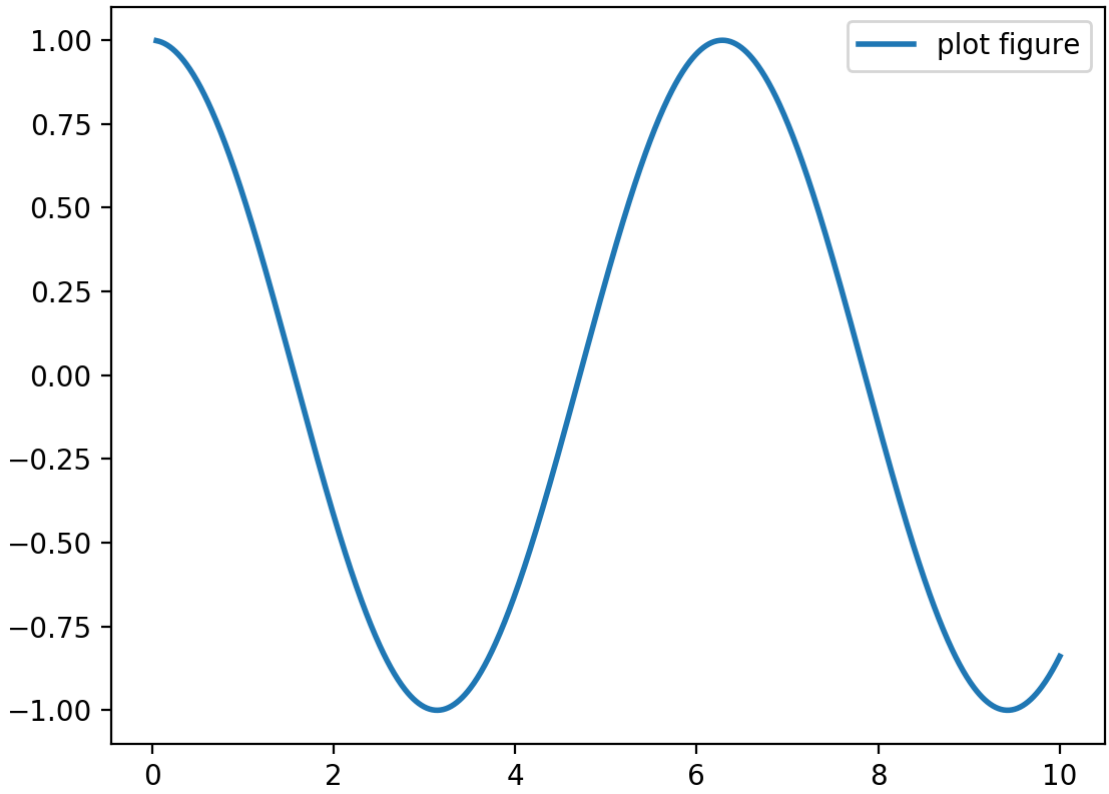
一、使用函数绘制图表

1.3.1 plot()——展现变量的趋势变化

`plt.plot(x,y,ls='-',lw=2,label='plot figure')`

- x：x轴上的数值
- y：y轴上的数值
- ls：折线图的线条风格
- lw：折线图的线条宽度
- label：标记图形内容的标签文本

```
x=np.linspace(0.05,10,1000)
y=np.cos(x)

plt.plot(x,y,ls='-',lw=2,label='plot figure')
plt.legend()
plt.show()
```

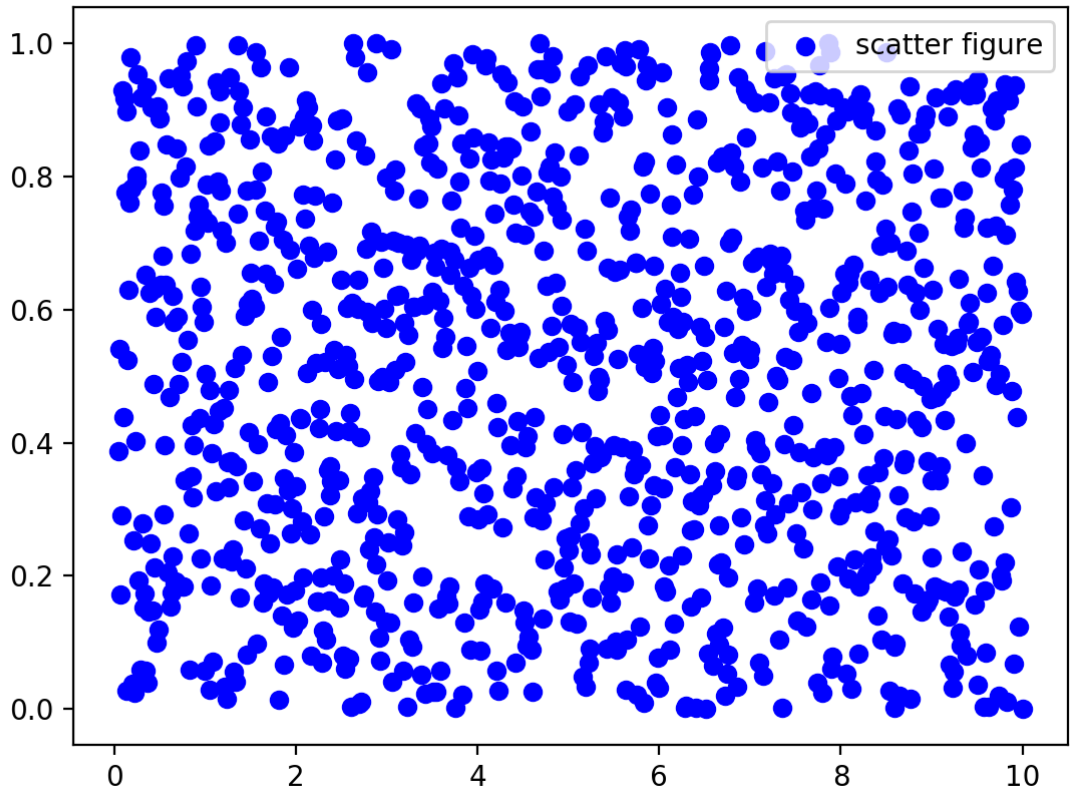


1.3.2 scatter()——寻找变量之间的关系

`plt.scatter(x,y,c='b',label='scatter figure')`

- x：x轴上的数值
- y：y轴上的数值
- c：散点图中的标记的颜色
- label：标记图形内容的标签文本

```
x=np.linspace(0.05,10,1000)
y=np.random.rand(1000)

plt.scatter(x,y,c='b',label='scatter figure')
plt.legend()
plt.show()
```



1.3.3 xlim()——设置x轴的数值显示范围

`plt.xlim(xmin, xmax)`

- xmin：x轴上的最小值
- xmax：x轴上的最大值，可以反过来变成逆序表示
- 平移性：函数同样对应ylim（）

1.3.4 xlabel()——设置x轴的标签文本

`plt.xlabel(string)`

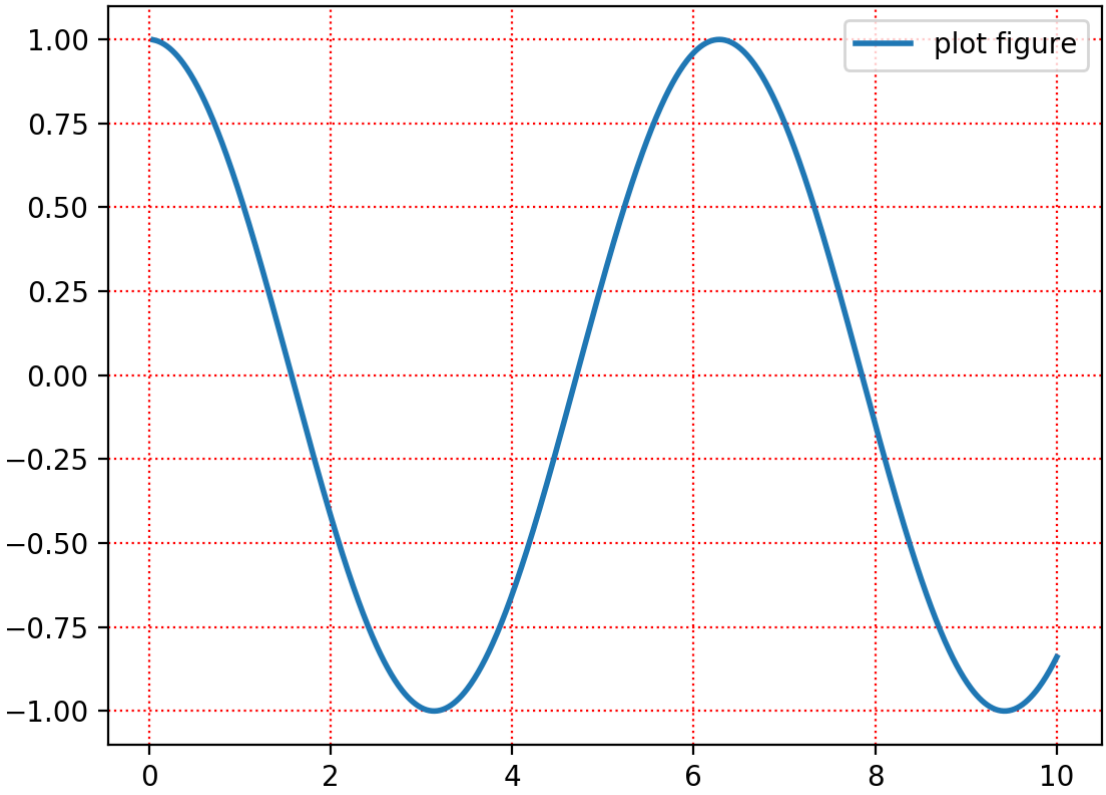
- string：标签文本内容
- 平移性：ylabel（）

1.3.5 grid()——绘制刻度线的网格线

`plt.grid(linestyle=":",color="r")`

- linestyle：网格线的线条风格
- color：网格线的线条颜色

```
x=np.linspace(0.05,10,1000)
y=np.cos(x)

plt.plot(x,y,ls='-',lw=2,label='plot figure')
plt.grid(linestyle=":",color="r")
plt.legend()
plt.show()
```

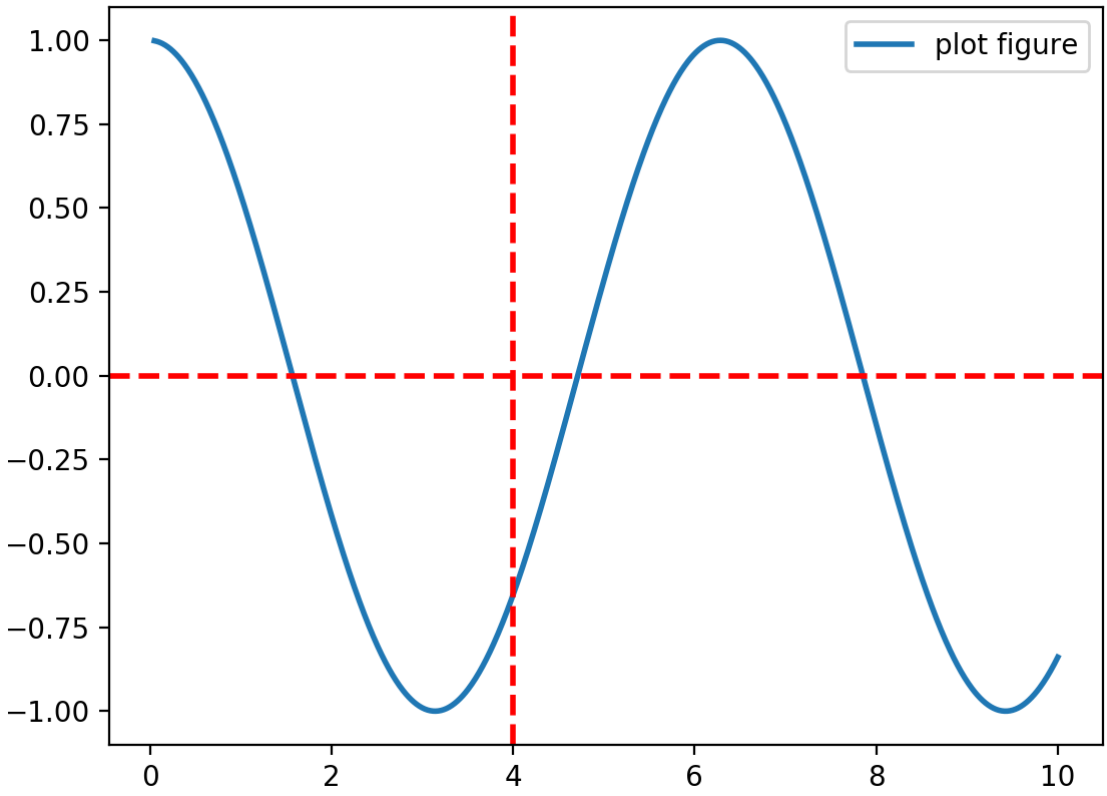


1.3.6 axhline()——绘制平行于x轴的水平参考线

`plt.axhline(y=0.0,c="r",ls="--",lw=2)`

- y：水平参考线的出发点
- c：参考线的线条颜色
- ls：参考线的线条风格
- lw：参考线的线条宽度
- 平移性：对应函数axvline（）

```
x=np.linspace(0.05,10,1000)
y=np.cos(x)

plt.plot(x,y,ls='-',lw=2,label='plot figure')
plt.axhline(y=0.0,c="r",ls="--",lw=2)
plt.axvline(x=4.0,c="r",ls="--",lw=2)
plt.legend()
plt.show()
```

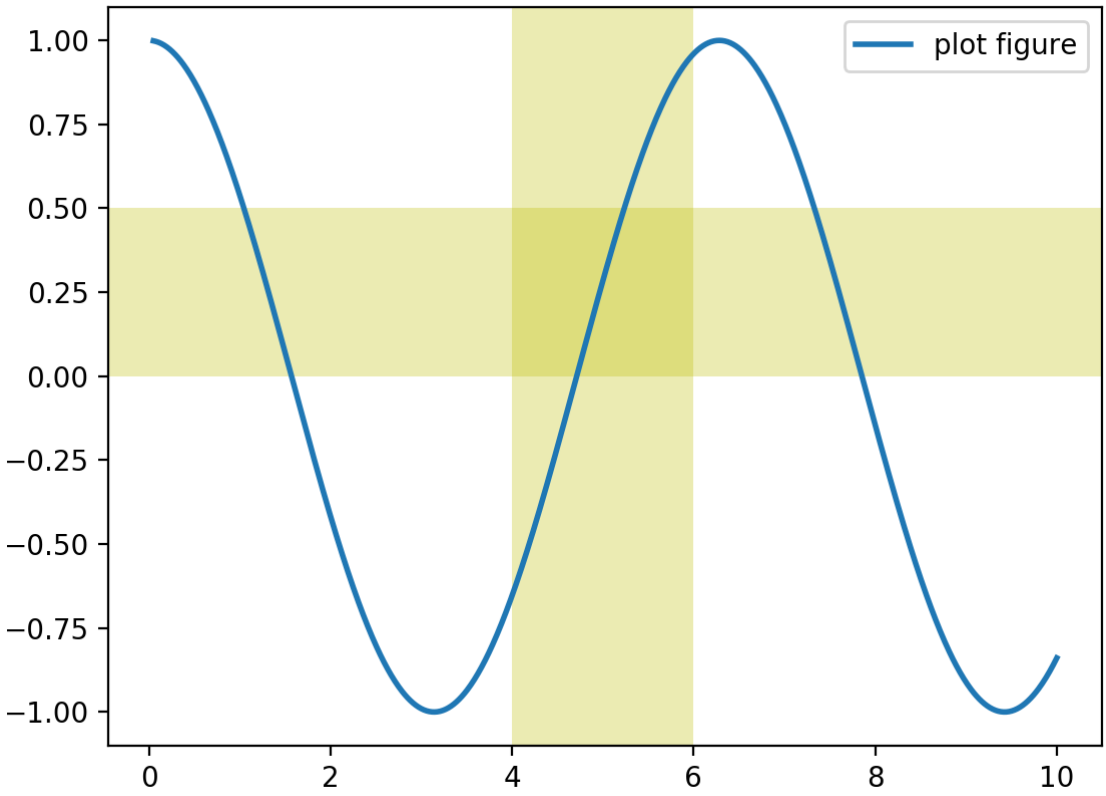


1.3.7 axvspan()——绘制垂直于x轴的参考区域

`plt.axvspan(xmin=4.0,xmax=6.0,facecolor="y",alpha=0.3)`

- xmin：参考区域的起始位置
- xmax：参考区域的终止位置
- facecolor：参考区域的填充颜色
- alpha：参考区域的填充颜色的透明度
- 平移性：axhspan（）

```
x=np.linspace(0.05,10,1000)
y=np.cos(x)

plt.plot(x,y,ls='-',lw=2,label='plot figure')
plt.axvspan(xmin=4.0,xmax=6.0,facecolor="y",alpha=0.3)
plt.axhspan(ymin=0.0,ymax=0.5,facecolor="y",alpha=0.3)
plt.legend()
```

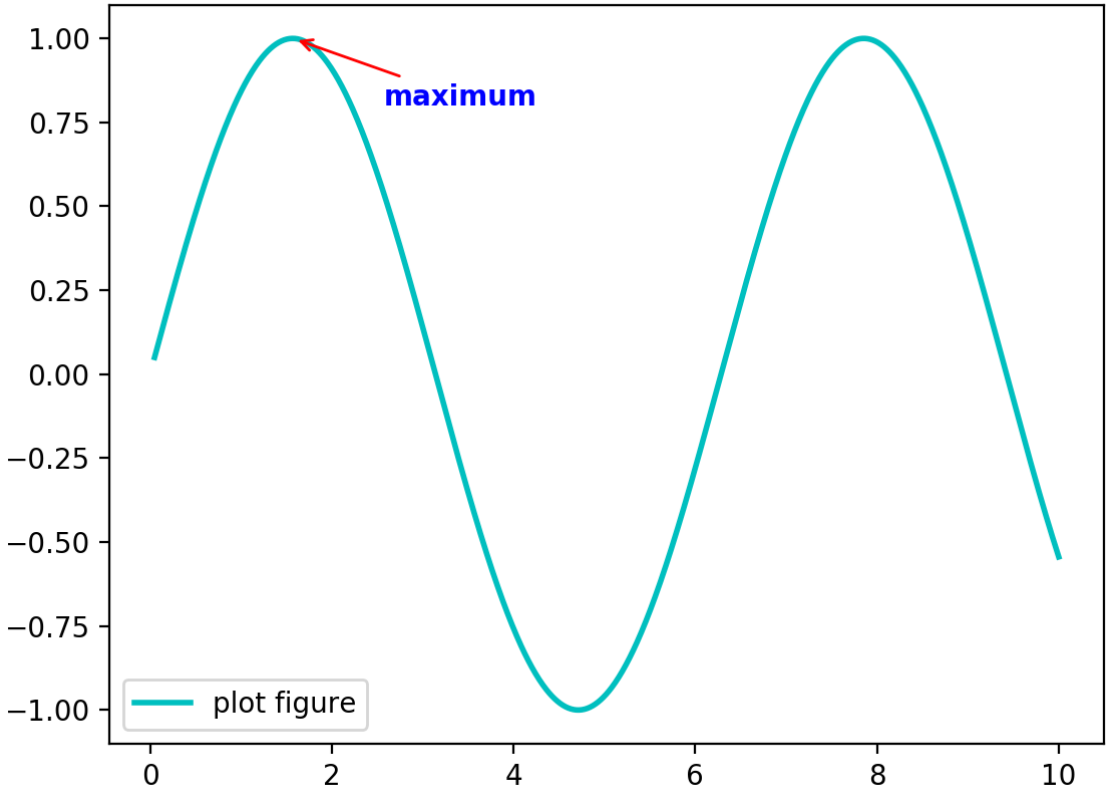


1.3.8 annotate()——添加图形内容细节的指向型注释文本

plt.annotate(string,
xy=(np.pi/2,1.0),
xytext=((np.pi/2)+1.0,.8),
weight="bold",
color="b",
arrowprops=dict(arrowstyle="->",connectionstyle='arc3',color="r"))

- string：图形内容的注释文本
- xy：被注释图形内容的位置坐标
- xytext：注释文本的位置坐标
- weight：注释文本的字体粗细风格
- color：注释文本的字体颜色
- arrowprops：指示被注释内容的箭头的属性字典

```
plt.plot(x,y,ls='-',lw=2,c='c',label='plot figure')
plt.annotate("maximum",
             xy=(np.pi/2,1.0),
             xytext=((np.pi/2)+1.0,.8),
             weight="bold",
             color="b",
             arrowprops=dict(arrowstyle="->",connectionstyle='arc3',color="r"))
plt.legend()
plt.show()
```

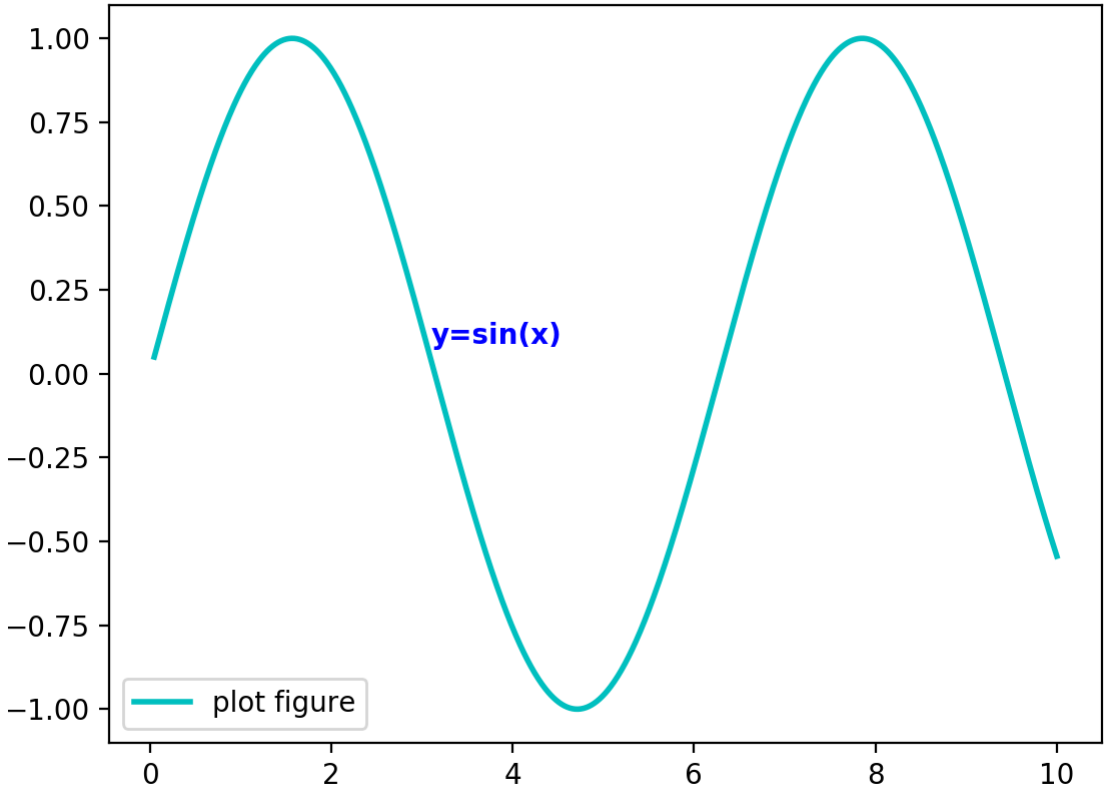


1.3.9 text()——添加图形内容细节的无指向型注释文本

`plt.text(3.10,0.09,"y=sin(x)",weight="bold",color="b")`

- x：注释文本内容所在位置的横坐标
- y：注释文本内容所在位置的纵坐标
- string：注释文本内容
- weight：注释文本内容的粗细风格
- color：注释文本内容的字体颜色

```
plt.plot(x,y,ls='-',lw=2,c='c',label='plot figure')
plt.text(3.10,0.09,"y=sin(x)",weight="bold",color="b")
plt.legend()
plt.show()
```

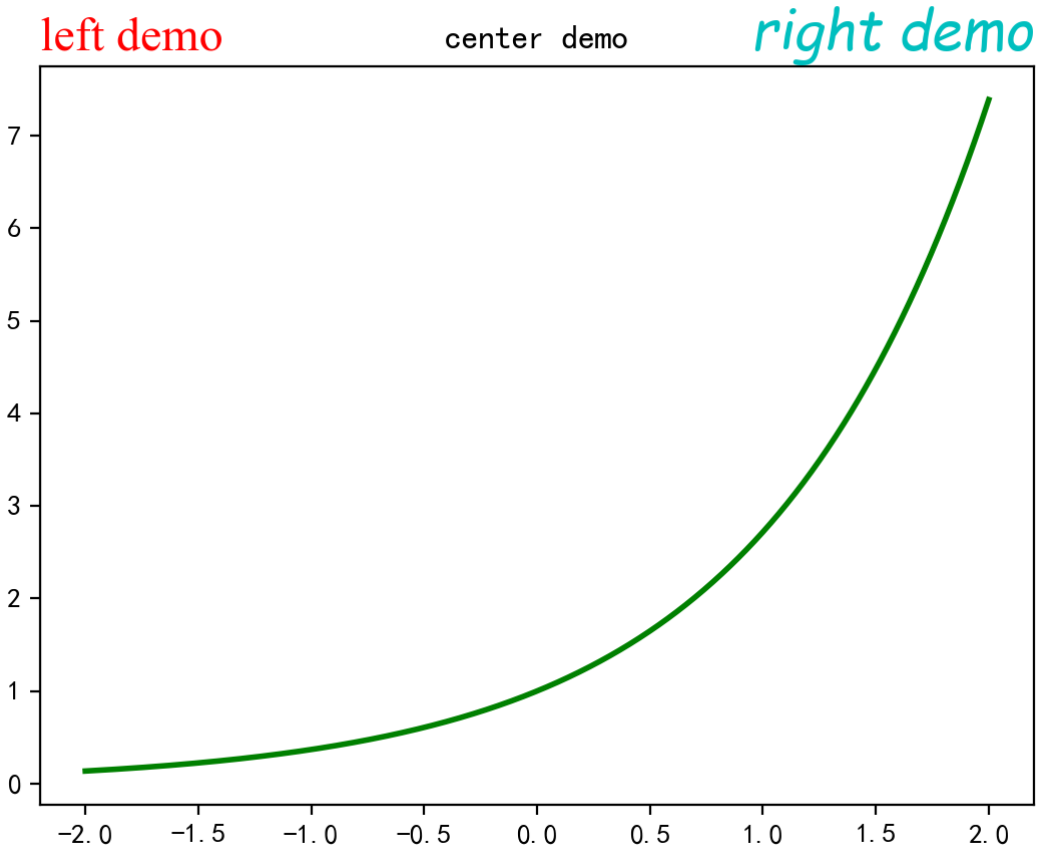


1.3.10 title()——添加图形内容的标题

```
ply.title(string)
```

- string：图形内容的标题文本
- fontdict：文本格式参数字典
  - family：字体类别
  - size：字体大小
  - color：字体颜色
  - style：字体风格
- fontsize：设置字体大小，默认12，参数 ['xx-small', 'x-small', 'small', 'medium', 'large','x-large', 'xx-large']
- fontweight：设置字体粗细，可选参数 ['light', 'normal', 'medium', 'semibold', 'bold', 'heavy', 'black']
- fontstyle：设置字体类型，可选参数['normal' | 'italic' | 'oblique']，italic斜体，oblique倾斜
- verticalalignment：设置水平对齐方式 ， 可选参数 ： 'center' , 'top' , 'bottom' , 'baseline'
- horizontalalignment：设置垂直对齐方式，可选参数：left,right,center
- rotation：(旋转角度)可选参数为:vertical,horizontal 也可以为数字
- alpha：透明度，参数值0至1之间
- backgroundcolor：标题背景颜色
- bbox：给标题增加外框 ， 常用参数如下：
  - boxstyle：方框外形
  - facecolor：(简写fc)背景颜色
  - edgecolor：(简写ec)边框线条颜色
  - edgewidth：边框线条大小

```
x=np.linspace(-2,2,1000)
y=np.exp(x)

plt.plot(x,y,ls='-',lw=2,color='g')

plt.title("center demo")
plt.title("left demo",loc="left",fontdict={"size":"xx-large",
                                           "color":"r",
                                           "family":"Times New Roman"})
plt.title("right demo",loc="right",family="Comic Sans MS",
          size=20,style="oblique",color='c')
plt.show()
```



1.3.11 legend()——标示不同图形的文本标签图例

`plt.legend(loc="lower left")`

- loc：图例在图中的位置

位置参数值	位置数值	位置参数值	位置数值	位置参数值	位置数值
upper right	1	upper left	2	lower left	3
lower right	4	center left	6	center right	7
lower center	8	upper center	9	center	10

- bbox_to_anchor：指定图例在轴的位置
- title：图例标签内容的标题参数
- shadow：线框阴影
- fancybox：线框圆角处理参数
- ncol：设置图例分为n列展示

- prop：字体参数
- markerscale：图例标记与原始标记的相对大小
- markerfirst：如果为True，则图例标记位于图例标签的左侧
- numpoints：为线条图图例条目创建的标记点数
- scatterpoints：为散点图图例条目创建的标记点数
- scatteryoffsets：为散点图图例条目创建的标记的垂直偏移量
- frameon：控制是否应在图例周围绘制框架
- fancybox：控制是否应在构成图例背景的FancyBboxPatch周围启用圆边
- framealpha：控制图例框架的 Alpha 透明度
- borderpad：图例边框的内边距
- labelspacing：图例条目之间的垂直间距
- handlelength：图例句柄的长度
- handleheight：图例句柄的高度
- handletextpad：图例句柄和文本之间的间距
- borderaxespad：轴与图例边框之间的距离
- columnspacing：列间距

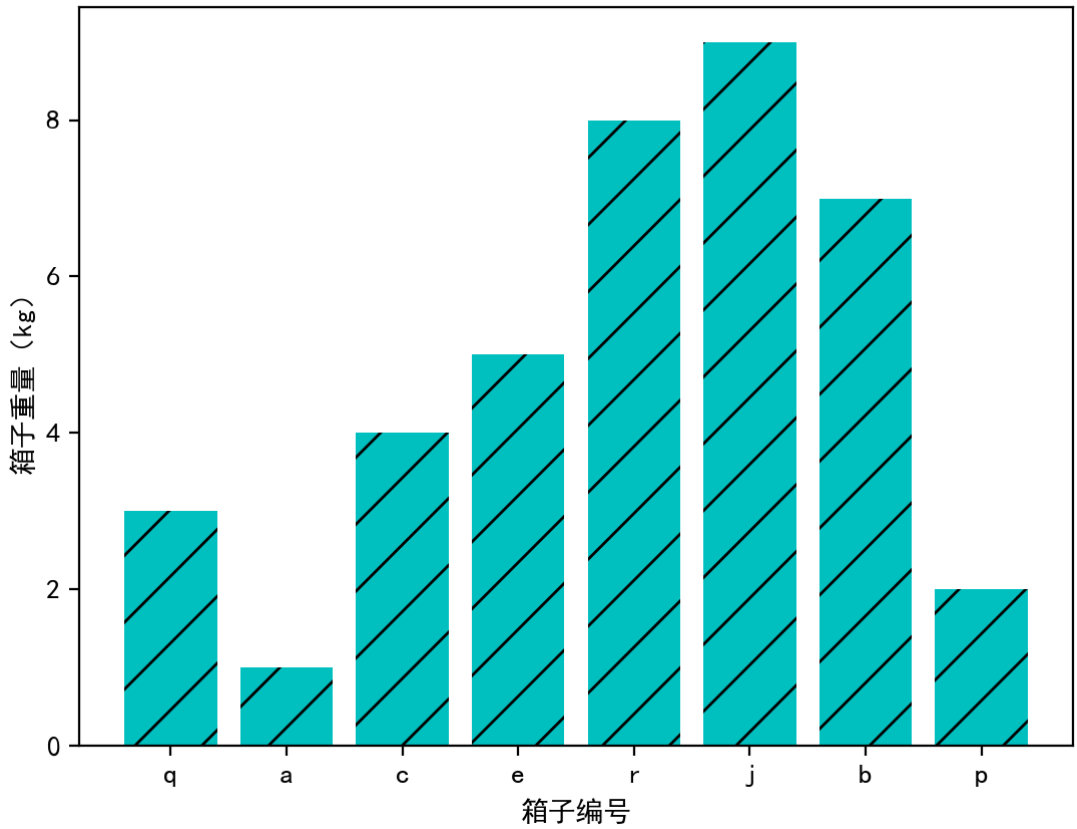
二、绘制简单图形

2.1 bar()——绘制柱状图

`plt.bar(x,height，width,bottom,align=" ",color=" ",edgecolor=" ",linewidth=int,tick_label=[],log=bool,orientation=" ")`

x	**x坐标**	**int,float**
height	条形的高度	int,float
width	线条的宽度	0~1，默认是0.8
bottom	条形的起始位置	也就是y轴的起始坐标
align	条形的中心位置	“center”,"lege"边缘
color	条形的颜色	“r”,“b”,“g”,“#123465",默认的颜色是“b”
edgecolor	边框的颜色	同上
linewidth	边框的宽度	像素，默认无，int
tick_label	下标的标签	可以是元组类型的字符组合
log	y轴使用科学计算法表示	bool
orientation	是竖直条还是水平条	竖直："vertical"，水平条："horizontal"
hatch	柱体的填充样式	”/“、”\\“、"|"、"-"

```
mpl.rcParams["font.sans-serif"]=["SimHei"]
mpl.rcParams["axes.unicode_minus"]=False

x=[1,2,3,4,5,6,7,8]
y=[3,1,4,5,8,9,7,2]
plt.bar(x,y,align="center",color="c",tick_label=
["q","a","c","e","r","j","b","p"],hatch="/")
plt.xlabel("箱子编号")
plt.ylabel("箱子重量（kg）")
plt.show()
```

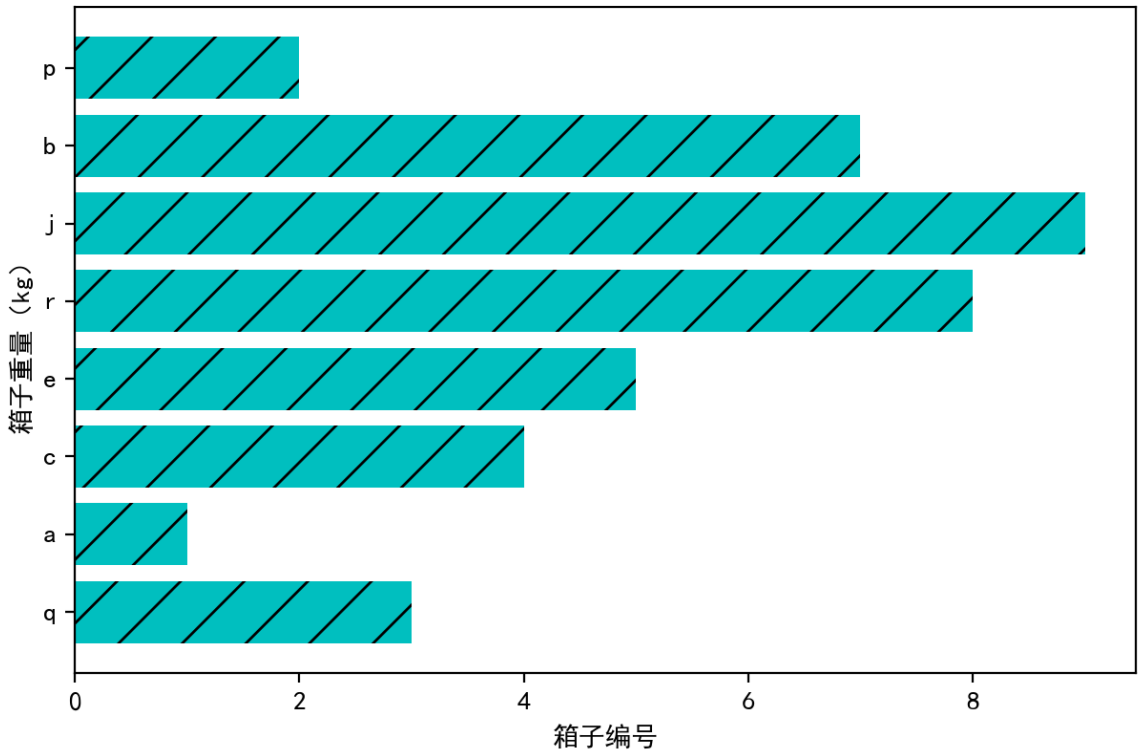


2.2 barh()——绘制条形图

`plt.barh(x,y)`

- x：标示在y轴上的定型数据的类别
- y：每种定性数据的类别的数量
- left：同上bottom

```
mpl.rcParams["font.sans-serif"]=["SimHei"]
mpl.rcParams["axes.unicode_minus"]=False

x=[1,2,3,4,5,6,7,8]
y=[3,1,4,5,8,9,7,2]
plt.barh(x,y,align="center",color="c",tick_label=
["q","a","c","e","r","j","b","p"],hatch="/")
plt.xlabel("箱子编号")
plt.ylabel("箱子重量（kg）")
plt.show()
```

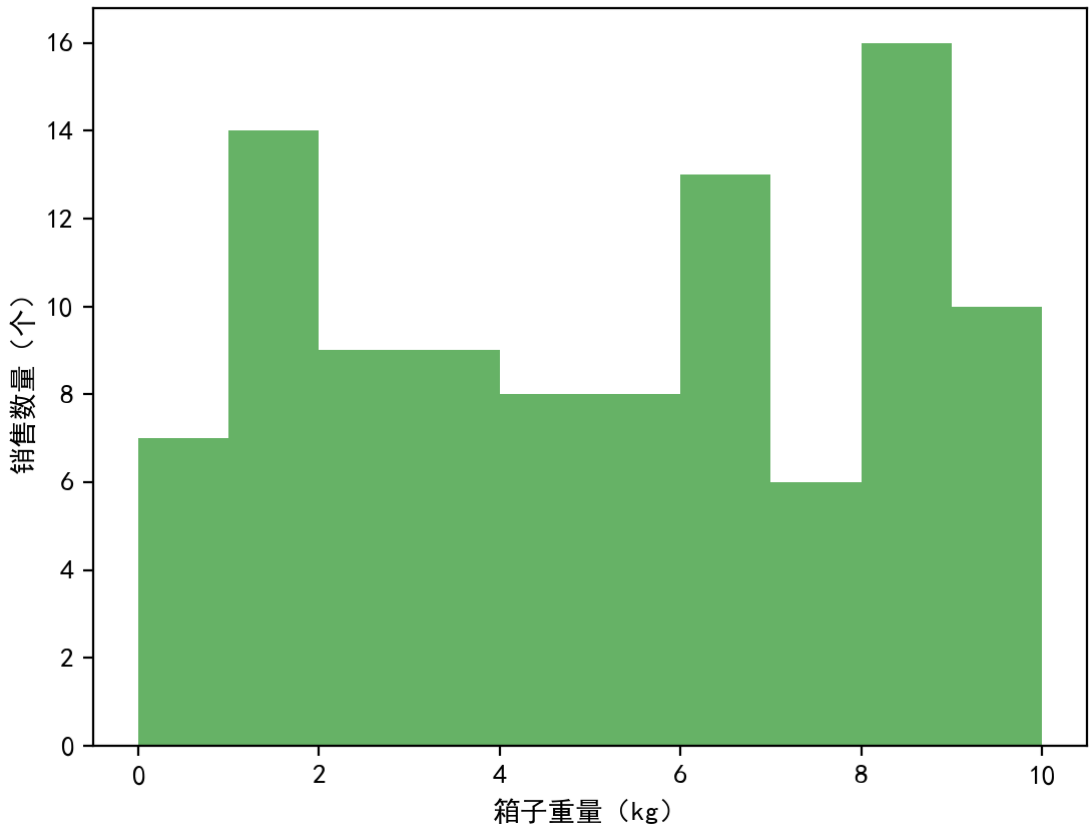


2.3 hist()——绘制直方图

`plt.hist(x,bins=int,color="",density=bool,range=[],bottom= ,histtype="",align="",orientation=' ',log=bool)`

- x：在x轴上绘制箱体的定量数据输入值
- bins：条形数
- density：是否以密度的形式显示
- range：x轴的范围
- bottom：y轴的起始位置
- histtype：线条的类型
  - bar：方形
  - barstacked：柱形
  - step：未填充线条
  - stepfilled：填充线条
- align：对齐方式（left、mid、right）
- orientation：是竖直条还是水平条——"horizontal":水平，"vertical":垂直
- log：单位是否以科学计数法
- label：图例内容
- rwidth：柱体宽度
- stacked：堆积直方图：True；并排放置：False

```
boxweight=np.random.randint(0,10,100)
x=boxweight
bins=range(11)
plt.hist(x,bins=bins,
         color="g",
         histtype="bar",
         rwidth=1,
         alpha=0.6)
plt.ylabel("销售数量（个）")
plt.xlabel("箱子重量（kg）")
plt.show()
```

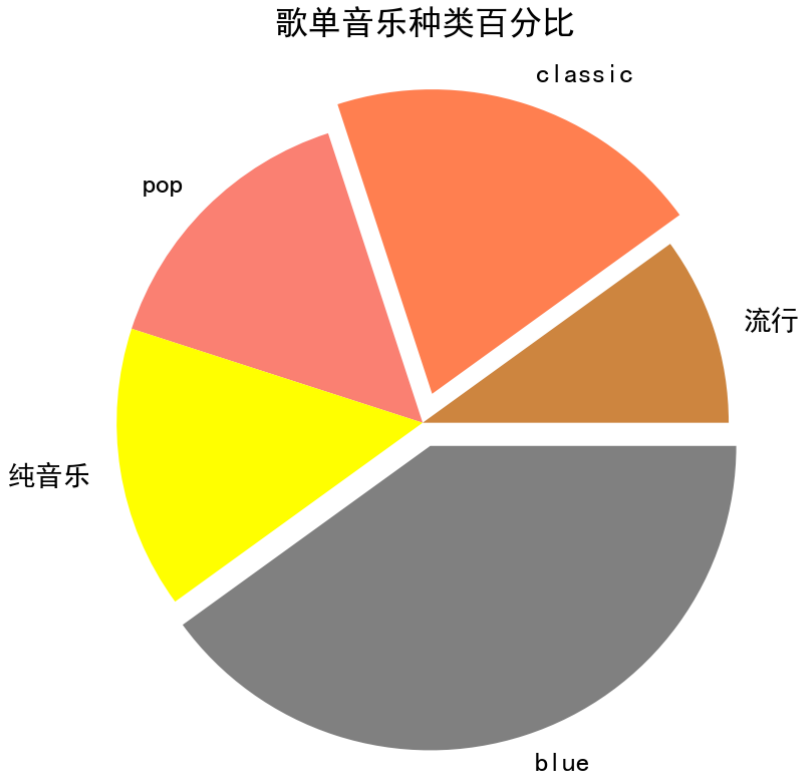


2.4 pie()——绘制饼图

`plt.pie(x,explode=None,labels=None,colors=None, shadow=False, labeldistance=1.1, startangle=None, radius=None)`

- x：为一个存放各部分占比的向量
- explode：list, 每一部分离开中心点的距离 ,元素数目与x相同且一一对应
- labels：list, 设置各类的标签，元素一一对应
- autopct：饼片文本标签内容对应的数值百分比样式
- colors：list, 设置为各部分染色列表，元素一一对应
- startangle：起始绘制角度,默认图是从x轴正方向逆时针画起,如设定=90则从y轴正方向画起
- shadow：显示阴影，默认为False，即不显示阴影
- labeldistance：labels标签位置，相对于半径的比例，默认值为1.1, 如<1则绘制在饼图内侧
- radius：控制饼图半径，默认值为1

```
ratios=[0.1,0.2,0.15,0.15,0.4]#存放比例列表
colors=['peru','coral','salmon','yellow','grey']#存放颜色列表，与比例相匹配
labels=["流行",'classic','pop','纯音乐','blue']#存放各类元素标签
explode=(0,0.1,0,0,0.08)

plt.pie(ratios,explode=explode,colors=colors,labels=labels)#绘制饼图
plt.title('歌单音乐种类百分比')
plt.axis('equal')#将饼图显示为正圆形
plt.show()
```

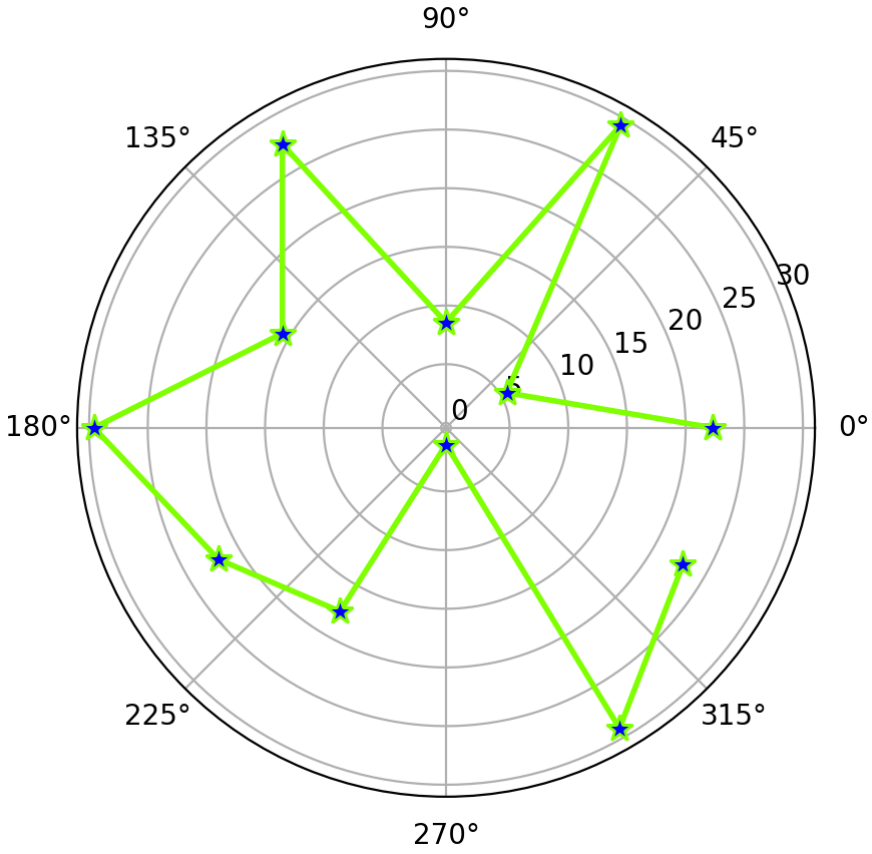


2.5 polar()——绘制极线图

`plt.polar(theta,r)`

- theta：每个标记所在射线与极径的夹角
- r：每个标记到原点的距离

```
barSlices=12
theta=np.linspace(0.0,2*np.pi,barSlices,endpoint=False)
r=30*np.random.rand(barSlices)

plt.polar(theta,r,
          color="chartreuse",
          linewidth=2,
          marker="*",
          mfc="b",
          ms=10)
plt.show()
```

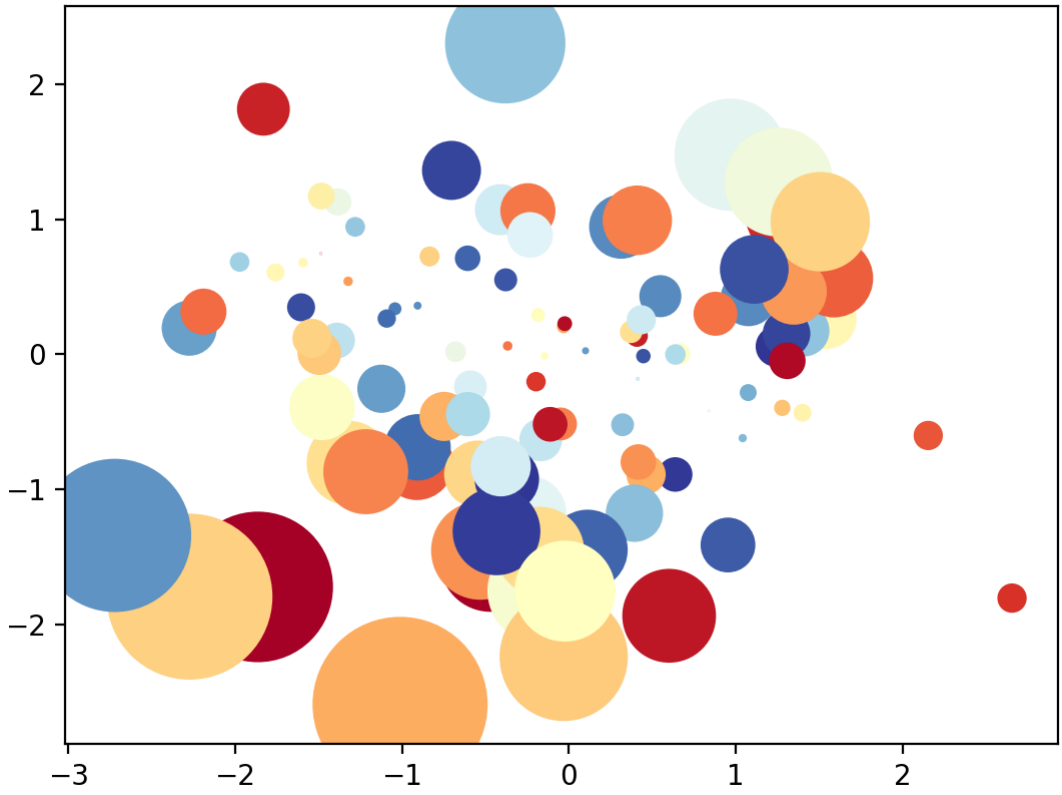


2.6 scatter()——绘制气泡图

`plt.scatter(x,y,s,c,cmap)`

- x：x轴上的数值
- y：y轴上的数值
- s：散点标记的大小
- c：散点标记的yanse
- cmap：将浮点数映射成颜色的颜色映射表

```
a=np.random.randn(100)
b=np.random.randn(100)

plt.scatter(a,b,
            s=np.power(10*a+20*b,2),
            c=np.random.rand(100),
            cmap=mpl.cm.RdYlBu,
            marker="o")
plt.show()
```

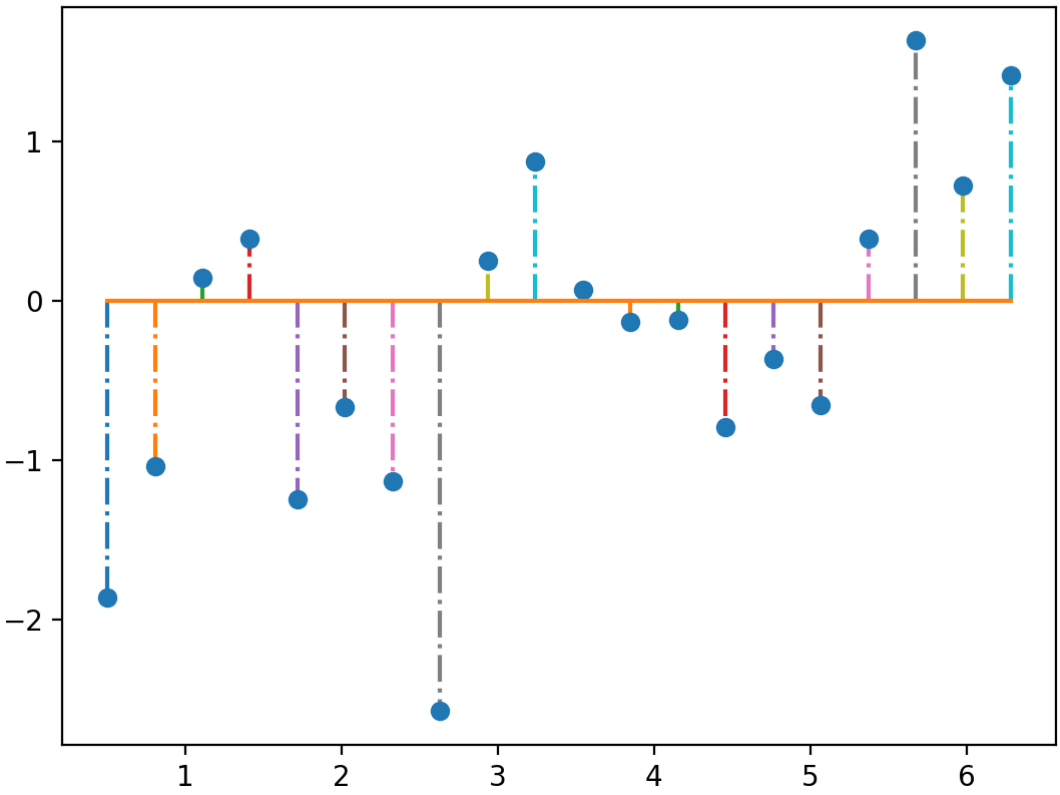


2.7 stem()——绘制棉棒图

`plt.stem(x,y,linefmt,markerfmt,basefmt)`

- x：指定棉棒的x轴基线上的位置
- y：绘制棉棒的长度
- linefmt：棉棒的样式
- markerfmt：棉棒末端的样式
- basefmt：指定基线的样式

```
x=np.linspace(0.5,2*np.pi,20)
y=np.random.randn(20)

plt.stem(x,y,linefmt="-.",markerfmt="o",basefmt="-")
plt.show()
```

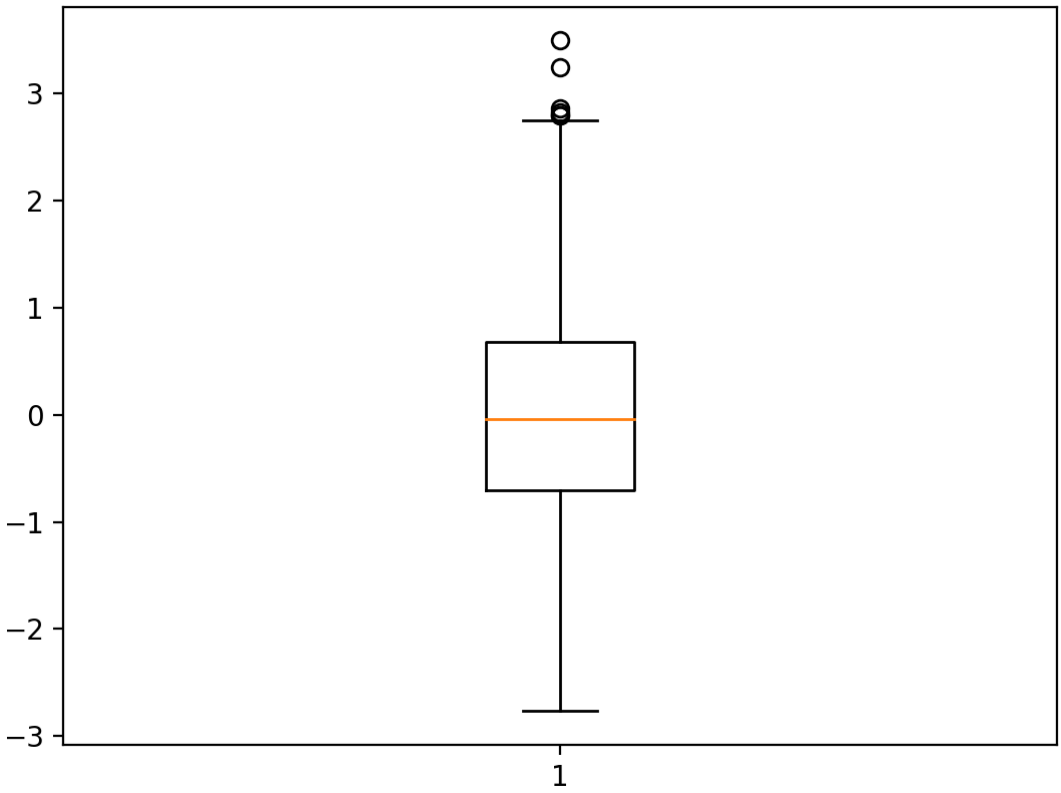


2.8 boxplot()——绘制箱线图

`plt.boxplot(x,notch,sym,vert,patch_artist,showmeans,labels)`

- x：绘制箱线图的输入数据
- whis：四分位间距的倍数，用来确定箱须包含数据的范围的大小
- widths：设置箱体的宽度
- notch：控制箱体图的形状
- sym：控制离群点的样式
- vert：控制箱体的方向
- patch_artist：进行箱体图的颜色填充
- showmeans：显示均值
- labels：指定x轴的坐标

```
x=np.random.randn(1000)

plt.boxplot(x)
plt.show()
```

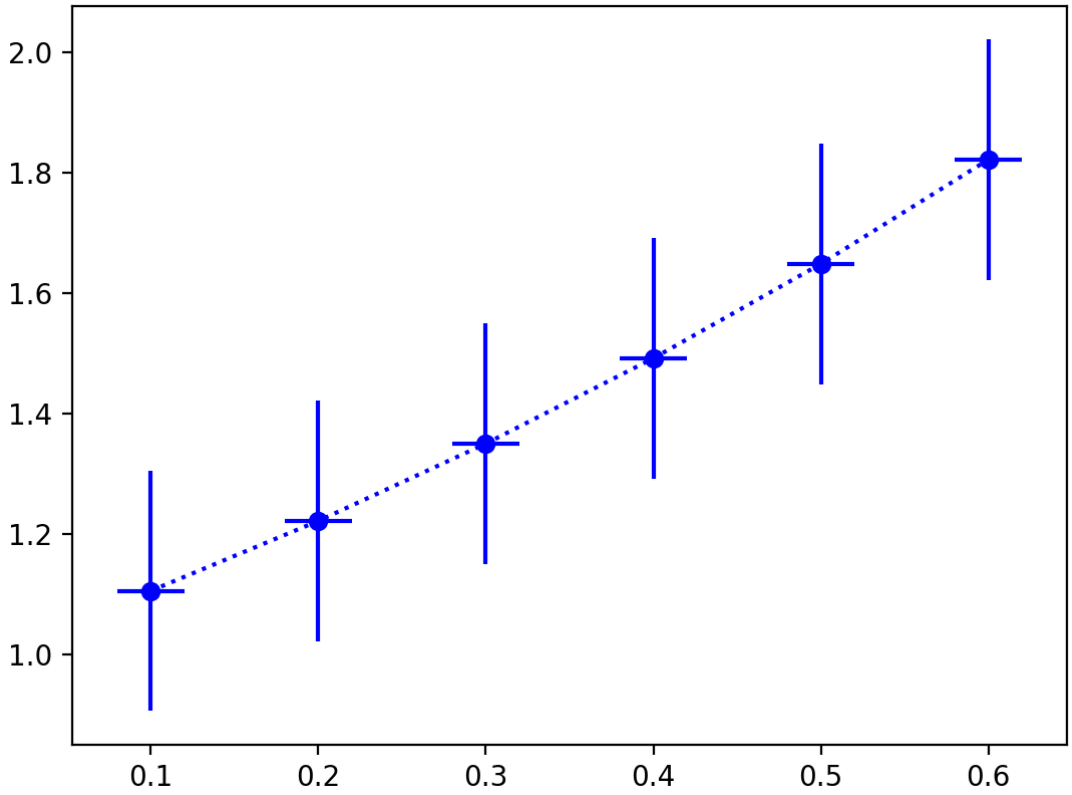


2.9 errorbar()——绘制误差棒图

`plt.errorbar(x,y,yerr=a,xerr=b)`

- x：数据点的水平位置
- y：数据点的垂直位置
- yerr：y轴方向的数据点的误差计算方法
- xerr：x轴方向的数据点的误差计算方法
- fmt: 数据点的标记样式以及相互之间连接线样式
- ecolor: 误差棒的线条颜色
- elinewidth: 误差棒的线条粗细
- capsize: 误差棒边界横杠的大小
- capthick: 误差棒边界横杠的厚度
- ms: 数据点的大小
- mfc: 数据点的颜色
- mec: 数据点边缘的颜色

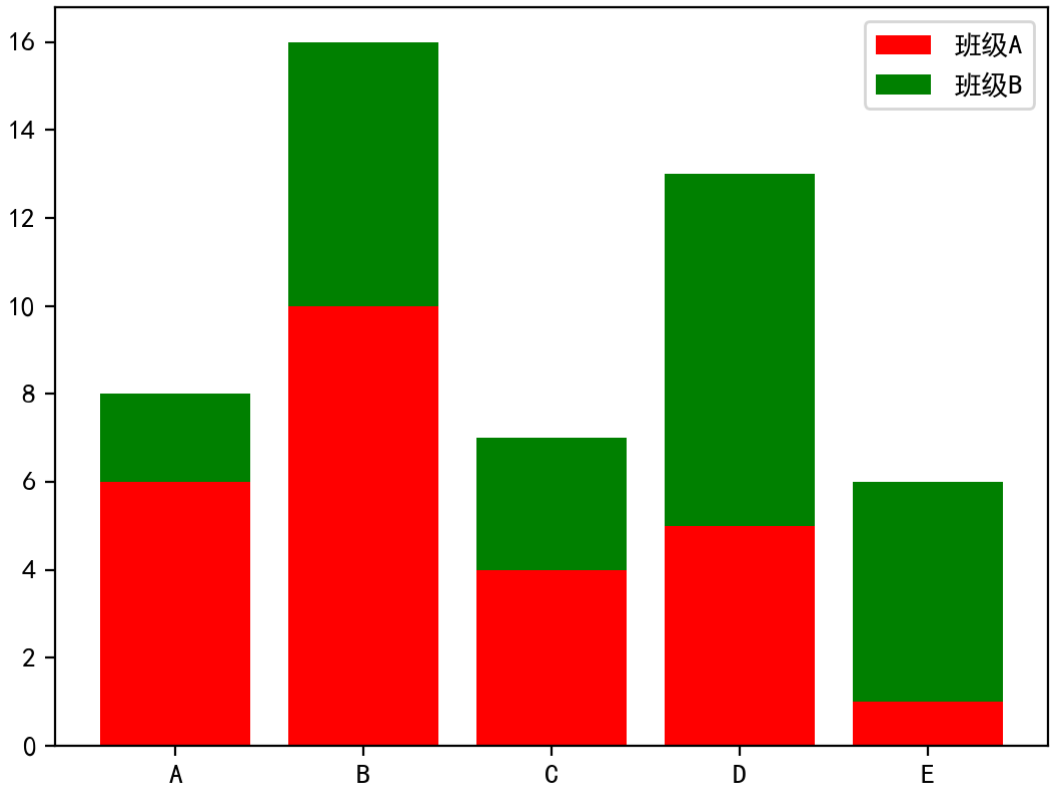
```
x=np.linspace(0.1,0.6,6)
y=np.exp(x)

plt.errorbar(x,y,fmt="bo:",yerr=0.2,xerr=0.02)
plt.show()
```



三、绘制统计图形

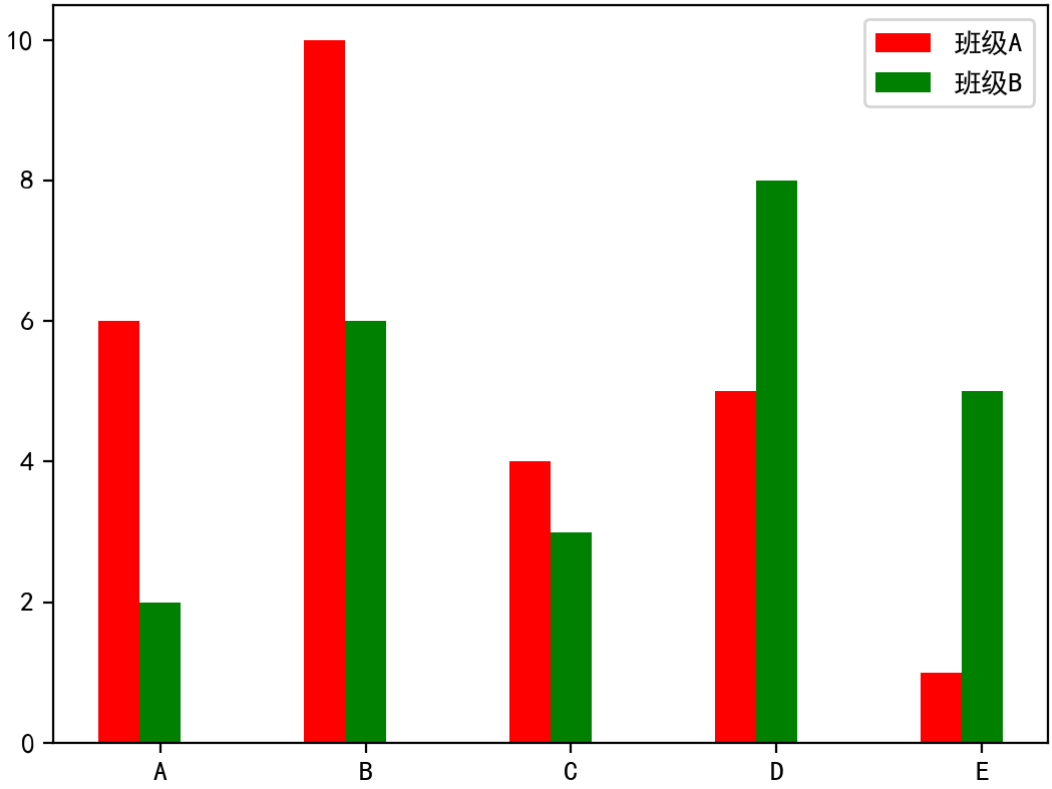
3.3 堆积图

```
x=[1,2,3,4,5]
y=[6,10,4,5,1]
y1=[2,6,3,8,5]

plt.bar(x,y,align="center",color="r",tick_label=["A","B","C","D","E"],label="班级A")
plt.bar(x,y1,align="center",color="g",bottom=y,tick_label=["A","B","C","D","E"],label="班级B")
plt.legend()
plt.show()
```



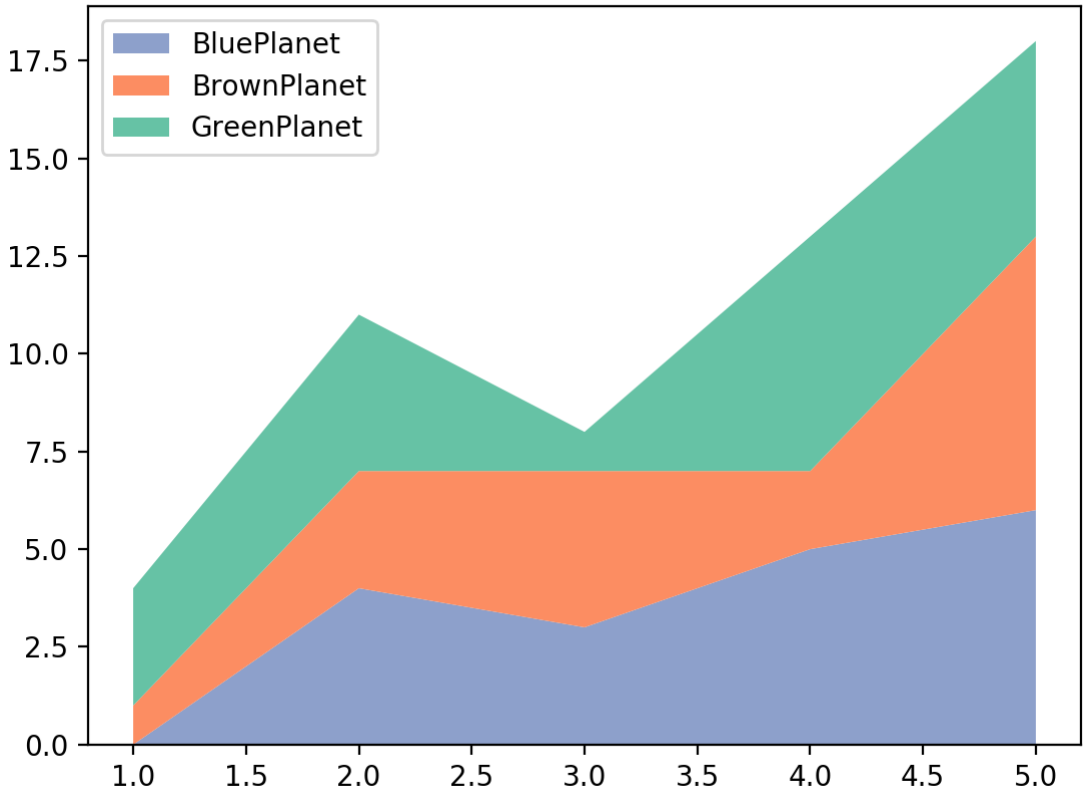
3.4 并列柱状图

```
x=np.arange(5)
y=[6,10,4,5,1]
y1=[2,6,3,8,5]
width=0.2

plt.bar(x,y,width=width,align="center",color="r",
        tick_label=["A","B","C","D","E"],label="班级A")
plt.bar(x+width,y1,width=width,align="center",color="g",
        tick_label=["A","B","C","D","E"],label="班级B")
plt.legend()
plt.show()
```



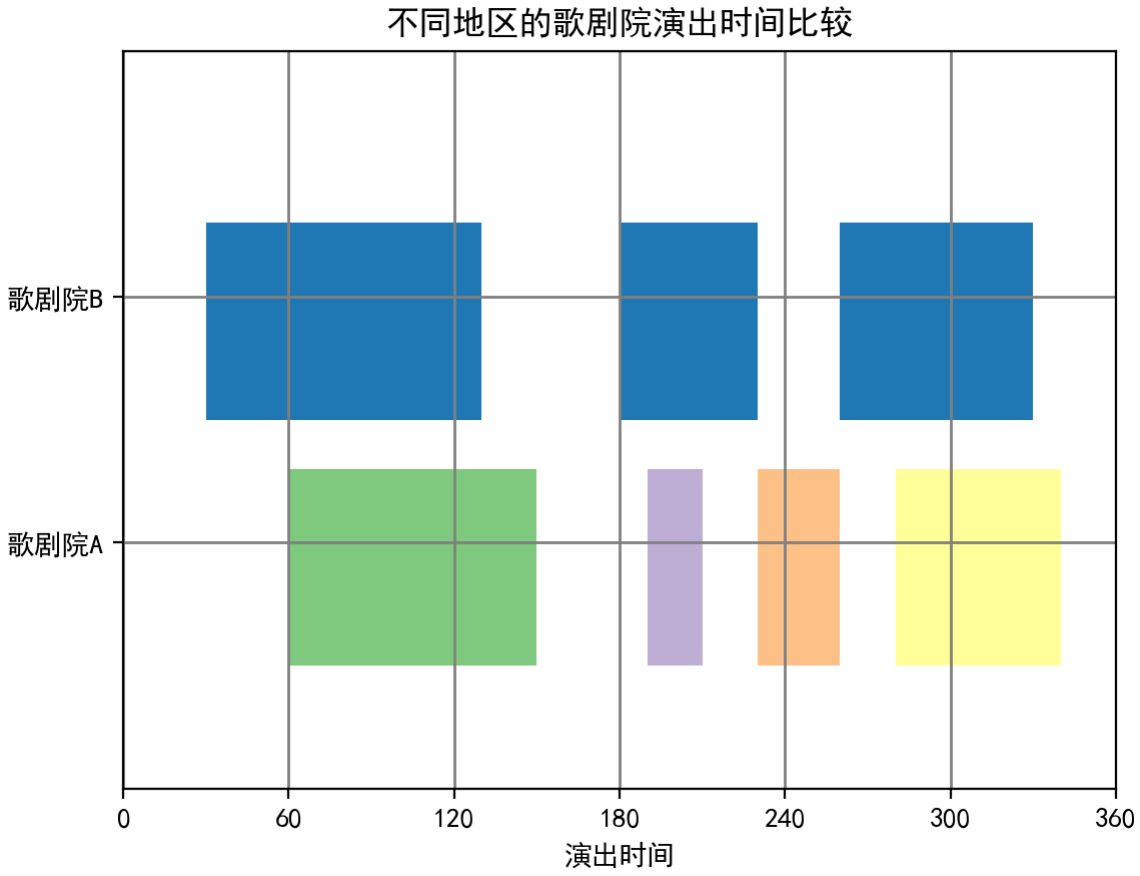
3.6.1 stackplot()——绘制堆积折线图

```
x = np.arange(1,6,1)
y = [0,4,3,5,6]
y1 = [1,3,4,2,7]
y2 = [3,4,1,6,5]

labels = ["BluePlanet","BrownPlanet","GreenPlanet"]
colors = ["#8da0cb","#fc8d62","#66c2a5"]

plt.stackplot(x,y,y1,y2,labels=labels,colors=colors)
plt.legend(loc="upper left")
plt.show()
```



3.6.2 broken_barh()——绘制间断条形图

```
mpl.rcParams["font.sans-serif"] = ["SimHei"]
mpl.rcParams["axes.unicode_minus"] = False

plt.broken_barh([(30,100),(180,50),(260,70)],(20,8),facecolors="#1f78b4")
plt.broken_barh([(60,90),(190,20),(230,30),(280,60)],(10,8),facecolors=
("#7fc97f","#beaed4","#fdc086","#ffff99"))

plt.xlim(0,360)
plt.ylim(5,35)
plt.xlabel("演出时间")
plt.xticks(np.arange(0,361,60))
plt.yticks([15,25],["歌剧院A","歌剧院B"])
plt.grid(ls="-",lw=1,color="gray")
plt.title("不同地区的歌剧院演出时间比较")

plt.show()
```

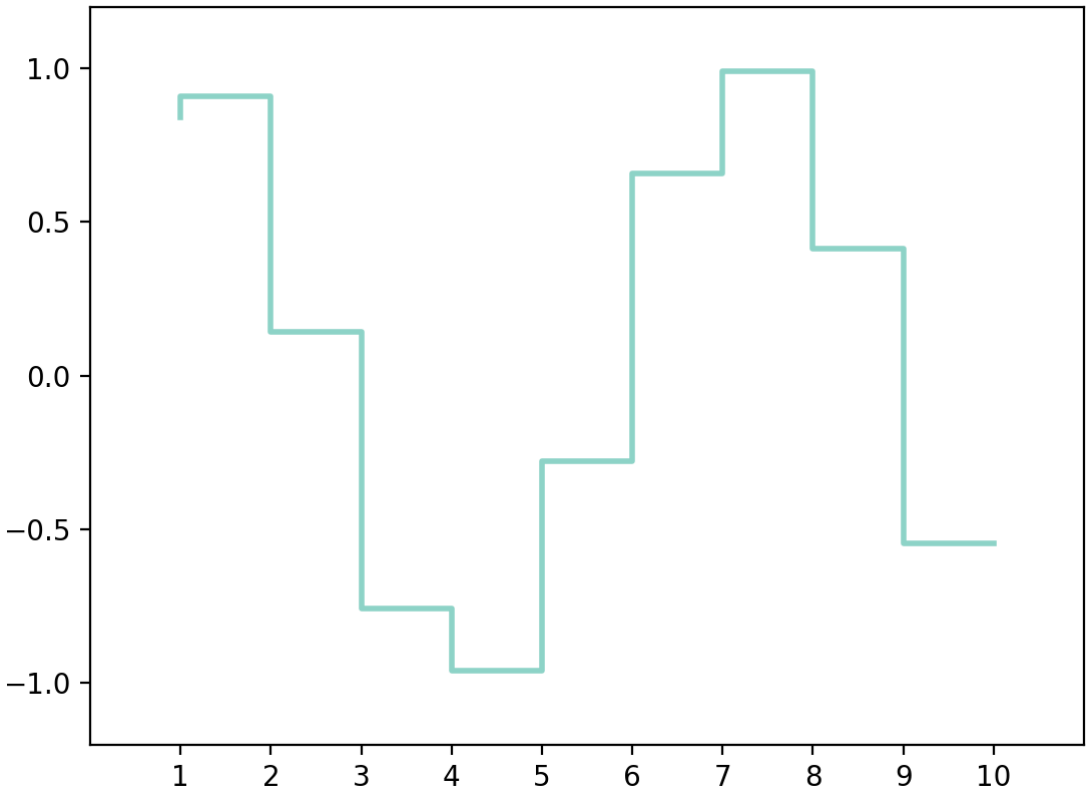


3.6.3 step()——绘制阶梯图

`plt.step(x, y, *args, data=None, **kwargs)`

- where: pre、post 或 mid，即数据点处于阶梯的右侧、左侧还是正中

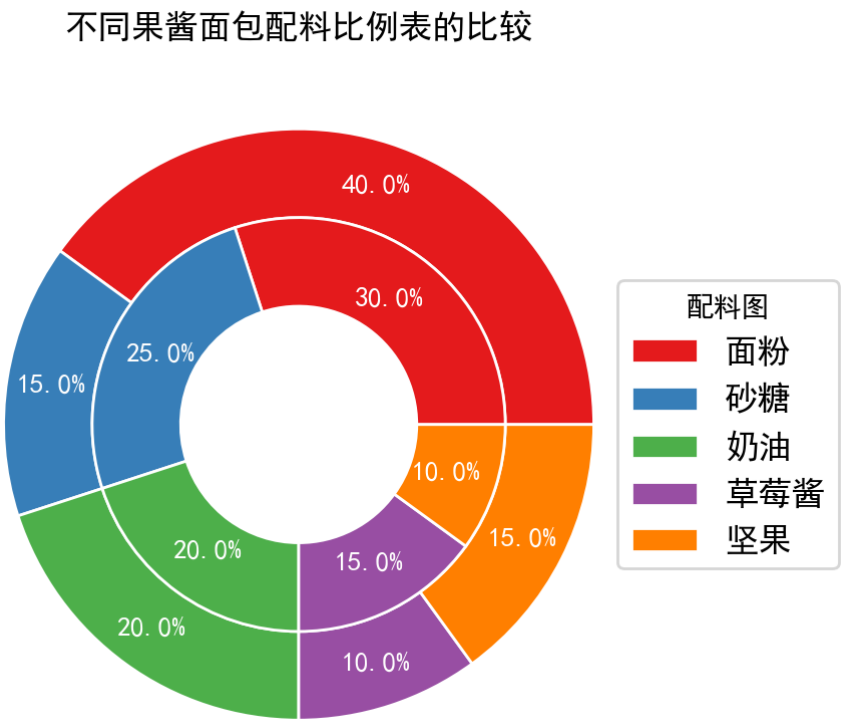
```
x = np.linspace(1,10,10)
y = np.sin(x)

plt.step(x,y,color="#8dd3c7", where="pre",lw=2)
plt.xlim(0,11)
plt.xticks(np.arange(1,11,1))
plt.ylim(-1.2,1.2)
plt.show()
```



3.8.4 绘制内嵌环形饼图

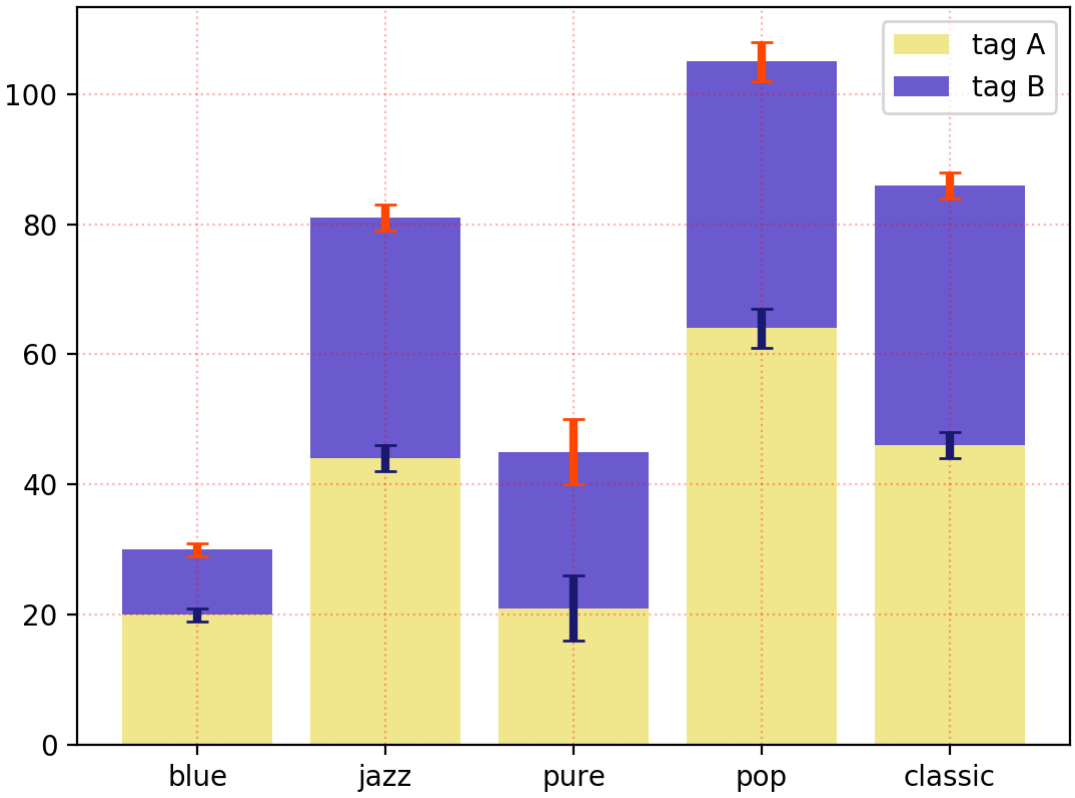
```
elements = ["面粉", "砂糖", "奶油", "草莓酱", "坚果"]

weight1 = [40, 15, 20, 10, 15]
weight2 = [30, 25, 20, 15, 10]

colormapList = ["#e41a1c", "#377eb8", "#4daf4a", "#984ea3", "#ff7f00"]
outer_colors = colormapList
inner_colors = colormapList

wedges1, texts1, autotexts1 = plt.pie(weight1,
                                      autopct="%3.1f%%",
                                      radius = 1,
                                      pctdistance=0.85,
                                      colors=outer_colors,
                                      textprops=dict(color="w"),
                                      wedgeprops=dict(width=0.3, edgecolor="w"))

wedges2, texts2, autotexts2 = plt.pie(weight2,
                                      autopct="%3.1f%%",
                                      radius = 0.7,
                                      pctdistance=0.75,
                                      colors=outer_colors,
                                      textprops=dict(color="w"),
                                      wedgeprops=dict(width=0.3, edgecolor="w"))

plt.legend(wedges1,
           elements,
           fontsize=12,
           title="配料图",
           loc="center left",
           bbox_to_anchor=(0.91, 0, 0.3, 1))
```

```
plt.setp(autotexts1, size=10, weight="bold")
plt.setp(autotexts2, size=10, weight="bold")
plt.setp(texts1, size=12)
plt.title("不同果酱面包配料比例表的比较")
plt.show()
```



3.10.3 带误差棒的柱状图

```
x=np.arange(5)
y1=[20,44,21,64,46]
y2=[10,37,24,41,40]
#误差列表
std_err1=[1,2,5,3,2]
std_err2=[2,4,3,1,2]
tick_label=['blue','jazz','pure','pop','classic']

error_params1=dict(elinewidth=3,ecolor='midnightblue',capsize=4)#设置误差标记参数
error_params2=dict(elinewidth=3,ecolor='orangered',capsize=4)#设置误差标记参数

#绘制柱状图，设置误差标记以及柱状图标签
plt.bar(x,y1,color='khaki',yerr=std_err1,error_kw=error_params1,label='tag A')
plt.bar(x,y2,bottom=y1,color='slateblue',yerr=std_err1,error_kw=error_params2,la
bel='tag B')

plt.xticks(x,tick_label)#设置x轴的标签
#设置网格
plt.grid(True,axis='both',ls=':',color='r',alpha=0.3)
plt.legend()
plt.show()
```



四、完善统计图形

4.2.2 subplot()——子区函数

`plt.subplot(xyz)`

- x：行数
- y：列数
- z：第几个

4.3 table()——添加表格

`plt.table(cellText,)`

- cellText：表格的数值，将源数据按照行进行分组，每组数组放在列表例存储，所有组数据再放在列表里存储
- cellLoc：表格中的数据对齐位置，可以左、居中、右对齐
- colWidths：表格每列的宽度
- colLabels：表格每列的列名称
- colColours：表格每列的列名称所在单元格的颜色
- rowLabels：表格每行的行名称
- rowLoc：表格每行的行名称对齐位置
- loc：表格再画布中的位置

五、图形样式

5.1 设置坐标轴的刻度样式

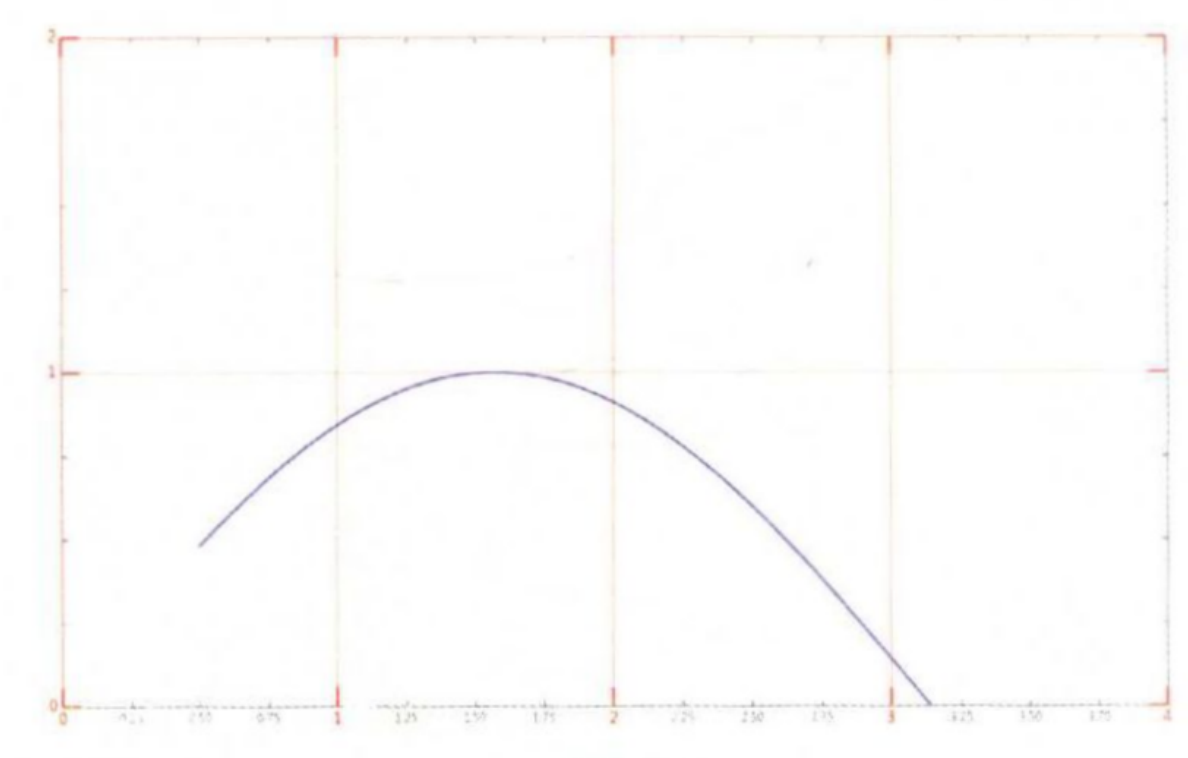
- 定位器（locator）：设置刻度线的位置
  - `AutoLocator`：`MaxNLocator`使用简单的默认值。这是大多数绘图的默认刻度线定位器。
  - `MaxNLocator`：在最合适的位置找到带有刻度的最大间隔数。
  - `LinearLocator`：空间从最小到最大均匀滴答。
  - `LogLocator`：空间从最小到最大以对数形式滴答。
  - `MultipleLocator`：刻度和范围是基数的倍数；整数或浮点数。
  - `FixedLocator`：刻度线位置是固定的。
  - `IndexLocator`：索引图的定位符（例如where ）。`x = range(len(y))`
  - `NullLocator`：没有滴答声。
  - `SymmetricalLogLocator`：用于符号规范的定位器；类似于`LogLocator`超出阈值的部分，如果在限制之内，则加0。
  - `LogitLocator`：用于logit缩放的定位器。
  - `OldAutoLocator`：选择一个`MultipleLocator`并动态重新分配它，以在导航期间进行智能打勾。
  - `AutoMinorLocator`：轴为线性且主刻度线等距分布时，副刻度线的定位器。将主刻度间隔细分为指定数量的次间隔，根据主间隔默认为4或5。
- 格式器（formatter）：设置刻度标签的显示样式
  - `NullFormatter`：刻度线上没有标签。
  - `IndexFormatter`：从标签列表中设置字符串。
  - `FixedFormatter`：手动设置标签的字符串。
  - `FuncFormatter`：用户定义的功能设置标签。
  - `StrMethodFormatter`：使用字符串`format`方法。
  - `FormatStrFormatter`：使用旧式的sprintf格式字符串。
  - `ScalarFormatter`：标量的默认格式化程序：自动选择格式字符串。
  - `LogFormatter`：日志轴的格式化程序。
  - `LogFormatterExponent`：使用来格式化日志轴的值。`exponent = log_base(value)`
  - `LogFormatterMathtext`：使用“数学”文本格式化对数轴的值。`exponent = log_base(value)`
  - `LogFormatterSciNotation`：使用科学计数法设置对数轴的值格式。
  - `LogitFormatter`：概率格式器。
  - `EngFormatter`：以工程标记格式格式化标签。
  - `PercentFormatter`：将标签格式化为百分比。

例：

```
x=np.linspace(0.5,3.5,100)
y=np.sin(x)

fig=plt.figure(figsize=(8,8))
ax=fig.add_subplot(111)

ax.xaxis.set_major_locator(MultipleLocator(1.0))     #在x轴的1倍处设置主刻度线
ax.yaxis.set_major_locator(MultipleLocator(1.0))

ax.xaxis.set_minor_locator(AutoMinorLocator(4))      #设置次刻度线的显示位置
ax.yaxis.set_minor_locator(AutoMinorLocator(4))

def minor_tick(x,pos):
    if not x%4.0:
        return ""
```

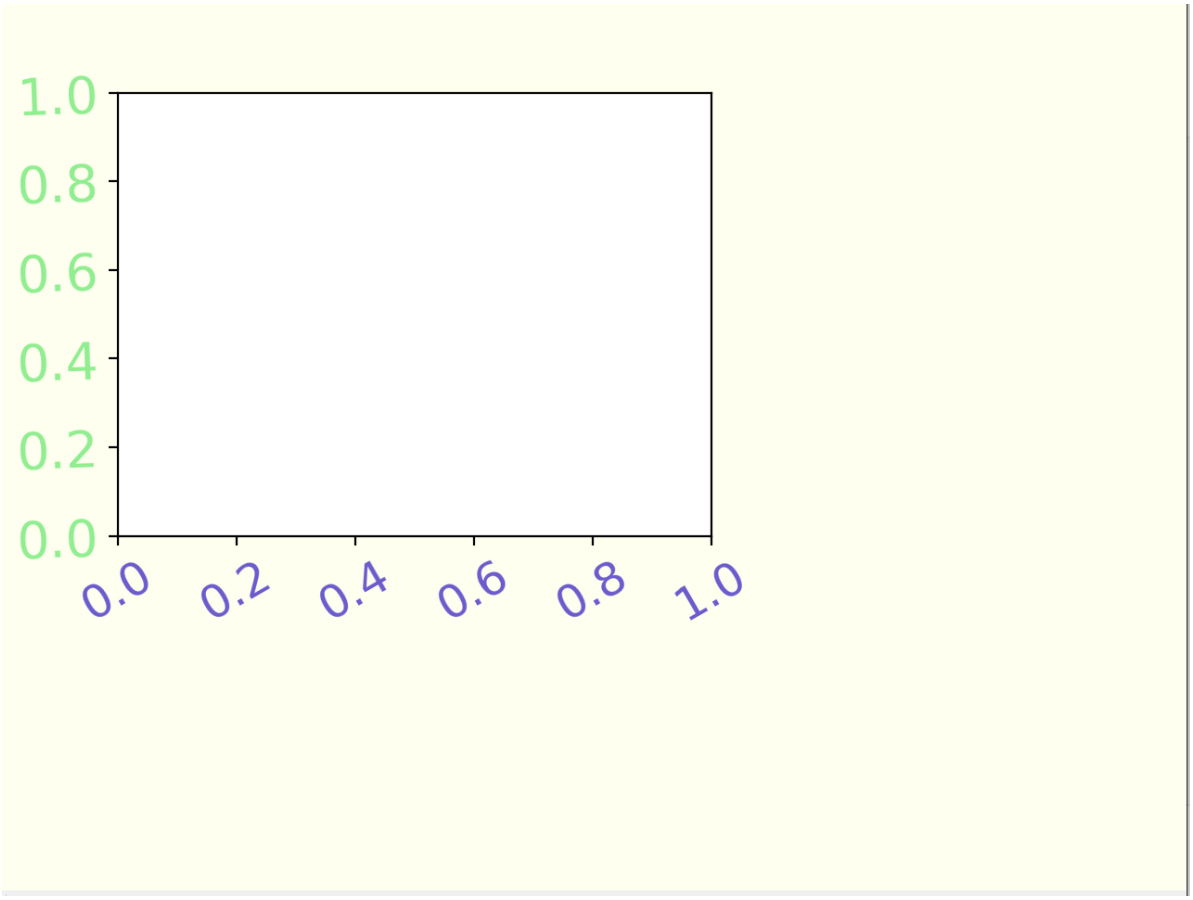
```
    return "%.2f"%x
ax.xaxis.set_minor_formatter(FuncFormatter(minor_tick)) #次刻度线显示位置的精度

ax.tick_params("y",which='major',            #主刻度样式
               length=15,width=2.0,
               colors='r')
ax.tick_params(which='minor',                #次刻度样式
               length=5,width=1.0,
               labelcolor='0.25',labelsize=10)

ax.set_xlim(0.4)
ax.set_ylim(0.2)

ax.plot(x,y,lw=2,zorder=10)
ax.grid(width=0.5,color='r')
plt.show()
```

`axis`	**{'x', 'y', 'both'}, optional 坐标轴；默认 'both'**
reset	bool, default: False 恢复初始设置
`which`	{'major', 'minor', 'both'} 主、副刻度线；默认 'major'
`direction`	{'in', 'out', 'inout'} 绘图区内侧、外侧和同时
`length`	float 刻度线长度
`width`	float 刻度线宽度
`color`	color 刻度线颜色
`labelcolor`	color 标签颜色
`labelsize`	刻度标签大小
`colors`	color 标签和刻度线颜色
`pad`	float 标签和刻度线的距离



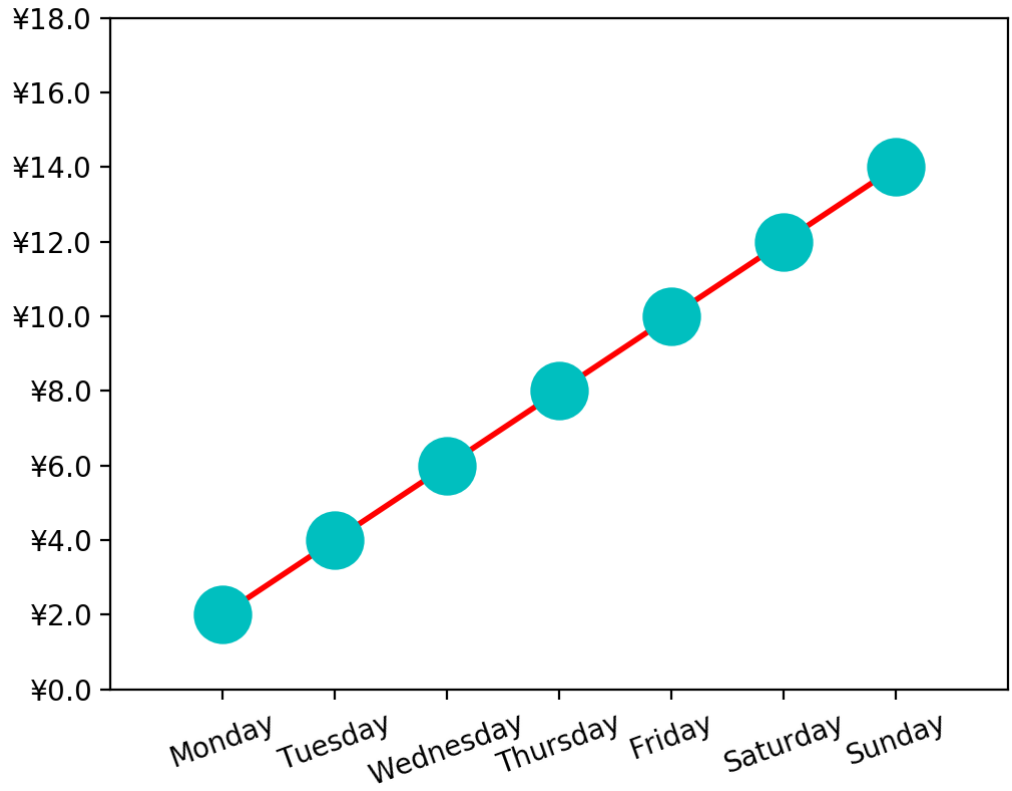
5.1.3 刻度标签和刻度线样式的定制化

```
fig=plt.figure(facecolor=(1.0,1.0,0.9412))
    #ax1 = fig.add_axes([left, bottom, width, height])
ax=fig.add_axes([0.1,0.4,0.5,0.5])

for ticklabel in ax.xaxis.get_ticklabels():
    ticklabel.set_color("slateblue")
    ticklabel.set_fontsize(18)
    ticklabel.set_rotation(30)

for ticklabel in ax.yaxis.get_ticklabels():
    ticklabel.set_color("lightgreen")
    ticklabel.set_fontsize(20)
    ticklabel.set_rotation(2)

plt.show()
```



5.1.4 货币和时间序列的刻度

```
from calendar import month_name,day_name
from matplotlib.ticker import FormatStrFormatter

fig=plt.figure()
ax=fig.add_axes([0.2,0.2,0.7,0.7])

x=np.arange(1,8,1)
y=2*x

ax.plot(x,y,ls='-',lw=2,color='r',marker='o',ms=20,mfc='c',mec='c')

ax.yaxis.set_major_formatter(FormatStrFormatter(r"$\yen%1.1f$"))
plt.xticks(x,day_name[0:7],rotation=20)
ax.set_xlim(0,8)
ax.set_ylim(0,18)
plt.show()
```



5.2 annotate()——有指示注解和无指示注解

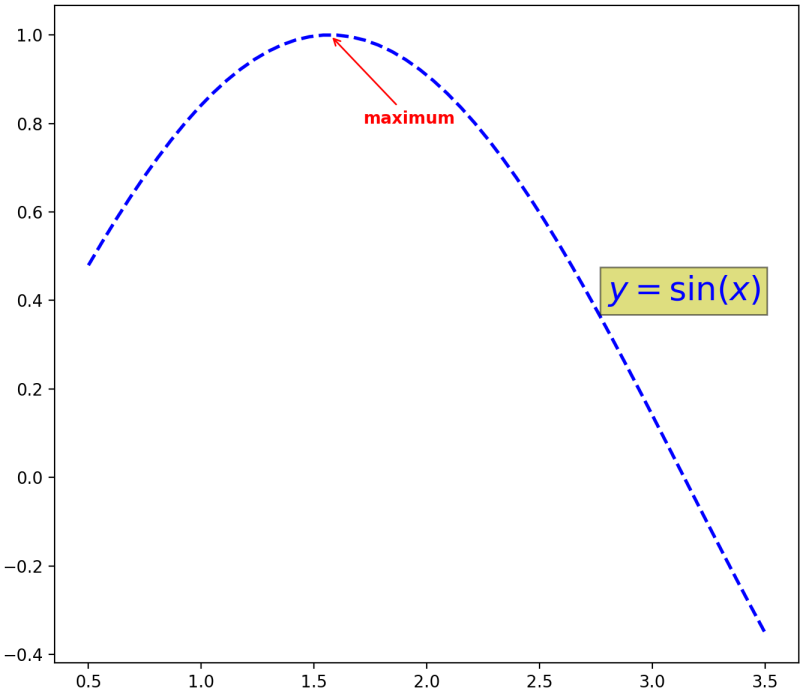
有指示注解：`annotate()`

- string：图形内容的注释文本
- xy：被注释图形内容的位置坐标
- xycoords：xy的坐标系统，参数值data表示与折线图使用相同的坐标系统
- xytext：注释文本的位置坐标
- textcoords：xytext的坐标系统
- weight：注释文本的字体粗细风格
- color：注释文本的字体颜色
- arrowprops：指示被注释内容的箭头的属性字典

无指示注解：`ax.text()`

- x,y：注解的横纵坐标
- s：注解内容

```
fig=plt.figure(figsize=(8,8))
ax=fig.add_subplot(111)
ax.plot(x,y,c='b',ls='--',lw=2)

ax.annotate("maximum",xy=(np.pi/2,1.0),xycoords='data',
            xytext=((np.pi/2)+0.15,0.8),textcoords="data",
            weight="bold",color='r',
            arrowprops=dict(arrowstyle="->",
                            connectionstyle="arc3",
                            color='r'))

ax.text(2.8,0.4,"$y=\sin(x)$",fontsize=20,color='b',
        bbox=dict(facecolor='y',alpha=0.5))
plt.show()
```



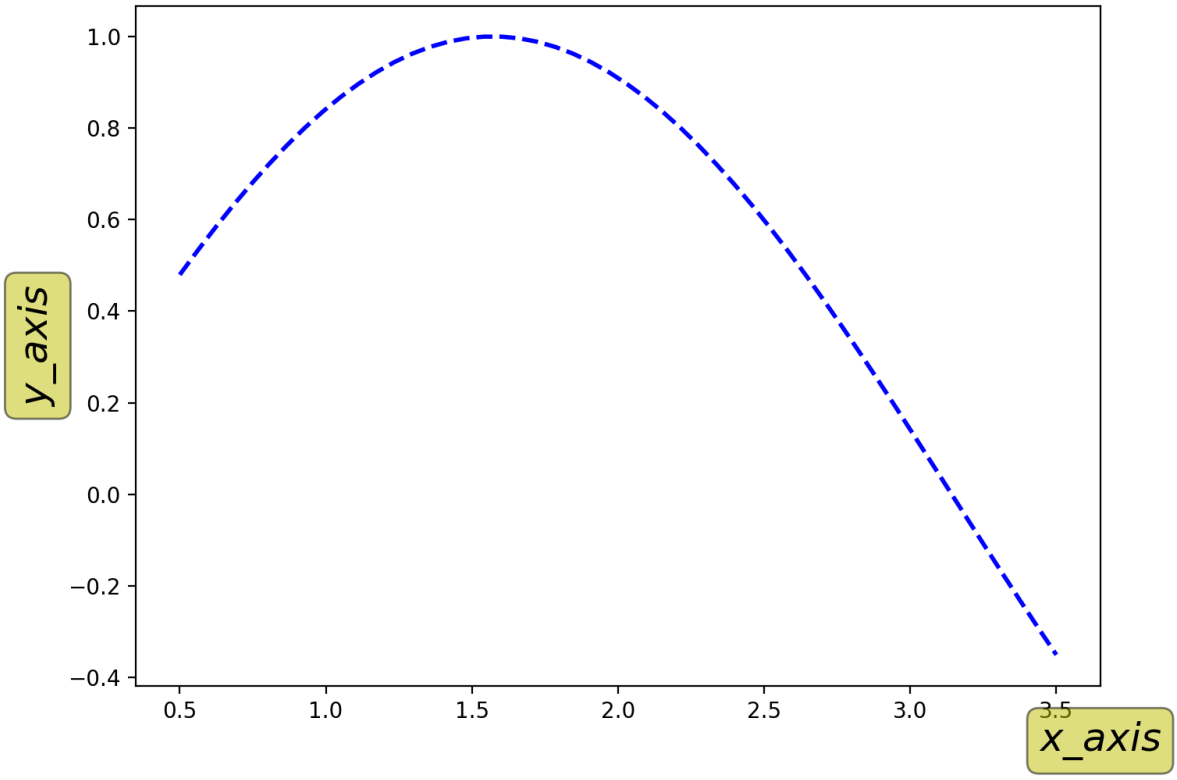
5.2.5 arrow()——有箭头指示的趋势线

`arrow()`

- x, y : 箭头起点坐标
- dx, dy : 箭头x上的长度和y轴上的长度
- width: 箭头宽度，默认0.001
- length_includes_head: bool，箭"头"是否包含在长度之中 默认False
- head_width: float,箭"头"的宽度，默认: 3*width
- head_length: float 箭"头"的长度，默认1.5 * head_width
- shape: [‘full’, ‘left’, ‘right’]，箭头形状， 默认 ‘full’
- overhang: float (default: 0)
- head_starts_at_zero: bool (default: False)开始坐标是否是0

5.3.2 bbox——给坐标轴标签添加文本框

```
bbox=dict(boxstyle="round",facecolor='y',alpha=0.5)
ax.set_xlabel("$x\_axis$",fontsize=18,bbox=bbox)     #坐标轴文本
ax.set_ylabel("$y\_axis$",fontsize=18,bbox=bbox)

ax.xaxis.set_label_coords(1.0,-0.05)                 #坐标轴位置
ax.yaxis.set_label_coords(-0.08,0.5)
```



六、划分画布

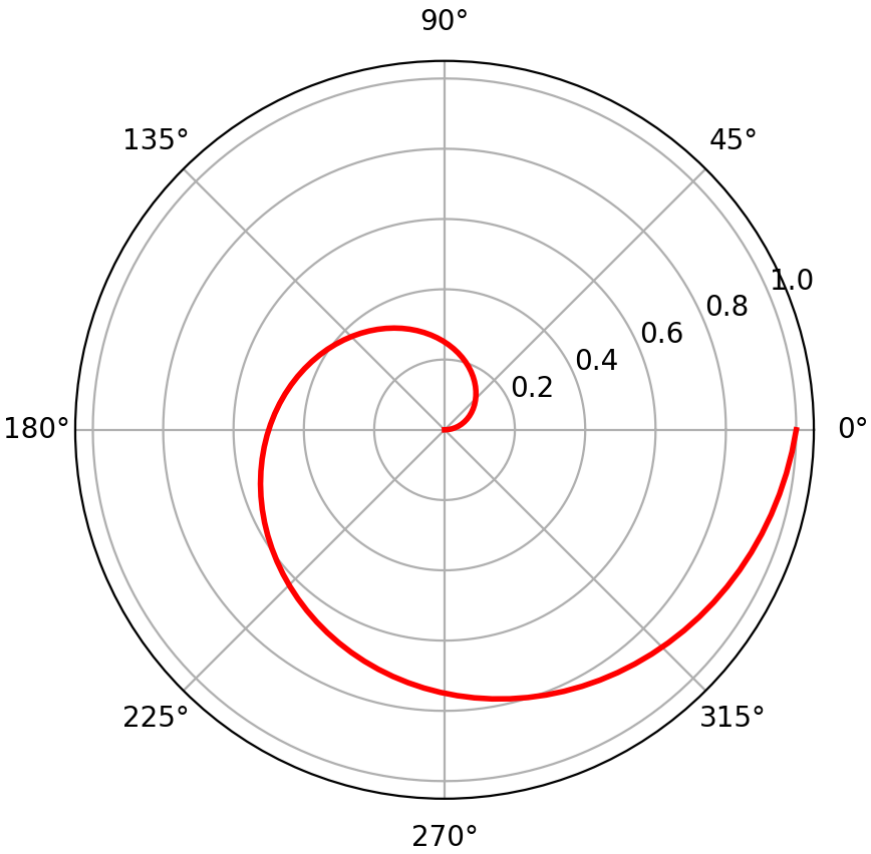
6.1.2 subplot()——子区布局

`subplot(CRP)`

- C：C行
- R：R列
- P：P个位置
- projection：子图的投影类型(`Axes`). *斯塔尔*是自定义投影的名称，请参见`projections`。默认的None结果是“直线”投影。{None, 'aitoff', 'hammer', 'lambert', 'mollweide', 'polar', 'rectilinear', str},
- polar：若为真，则等于投影=“极”
- sharex, sharey：该轴将具有与共享轴的轴相同的限制、刻度和刻度
- label：返回的轴的标签

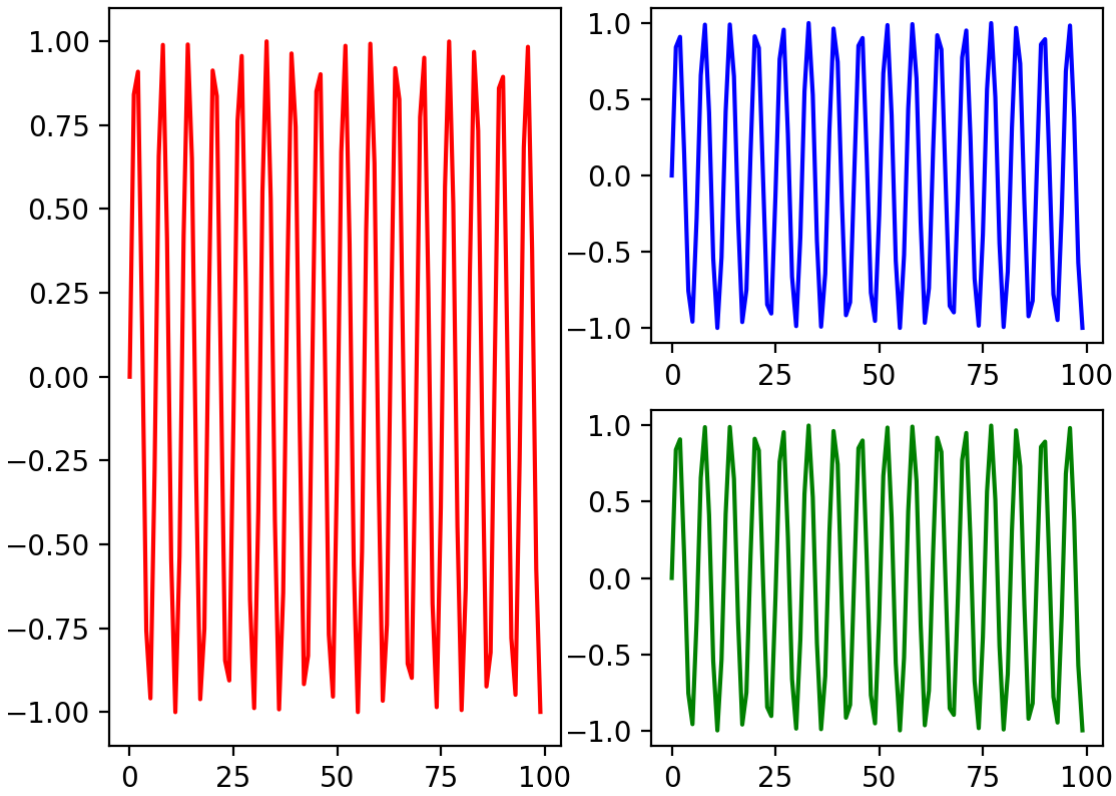
例：极坐标绘图

```
radii=np.linspace(0,1,100)
theta=2*np.pi*radii

ax=plt.subplot(111,polar=True)
ax.plot(theta,radii,color="r",linestyle="-",linewidth=2)
plt.show()
```



6.1.4 在非等画布的绘图区域

```
x=np.arange(100)
y=np.sin(x)
fig=plt.figure()

ax1=fig.add_subplot(121)
ax1.plot(x,y,ls="-",color="r")

ax2=fig.add_subplot(222)
ax2.plot(x,y,ls="-",color="b")

ax3=fig.add_subplot(224)
ax3.plot(x,y,ls="-",color="g")
plt.show()
```

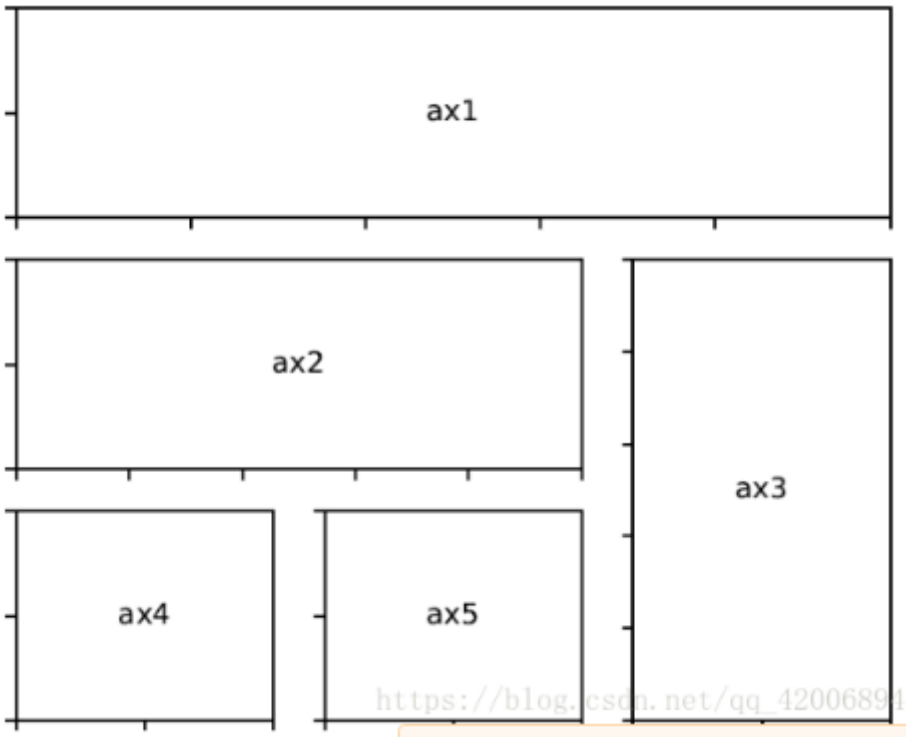


6.2.1 subplot2grid()——子区跨越固定的网格布局

`subplot2grid((x,y),(xi,yi),colspan,rowspan)`

- (x,y)：总行，总列
- (xi,yi)：起始坐标
- colspan：横向占用多少行
- rowspan：纵向占用多少行

```
ax1 = plt.subplot2grid((3,3), (0,0), colspan=3)
ax2 = plt.subplot2grid((3,3), (1,0), colspan=2)
ax3 = plt.subplot2grid((3,3), (1, 2), rowspan=2)
ax4 = plt.subplot2grid((3,3), (2, 0))
ax5 = plt.subplot2grid((3,3), (2, 1))
```

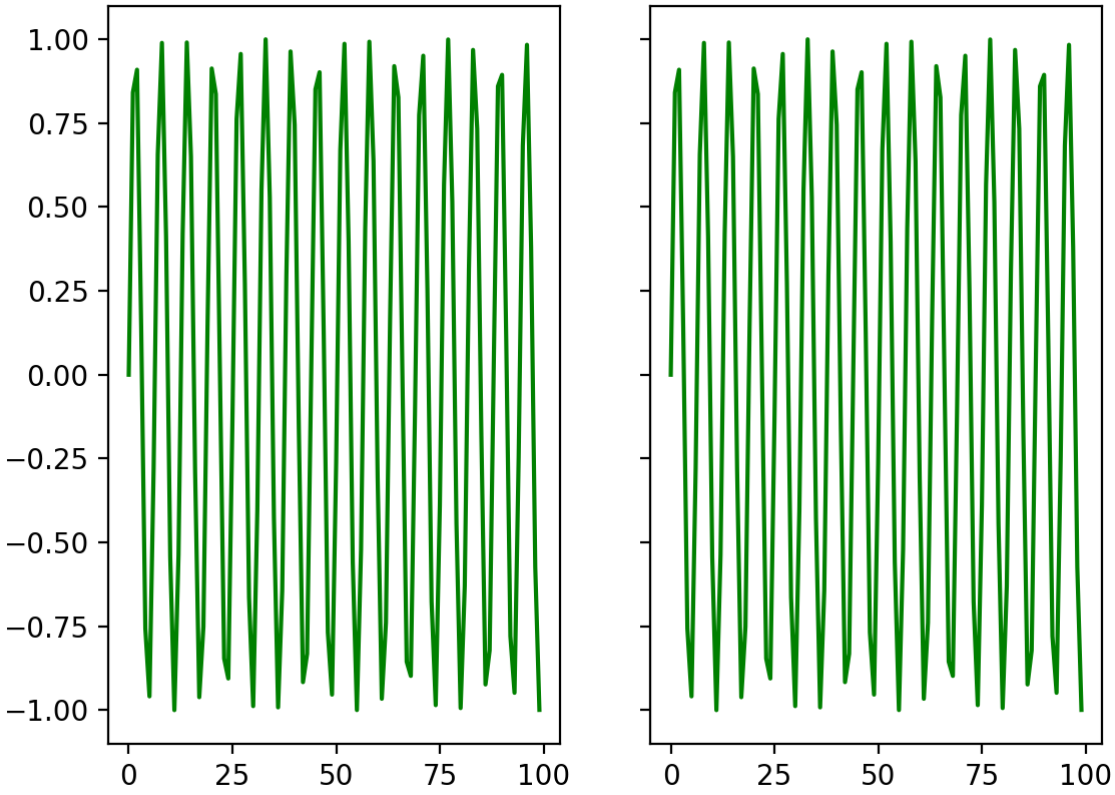


6.3 subplots()——创建一张画布带有多个子区

`subplots(nrows=1, ncols=1, *, sharex=False, sharey=False, squeeze=True, subplot_kw=None, gridspec_kw=None)`：返回一个（fig，ax）的元组，fig是Figure实例；ax是axis对象

- nrows, ncols：子图网格的行/列数。
- sharex, sharey：控制x或y轴之间的属性共享：
- squeeze：
  - 如果为True，则从返回的Axes
    - 如果只构造一个子图(nrow=ncols=1)，则生成的单轴对象作为标量返回。
    - 对于NX1或1xm子图，返回的对象是一个由Axis对象组成的一维Numpy对象数组。
    - 对于NxM，N>1和M>1的子图作为二维数组返回。
  - 如果为false，则根本不进行压缩：返回的AXIS对象始终是一个包含轴实例的2D数组，即使它最终为1x1。
- subplot_kw：将关键字传递给`add_subplot`用于创建每个子图的调用。
- gridspec_kw：将关键字传递给`GridSpec`构造函数，用于创建子图所在的网格。

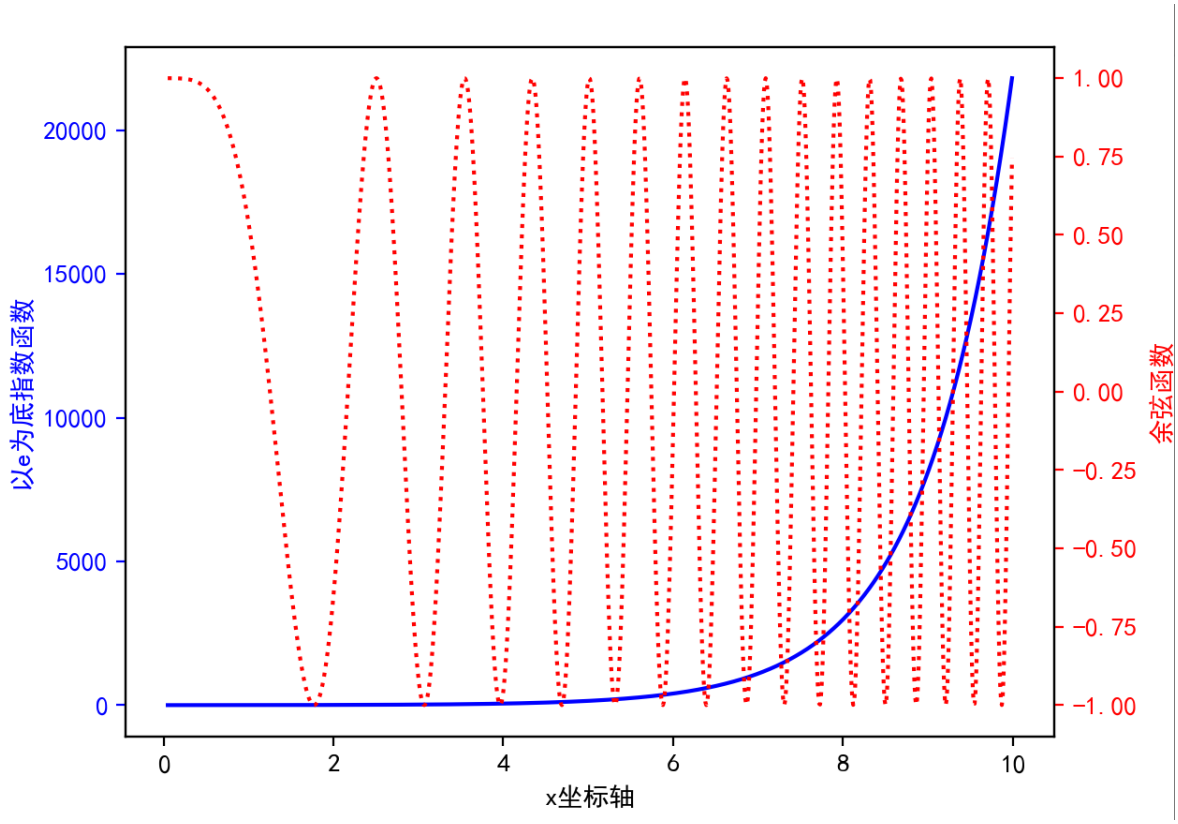
```
fig,ax=plt.subplots(1,2,sharey=True)

ax1=ax[0]
ax1.plot(x,y,ls="-",color="g")

ax2=ax[1]
ax2.plot(x,y,ls="-",color="g")
plt.show()
```



七、共享绘图区域的坐标轴

7.1 twinx()——共享单一绘图区域的坐标轴

```
fig,ax1=plt.subplots()
t=np.arange(0.05,10.0,0.01)
s1=np.exp(t)
ax1.plot(t,s1,c="b",ls="-")

ax1.set_xlabel("x坐标轴")

ax1.set_ylabel("以e为底指数函数",color="b")     #y轴标签
ax1.tick_params("y",colors="b")                 #y轴刻度线

ax2=ax1.twinx()          #生成实例ax2，共享x轴，twiny()

s2=np.cos(t**2)
ax2.plot(t,s2,c="r",ls=":")
ax2.set_ylabel("余弦函数",color="r")
ax2.tick_params("y",colors="r")

plt.show()
```



7.2 共享不同子区绘图区域的坐标轴

使用`subplot()`中的sharex、sharey参数。

- sharex="all"：等同于True，所有子区的x轴一样
- sharex="none"：等同于False，默认参数，x轴各不一样
- sharex="row"：子区中每一行的图形x轴取值范围相同，选择每一行x轴取值范围最大的最为共享范围
- sharex="col"：子区中每一列的图形x轴取值范围相同，选择每一列x轴取值范围最大的最为共享范围

7.2.2 subplots_adjust()——去除子区之间的空隙

```
fig,ax=plt.subplots(4,1,sharex="all")
```

`fig.subplots_adjust(hspace=0)`——水平方向的孔隙去除

- left = 0.125：图片中子图的左侧
- right = 0.9：图片中子图的右侧
- bottom = 0.1：图片中子图的底部
- top = 0.9：图片中子图的顶部
- wspace = 0.2：为子图之间的空间保留的宽度，平均轴宽的一部分
- hspace = 0.2：为子图之间的空间保留的高度，平均轴高度的一部分

7.3.2 autoscale()——调整坐标轴范围

```
autoscale(enable=True,axis="both",tight=True)
```

- enable：进行坐标轴范围的自适应调整
- axis：是x、y轴都进行自适应调整
- tight：让坐标轴的范围调整到数据的范围上

在子区的代码部分加入这句话就可以

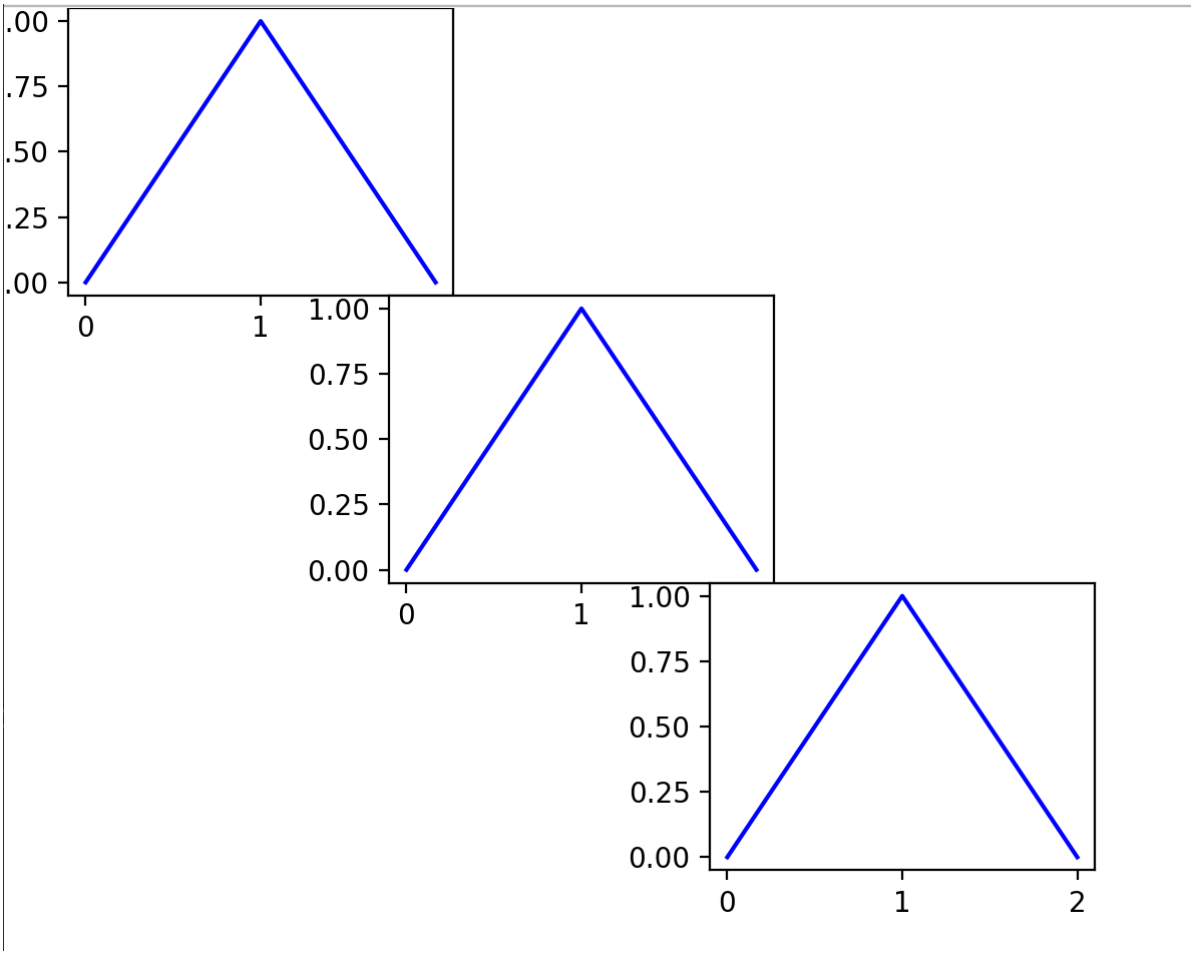
八、坐标轴高级应用

8.1.1 axes()——向画布任意位置添加任意数量坐标轴

`plt.axes(rect,frameon=True,axisbg='y')`

- rect：[left,bottom,width,height]
- frameon：为True时，会绘制坐标轴的四条轴脊
- axisbg：填充坐标轴背景的颜色

```
plt.axes([0.05,0.7,.3,.3],frameon=True)
plt.plot(np.arange(3),[0,1,0],color="b")

plt.axes([0.3,0.4,.3,.3],frameon=True)
plt.plot(np.arange(3),[0,1,0],color="b")

plt.axes([0.55,0.1,.3,.3],frameon=True)
plt.plot(np.arange(3),[0,1,0],color="b")
```

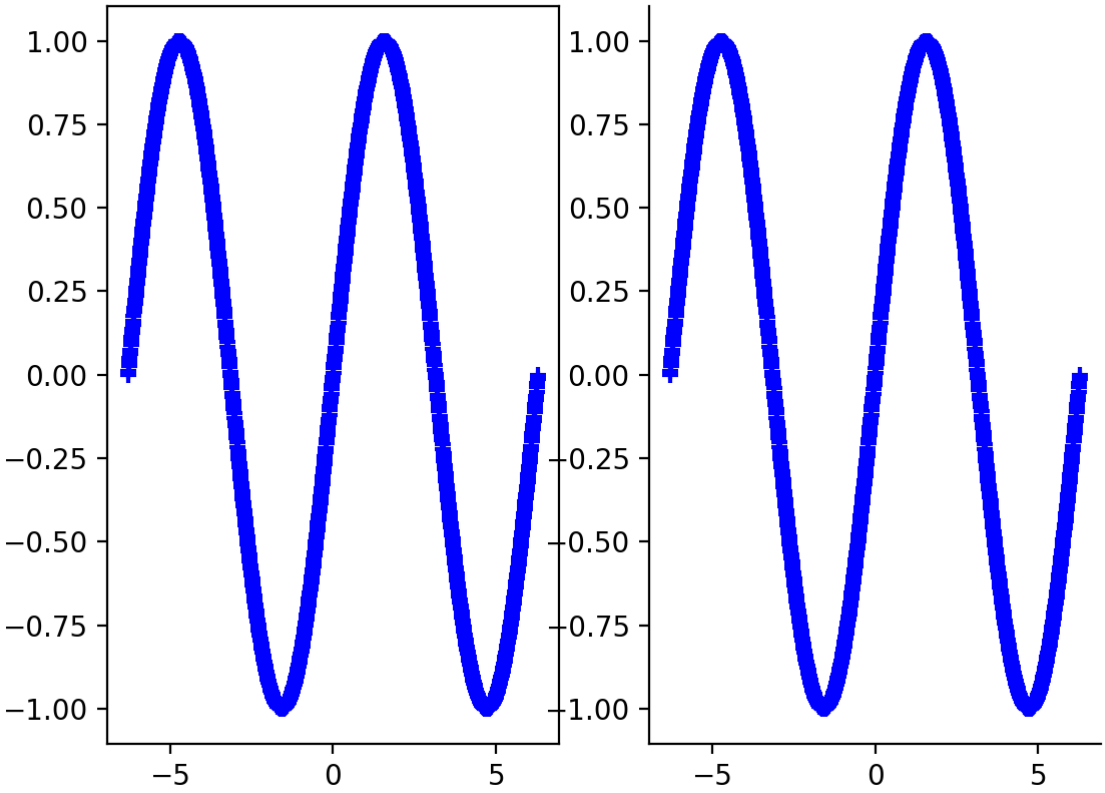


8.1.2 axis()——隐藏坐标轴

- plt.axis("image")：使画面更紧凑
- plt.axis([xmin,xmax,ymin,ymax])：重新改变坐标轴范围
- plt.axis("off")：隐藏坐标轴

8.3 spines()——控制坐标轴的显示

```
ax1=plt.subplot(121)
plt.scatter(x,y,marker="+",color="b")

ax2=plt.subplot(122)
ax2.spines["right"].set_color("none")          #将顶边框和右边框去掉
ax2.spines["top"].set_color("none")
ax2.xaxis.set_ticks_position("bottom")         #将顶边框和右边框的刻度线去掉
ax2.yaxis.set_ticks_position("left")

plt.scatter(x,y,marker="+",color="b")
```



8.4 移动坐标轴位置

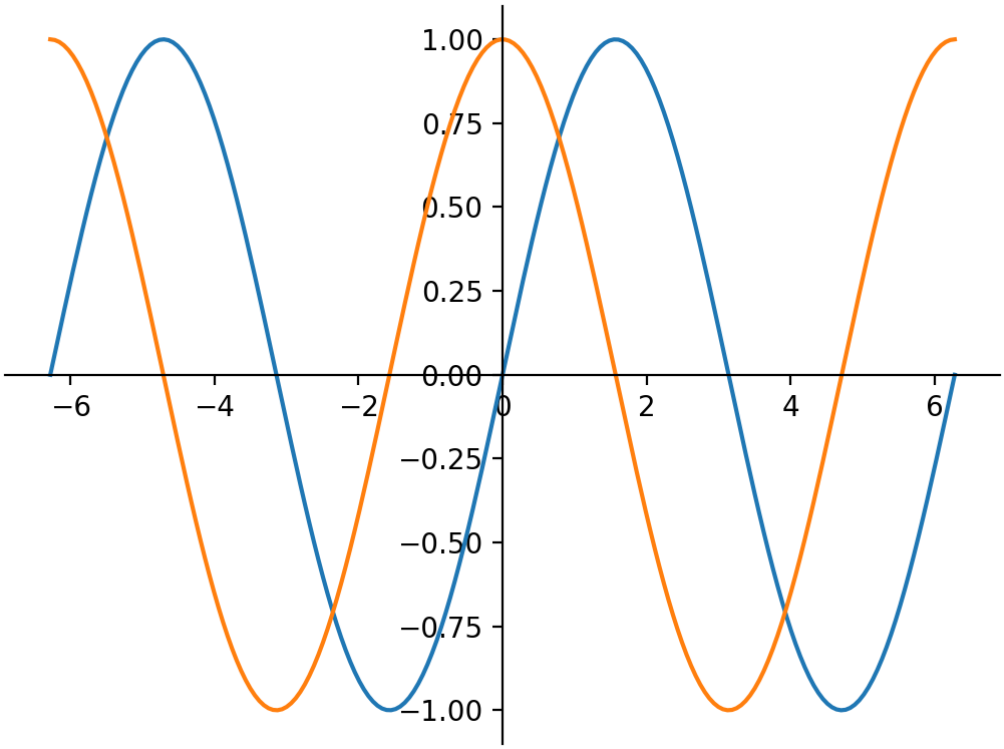
```
x=np.linspace(-2*np.pi,2*np.pi,200)
y1=np.sin(x)
y2=np.cos(x)

ax=plt.subplot(111)
ax.plot(x,y1,ls="-")
ax.plot(x,y2,ls="-")

ax.spines["right"].set_color("none")
ax.spines["top"].set_color("none")
ax.spines["bottom"].set_position(("data",0))
ax.spines["left"].set_position(("data",0))
ax.xaxis.set_ticks_position("bottom")
ax.yaxis.set_ticks_position("left")
plt.show()
```

ax.spines会调用轴脊自带你，其中键是轴脊位置[top、right、bottom、left]

set_position()就是对轴脊位置的控制方法

其中data说明控制轴脊位置的坐标值与折线图的坐标系统一致。

因此，参数0就表示将底端轴脊移动到左侧轴脊的零点处

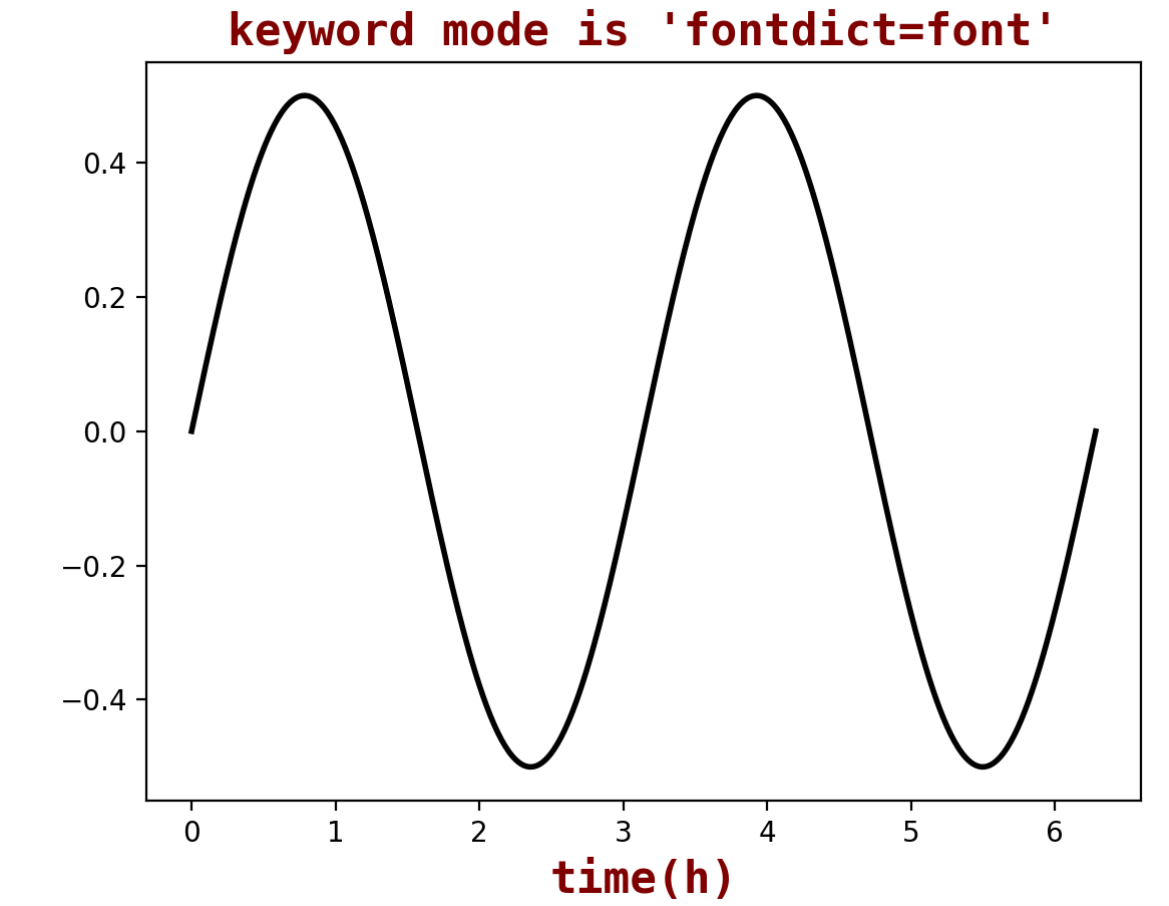


九、设置线条类型和标记类型的显示样式

9.1 字典调用参数

- fontdict=font:
- **font:

```
fig=plt.figure()
ax=fig.add_subplot(111)

font={"family":"monospace","color":"maroon","weight":"bold","size":16}

x=np.linspace(0.0,2*np.pi,500)
y=np.cos(x)*np.sin(x)

ax.plot(x,y,color="k",ls="-",lw=2)
ax.set_title("keyword mode is 'fontdict=font'",fontdict=font)
ax.set_xlabel("time(h)",**font)
plt.show()
```



11.1 设置字体属性和文本属性

- 改变配置文件matploylibrc
- 通过属性字典rcParams

```
plt.rcParams["font.family"] = "serif"
plt.rcParams["font.serif"] = "New Century Schoolbook"
plt.rcParams["font.style"] = "normal"
plt.rcParams["font.variant"] = "small-caps"
plt.rcParams["font.weight"] = "black"
plt.rcParams["font.size"] = 12.0
```

- 通过设置函数的关键字参数

```
plt.title("Line Chart",color="red",family="New Century Schoolbook",
        style="normal",variant="small-caps",weight="black",size=18)
```

12.2.5 颜色标尺

`plt.colorbar`

13.2 保存图形

- 使用按钮保存
- 使用代码保存:

`plt.savefig(fname, dpi=None, facecolor='w', edgecolor='w', orientation='portrait', papertype=None, format=None, transparent=False, bbox_inches=None, pad_inches=0.1, frameon=None, metadata=None)`

- fname: （字符串或者仿路径或仿文件）如果格式已经设置，这将决定输出的格式并将文件按fname来保存。如果格式没有设置，在fname有扩展名的情况下推断按此保存，没有扩展名将按照默认格式存储为“png”格式，并将适当的扩展名添加在fname后面。
- dpi: 分辨率，每英寸的点数
- facecolor（颜色或“auto”，默认值是“auto”）：图形表面颜色。如果是“auto”，使用当前图形的表面颜色。
- edgecolor（颜色或“auto”，默认值：“auto”）：图形边缘颜色。如果是“auto”，使用当前图形的边缘颜色。
- orientation – {‘landscape,’ ‘portrait’}: 目前只有后端支持。.
- format（字符串）：文件格式，比如“png”，“pdf”，“svg”等，未设置的行为将被记录在fname中。
- papertype: papertypes可以设置为“a0到a10”， “executive,” “b0 to b10”, “letter,” “legal,” “ledger.”
- bbox_inches: 只有图形给定部分会被保存。设置为“tight”用以恰当的匹配所保存的图形。
- pad_inches: (默认: 0.1)所保存图形周围的填充量。
- transparent: 用于将图片背景设置为透明。图形也会是透明，除非通过关键字参数指定了表面颜色和/或边缘颜色。