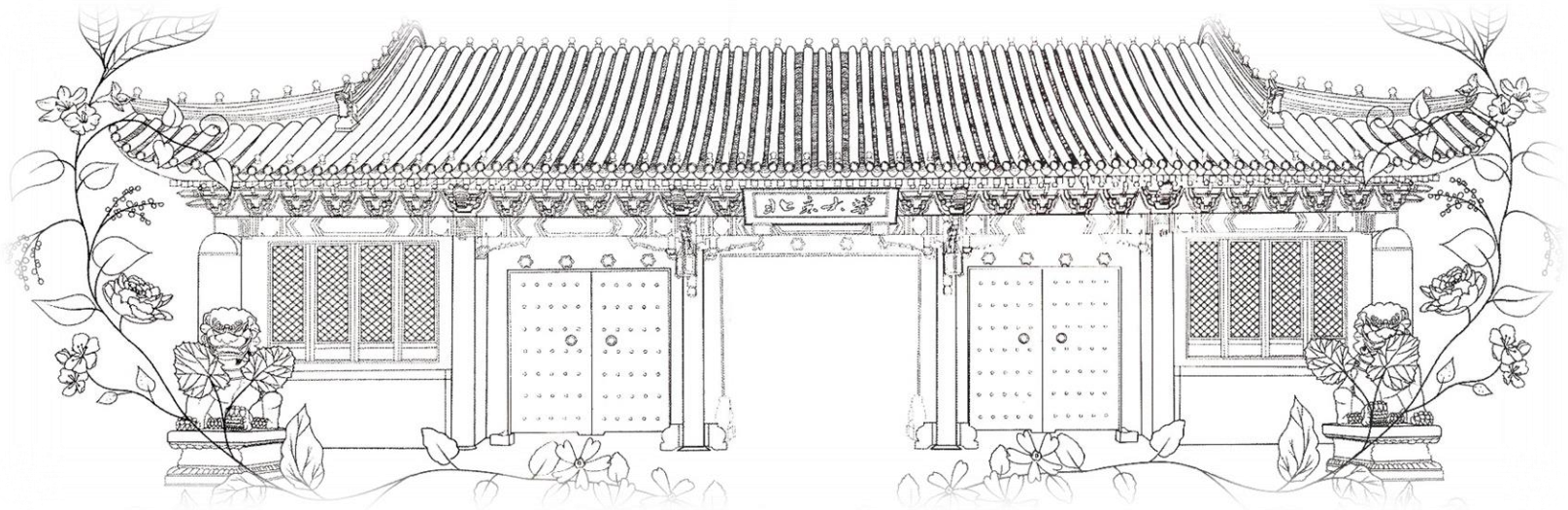
第2章 数据可视化 I

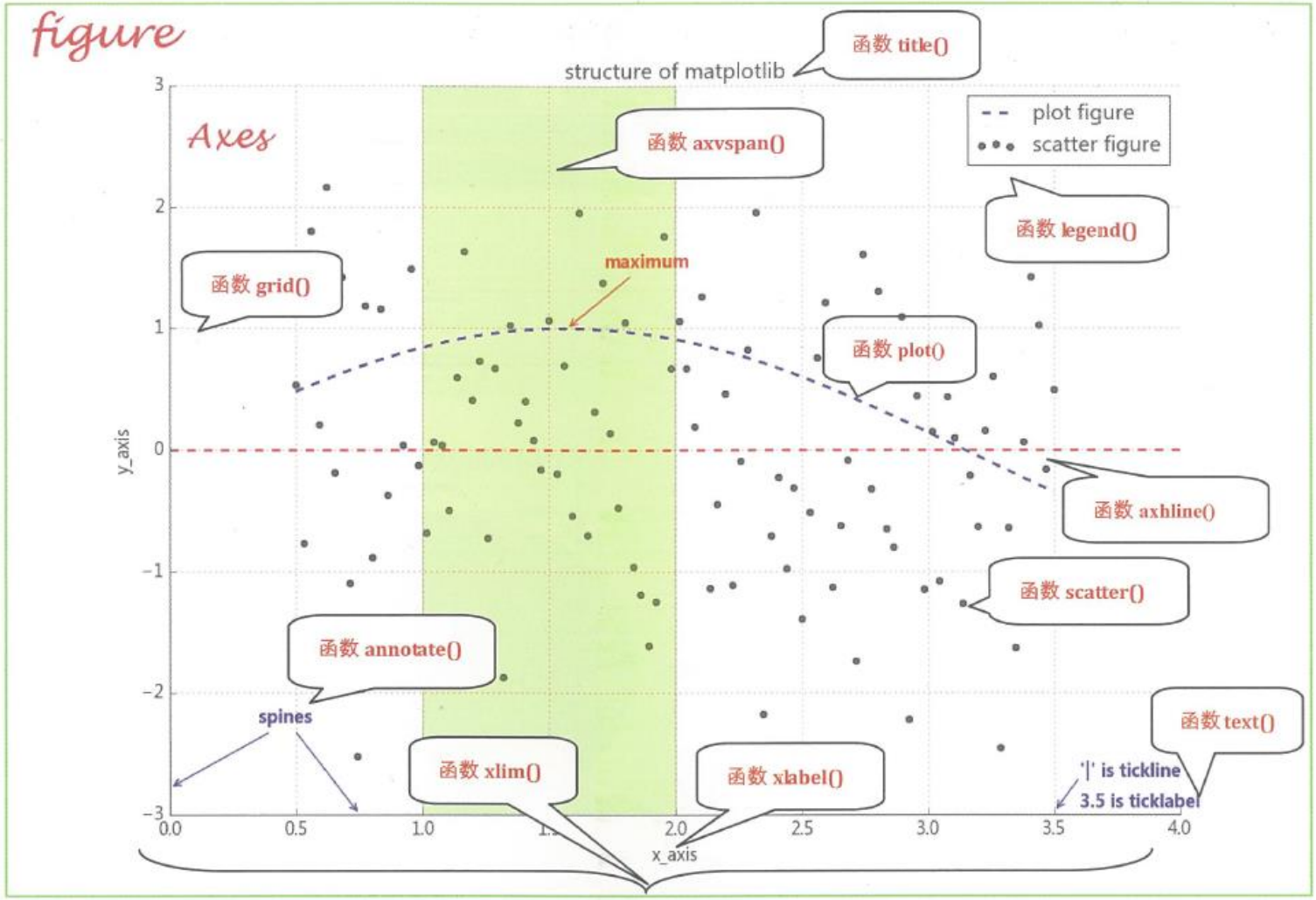


目录

- 使用函数绘制matplotlib的图表组成要素
- 使用统计函数绘制简单图形
- 完善统计图形
- 画布和绘图区域

使用函数绘制matplotlib的图表组成要素

Matplotlib中的常用函数



准备工作

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

plot()：展现变量的趋势变化

函数功能：展现变量的趋势变化。

调用签名：plt.plot(x,y,ls="-",lw=2,label="plot figure")

参数说明

- 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。

plot()：展现变量的趋势变化

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

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()
```

matplotlib.pyplot.plot(*args, scalex=True, scaley=True, data=None, **kwargs)

**kwargs

Property	Description
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
animated	bool
antialiased or aa	bool
clip_box	Bbox
clip_on	bool
clip_path	Patch or (Path, Transform) or None
color or c	color
contains	unknown
dash_capstyle	{'butt', 'round', 'projecting'}
dash_joinstyle	{'miter', 'round', 'bevel'}
dashes	sequence of floats (on/off ink in points) or (None, None)
data	(2, N) array or two 1D arrays

color

black	k	dimgray	dimgrey
gray	grey	darkgray	darkgrey
silver	lightgray	lightgrey	gainsboro
whitesmoke	w	white	snow
rosybrown	lightcoral	indianred	brown
firebrick	maroon	darkred	r
red	mistyrose	salmon	tomato
darksalmon	coral	orangered	lightsalmon
sienna	seashell	chocolate	saddlebrown
sandybrown	peachpuff	peru	linen
bisque	darkorange	burlywood	antiquewhite
tan	navajowhite	blanchedalmond	papayawhip
moccasin	orange	wheat	oldlace
floralwhite	darkgoldenrod	goldenrod	cornsilk
gold	lemonchiffon	khaki	palegoldenrod
darkkhaki	ivory	beige	lightyellow
lightgoldenrodyellow	olive	y	yellow
olivedrab	yellowgreen	darkolivegreen	greenyellow
chartreuse	lawngreen	honeydew	darkseagreen
palegreen	lightgreen	forestgreen	limegreen
darkgreen	g	green	lime
seagreen	mediumseagreen	springgreen	mintcream
mediumspringgreen	mediumaquamarine	aquamarine	turquoise
lightseagreen	mediumturquoise	azure	lightcyan
paleturquoise	darkslategray	darkslategrey	teal
darkcyan	c	aqua	cyan
darkturquoise	cadetblue	powderblue	lightblue
deepskyblue	skyblue	lightskyblue	steelblue
aliceblue	dodgerblue	lightslategray	lightslategrey
slategray	slategrey	lightsteelblue	cornflowerblue
royalblue	ghostwhite	lavender	midnightblue
navy	darkblue	mediumblue	b
blue	slateblue	darkslateblue	mediumslateblue
mediumpurple	rebeccapurple	blueviolet	indigo
darkorchid	darkviolet	mediumorchid	thistle
plum	violet	purple	darkmagenta
m	fuchsia	magenta	orchid
mediumvioletred	deeppink	hotpink	lavenderblush
palevioletred	crimson	pink	lightpink

**kwargs

drawstyle or ds	{'default', 'steps', 'steps-pre', 'steps-mid', 'steps-post'}, default: 'default'
figure	Figure
fillstyle	{'full', 'left', 'right', 'bottom', 'top', 'none'}
gid	str
in_layout	bool
label	object
linestyle or ls	{'-', '--', '-.', ':', '', (offset, on-off-seq), ...}
linewidth or lw	float
marker	marker style string, Path or MarkerStyle
markeredgecolor or mec	color
markeredgewidth or mew	float
markerfacecolor or mfc	color
markerfacecoloralt or mfcalt	color
markersize or ms	float
markevery	None or int or (int, int) or slice or List[int] or float or (float, float) or List[bool]
path_effects	AbstractPathEffect
picker	unknown
pickradius	float

linestyle

- '-' solid line style
- '--' dashed line style
- '-.' dash-dot line style
- ':' dotted line style

marker

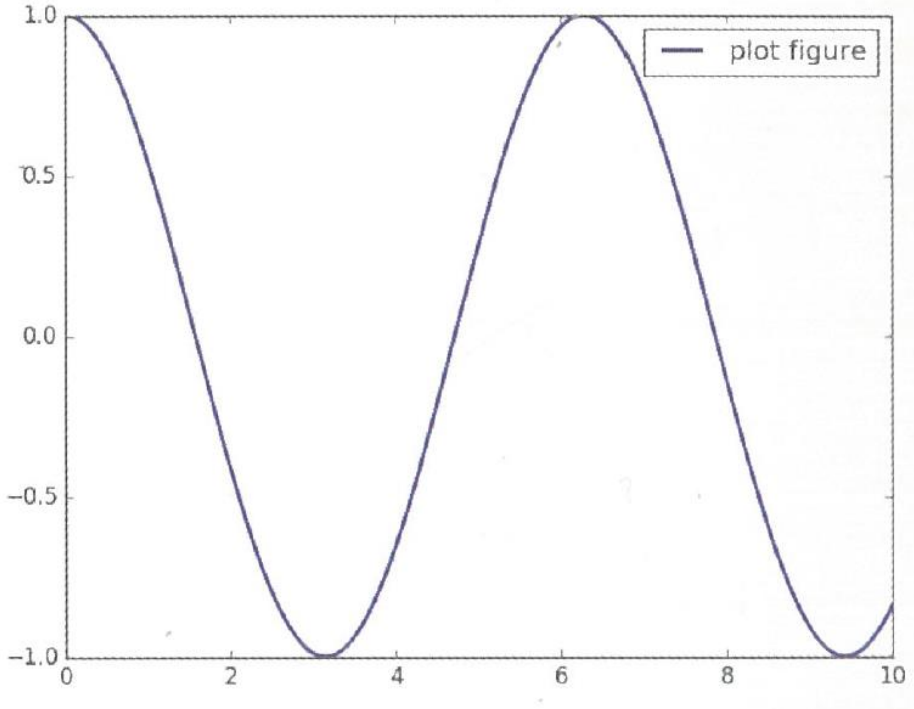
```
'.'      point marker
','      pixel marker
'o'      circle marker
'v'      triangle_down marker
'^'      triangle_up marker
'<'      triangle_left marker
'>'      triangle_right marker
'1'      tri_down marker
'2'      tri_up marker
'3'      tri_left marker
'4'      tri_right marker
's'      square marker
'p'      pentagon marker
'*'      star marker
'h'      hexagon1 marker
'H'      hexagon2 marker
'+'      plus marker
'x'      x marker
'D'      diamond marker
'd'      thin_diamond marker
'|'      vline marker
'_'      hline marker
```

**kwargs

`rasterized`	bool or None
`sketch_params`	(scale: float, length: float, randomness: float)
`snap`	bool or None
`solid_capstyle`	{'butt', 'round', 'projecting'}
`solid_joinstyle`	{'miter', 'round', 'bevel'}
`transform`	`matplotlib.transforms.Transform`
`url`	str
`visible`	bool
`xdata`	1D array
`ydata`	1D array
`zorder`	float

plot()：展现变量的趋势变化

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

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()
```



scatter()：寻找变量之间的关系

函数功能：寻找变量之间的关系。

调用签名：plt.scatter(x,y1,c="b",label="scatter figure")

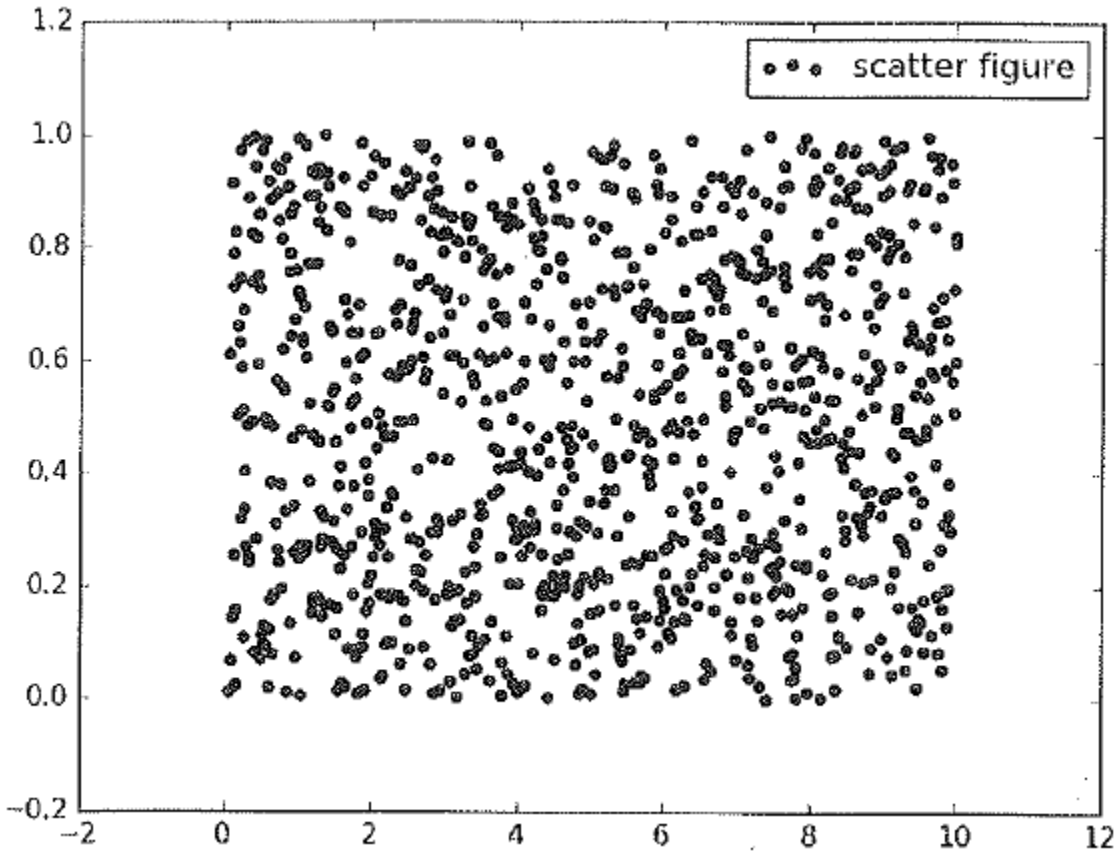
参数说明

- 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。

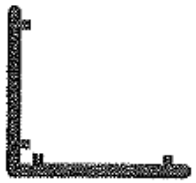
scatter()：寻找变量之间的关系

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

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

plt.scatter(x,y,label="scatter figure")

plt.legend()

plt.show()
```

scatter()：寻找变量之间的关系



xlim()：设置x轴的数值显示范围



函数功能：设置 x 轴的数值显示范围。

调用签名：plt.xlim(xmin,xmax)

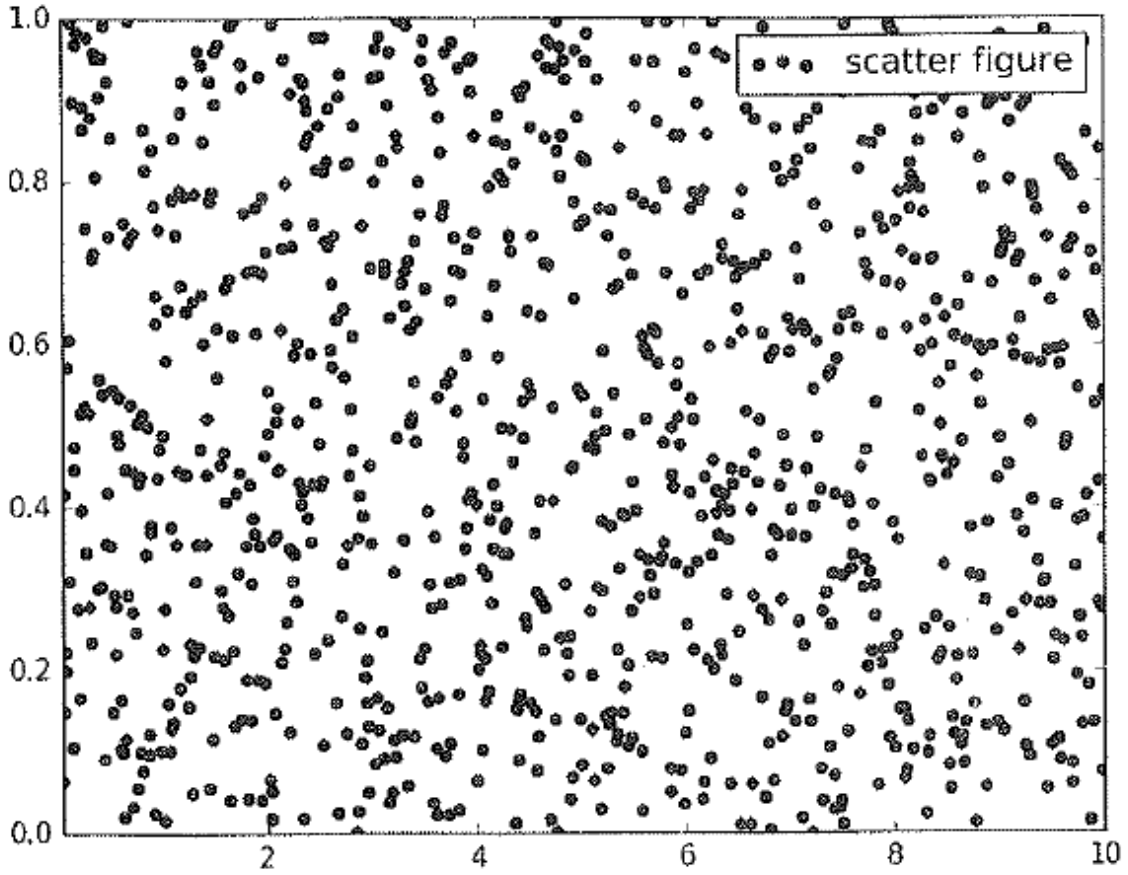
参数说明

- xmin：x 轴上的最小值。
- xmax：x 轴上的最大值。
- 平移性：上面的函数功能，调用签名和参数说明同样可以平移到函数 ylim()上。

xlim()：设置x轴的数值显示范围

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

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

plt.scatter(x,y,label="scatter figure")

plt.legend()

plt.xlim(0.05,10)
plt.ylim(0,1)

plt.show()
```

xlim()：设置x轴的数值显示范围



xlabel()：设置x轴的标签文本

函数功能：设置 *x* 轴的标签文本。

调用签名：plt.xlabel(string)

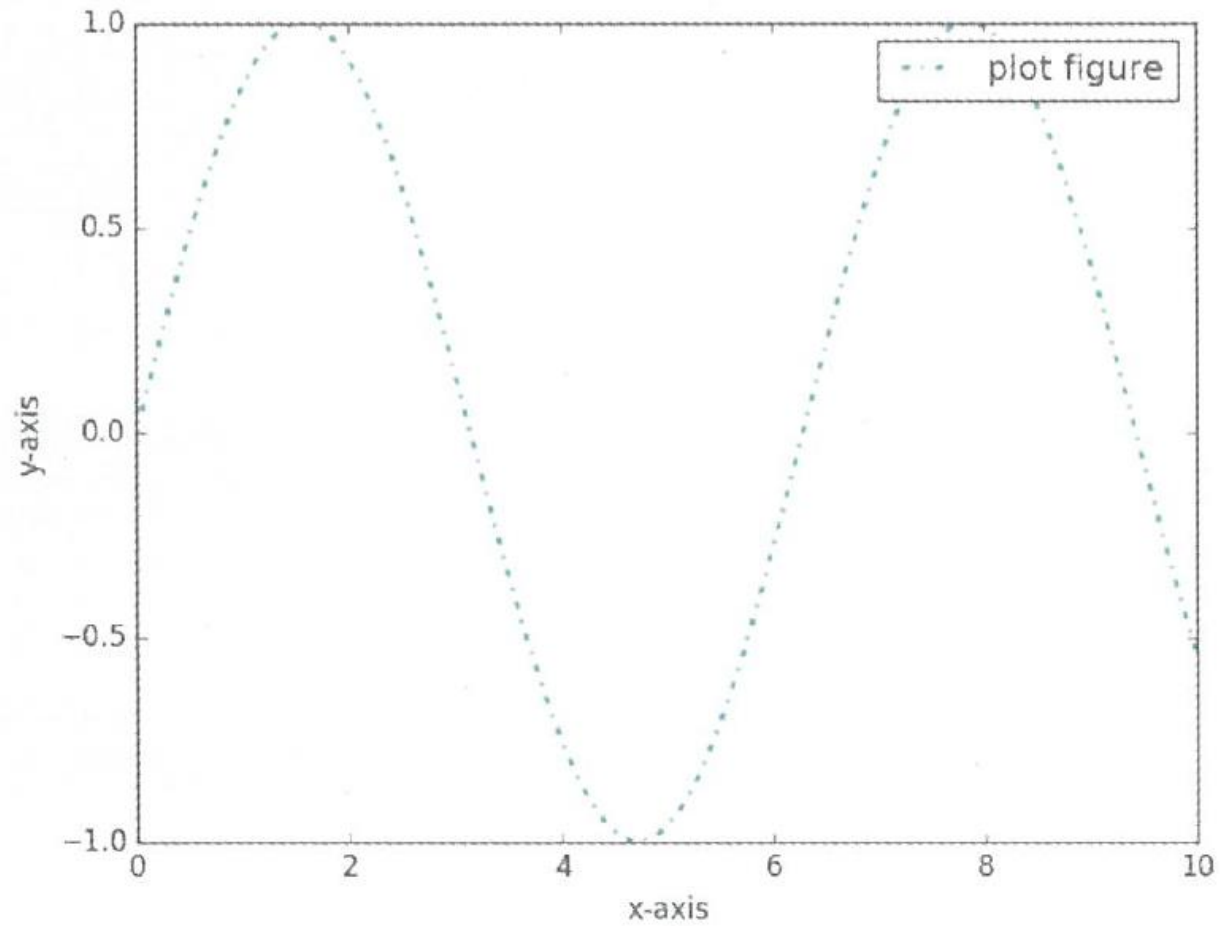
参数说明

- string：标签文本内容。
- 平移性：上面的函数功能，调用签名和参数说明同样可以平移到函数 ylabel()上。

xlabel()：设置x轴的标签文本

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

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

plt.plot(x,y,ls="-.",lw=2,c="c",label="plot figure")

plt.legend()

plt.xlabel("x-axis")
plt.ylabel("y-axis")

plt.show()
```

xlabel()：设置x轴的标签文本



grid()：绘制刻度线的网格线

函数功能：绘制刻度线的网格线。

调用签名：plt.grid(linestyle=":",color="r")

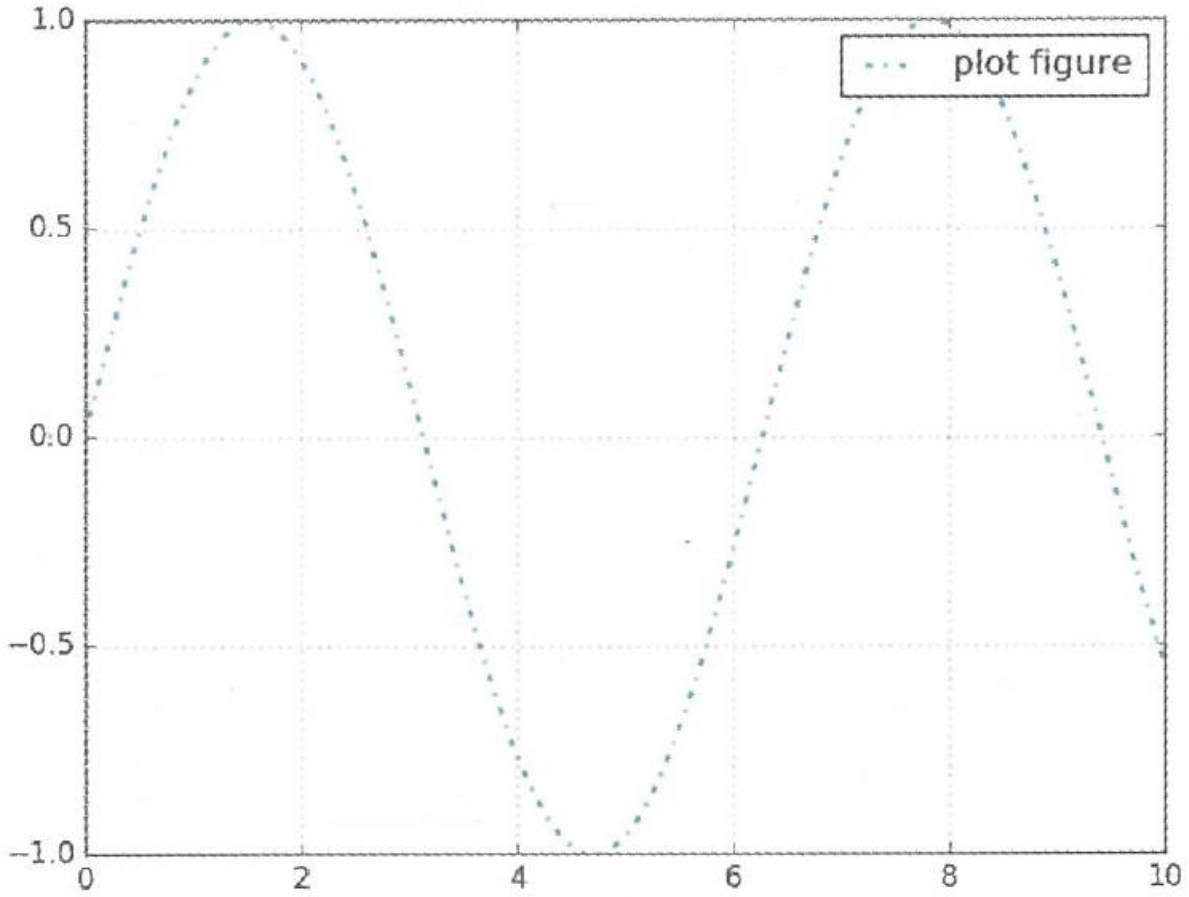
参数说明

- linestyle：网格线的线条风格。
- color：网格线的线条颜色。

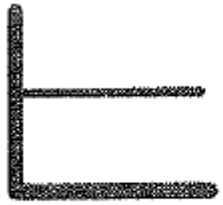
grid()：绘制刻度线的网格线

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

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

plt.plot(x,y,ls="-.",lw=2,c="c",label="plot figure")

plt.legend()

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

plt.show()
```

grid()：绘制刻度线的网格线



axhline()：平行于x轴的水平参考线



函数功能：绘制平行于 x 轴的水平参考线。

调用签名：plt.axhline(y=0.0,c="r",ls="--",lw=2)

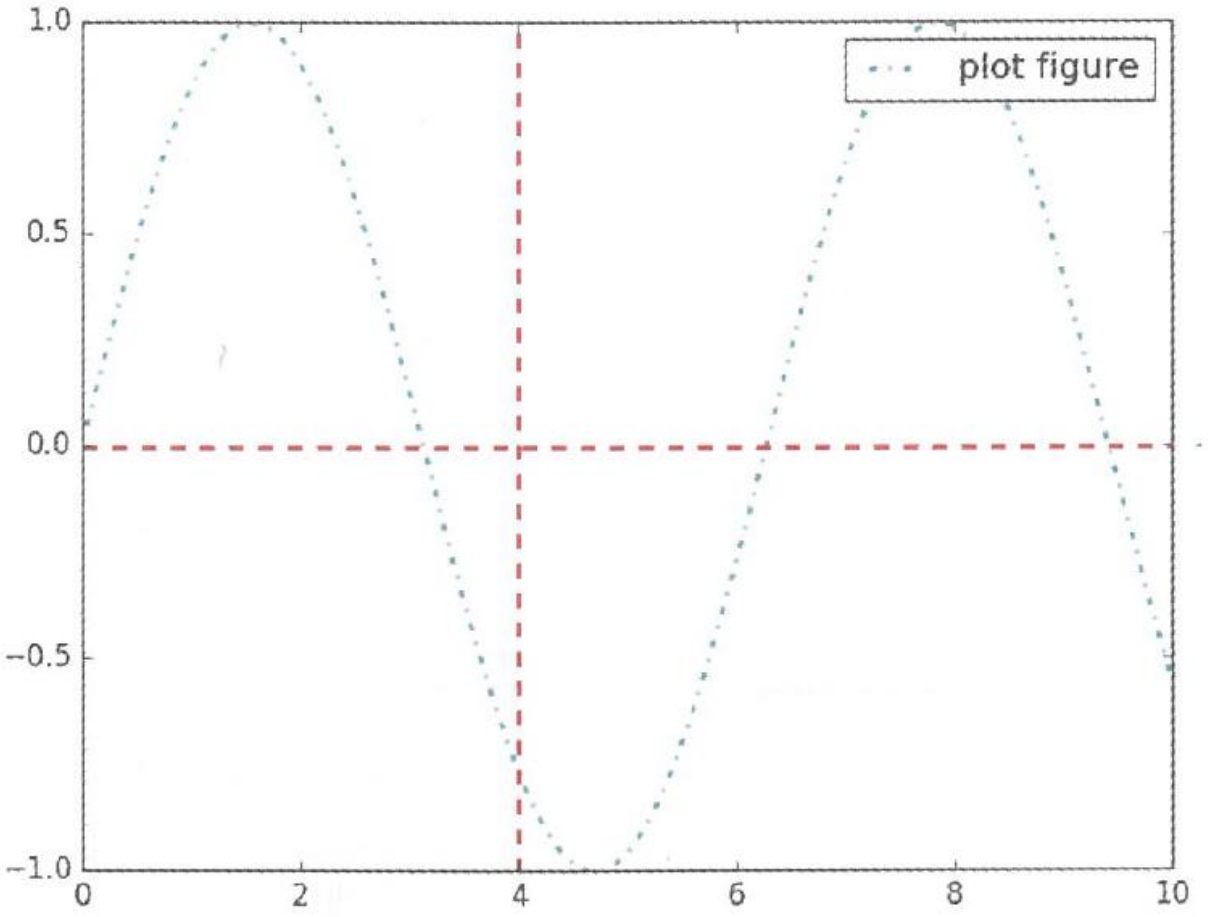
参数说明

- 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：上面的函数功能，调用签名和参数说明同样可以平移到函数 axvline()上。

axhline()：平行于x轴的水平参考线

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

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

plt.plot(x,y,ls="-.",lw=2,c="c",label="plot figure")

plt.legend()

plt.axhline(y=0.0,c="r",ls="--",lw=2)
plt.axvline(x=4.0,c="r",ls="--",lw=2)

plt.show()
```

axhline()：平行于x轴的水平参考线



axvspan()：垂直于x轴的参考区域

函数功能：绘制垂直于 x 轴的参考区域。

调用签名：plt.axvspan(xmin=1.0,xmax=2.0,facecolor="y",alpha=0.3)。

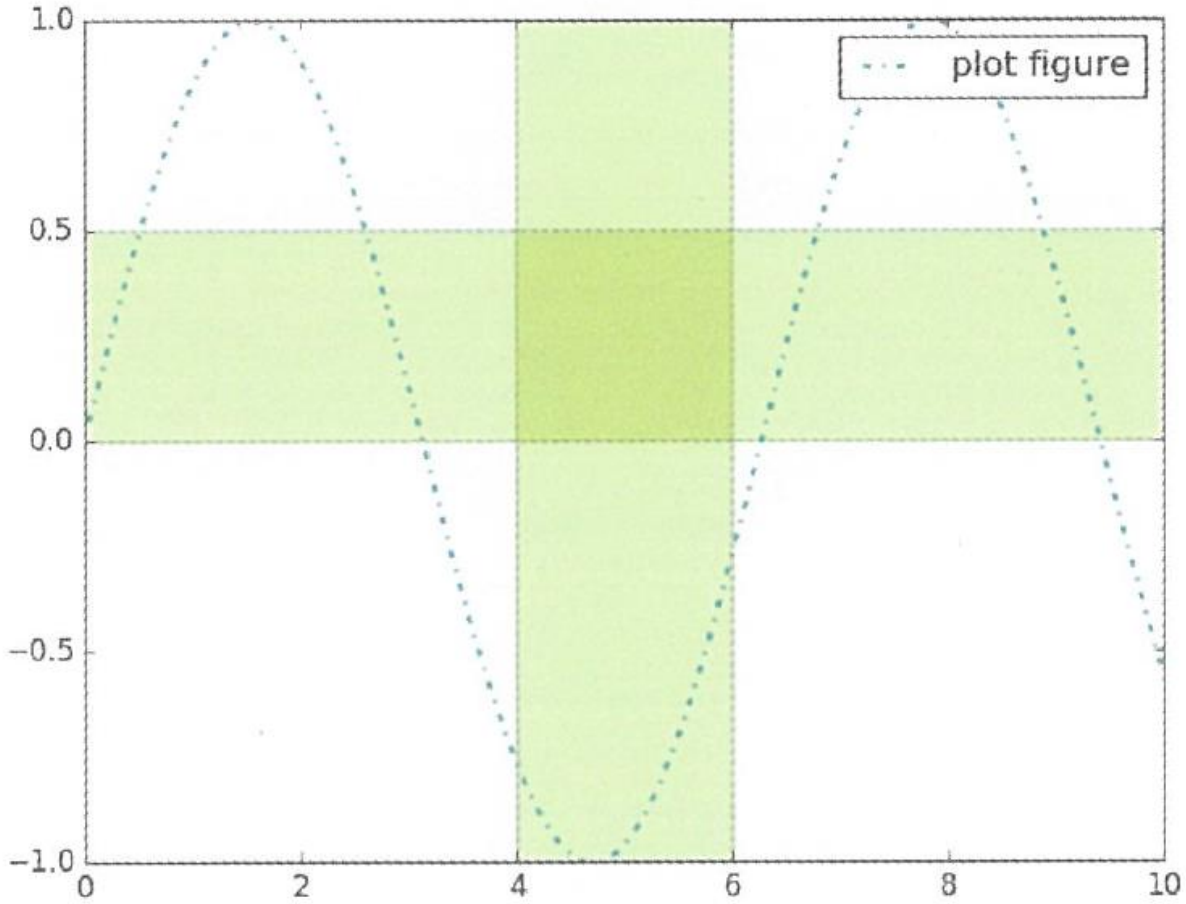
参数说明

- 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：上面的函数功能、调用签名和参数说明可以平移到函数 axhspan()上。

axvspan()：垂直于x轴的参考区域

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

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

plt.plot(x,y,ls="-.",lw=2,c="c",label="plot figure")

plt.legend()

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.show()
```

axvspan()：垂直于x轴的参考区域



annotate()：添加图形内容细节的指向型注释文本

函数功能：添加图形内容细节的指向型注释文本。

调用签名：plt.annotate(string,xy=(np.pi/2,1.0),xytext=((np.pi/2)+0.15,1.5),weight="bold", color="b", arrowprops=dict(arrowstyle="->",connectionstyle="arc3",color="b"))。

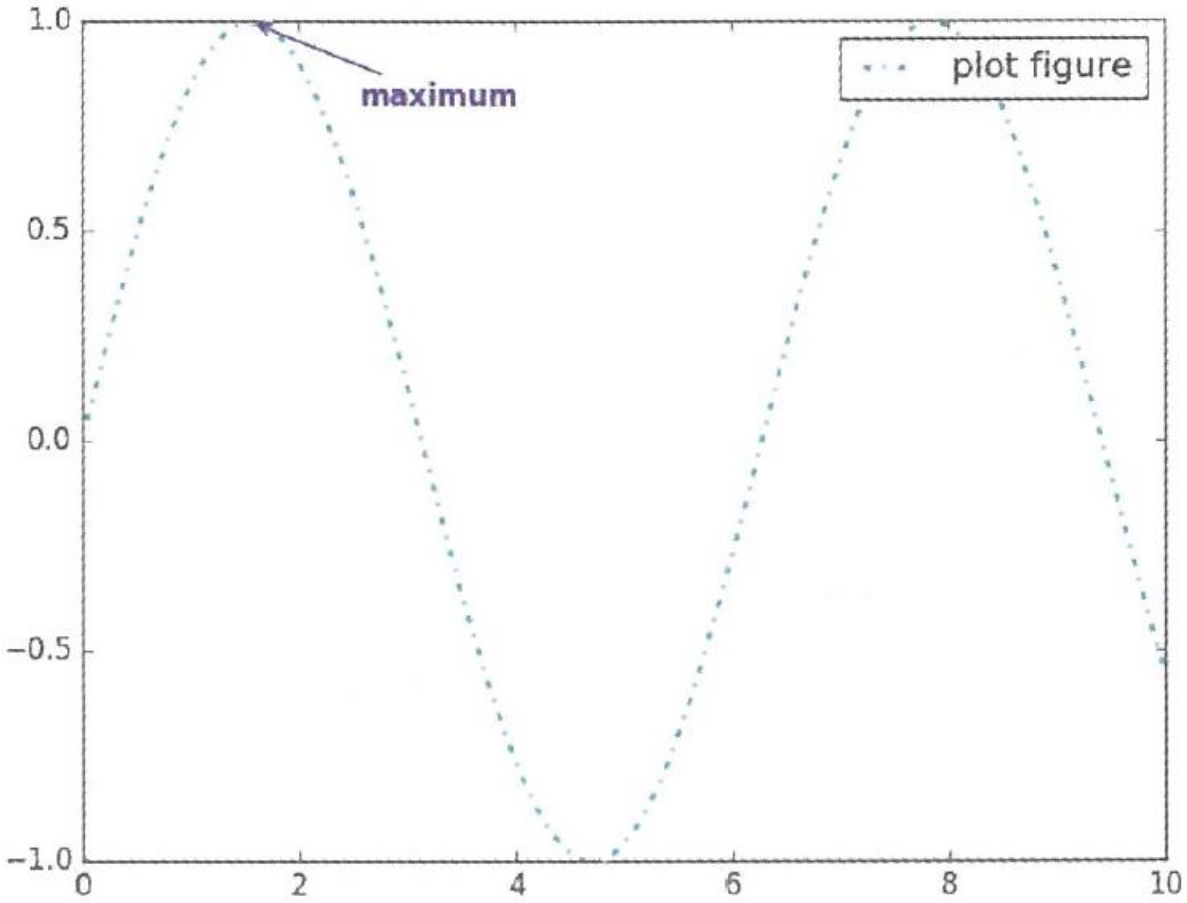
参数说明

- 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。

annotate()：添加图形内容细节的指向型注释文本

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

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

plt.plot(x,y,ls="-.",lw=2,c="c",label="plot figure")

plt.legend()

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="b"))

plt.show()
```

annotate()：添加图形内容细节的指向型注释文本



text()：添加图形内容细节的无指向型注释文本

函数功能：添加图形内容细节的无指向型注释文本。

调用签名：plt.text(x,y,string,weight="bold",color="b")。

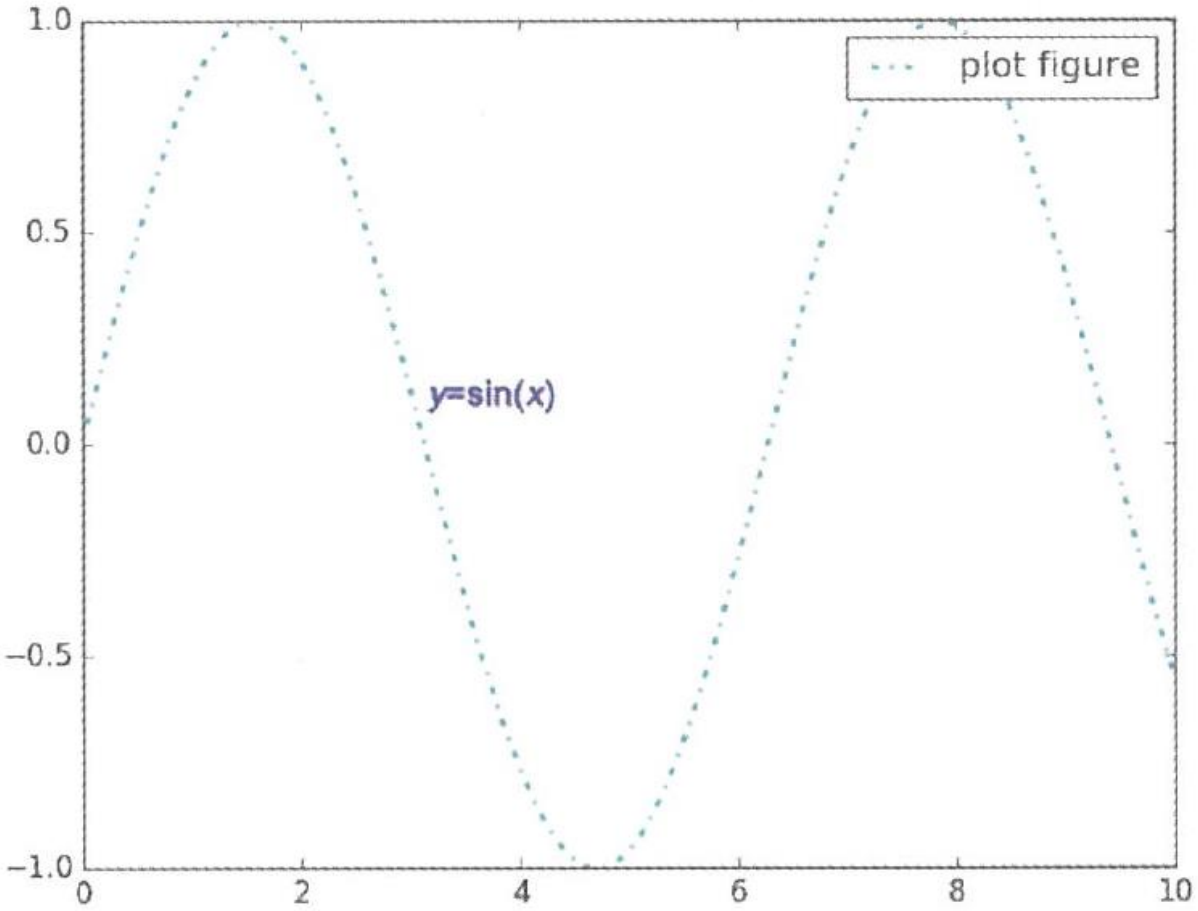
参数说明

- 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。

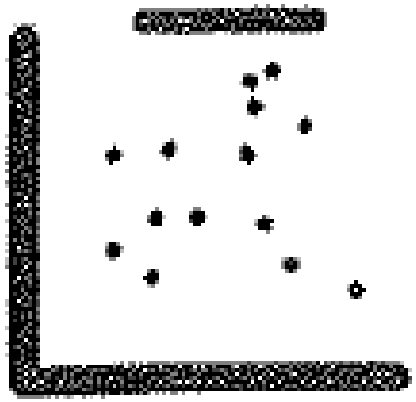
text()：添加图形内容细节的无指向型注释文本

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

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

plt.plot(x,y,ls="-.",lw=2,c="c",label="plot figure")

plt.legend()

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

plt.show()
```

text()：添加图形内容细节的无指向型注释文本



title()：添加图形内容的标题



函数功能：添加图形内容的标题。

调用签名：plt.title(string)。

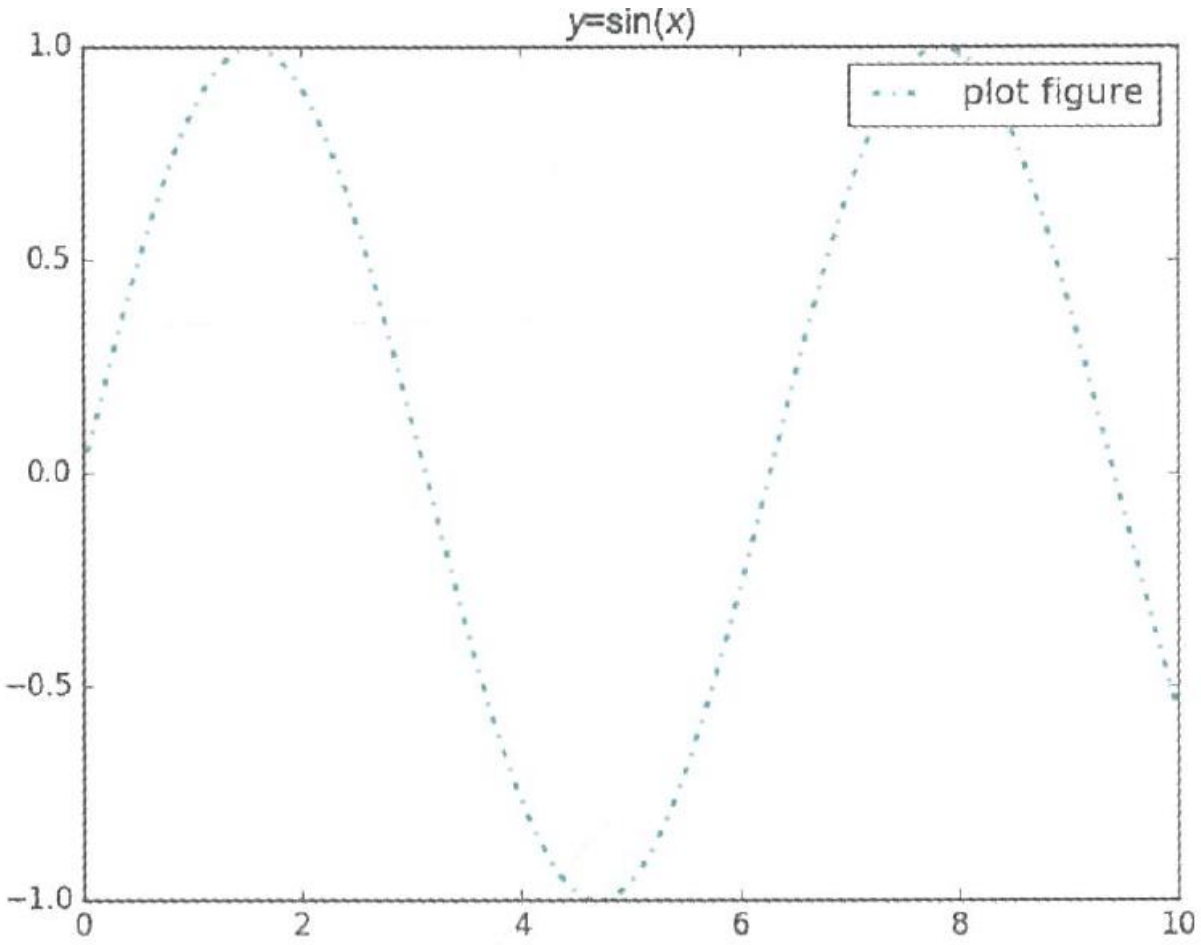
参数说明

- string：图形内容的标题文本。

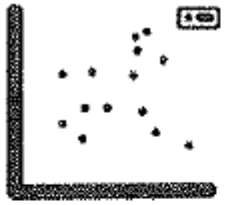
title()：添加图形内容的标题

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

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

plt.plot(x,y,ls="-.",lw=2,c="c",label="plot figure")

plt.legend()

plt.title("y=sin(x)")

plt.show()
```

title()：添加图形内容的标题



legend()：标示不同图形的文本标签图例



函数功能：标示不同图形的文本标签图例。

调用签名：plt.legend(loc="lower left")。

参数说明

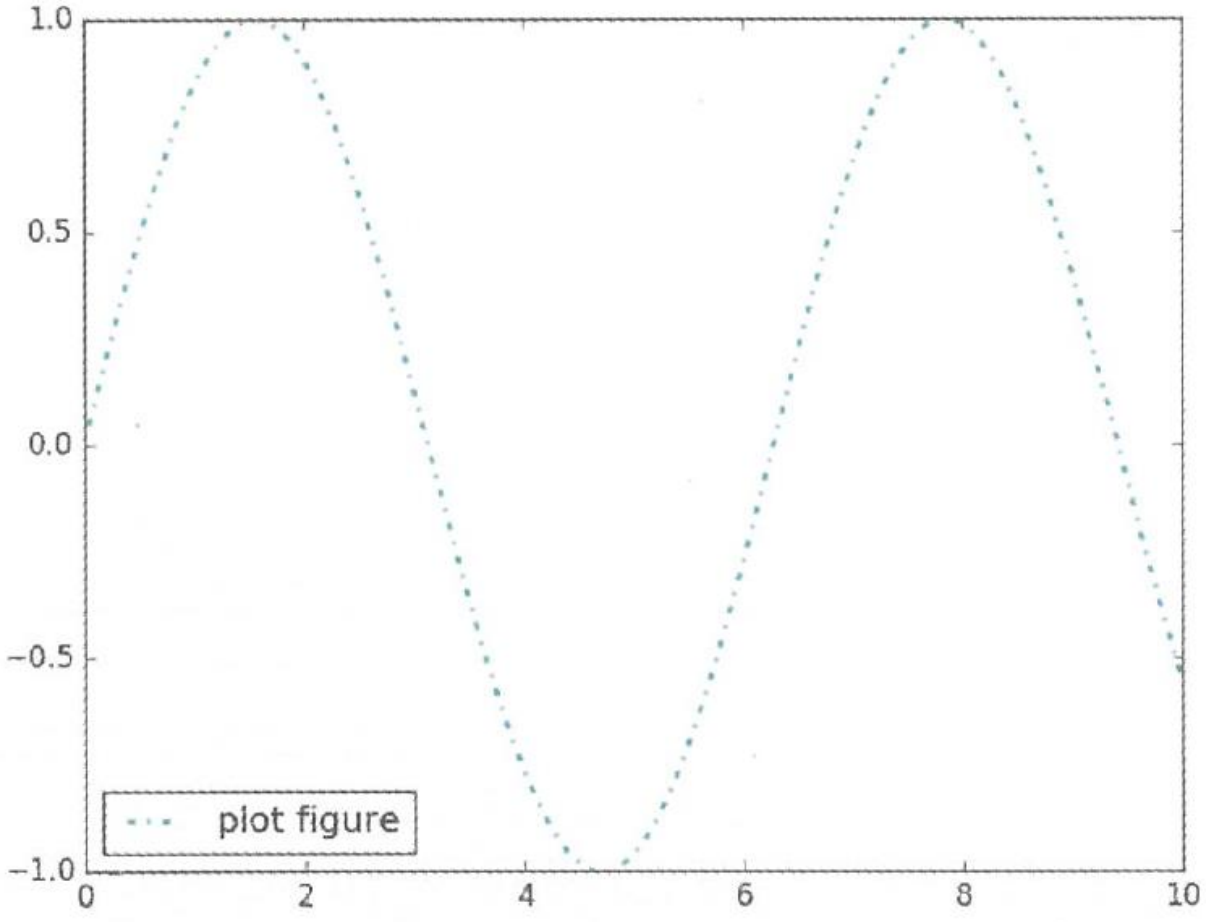
- loc：图例在图中的地理位置。

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

legend()：标示不同图形的文本标签图例

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

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

plt.plot(x,y,ls="-.",lw=2,c="c",label="plot figure")

plt.legend(loc="lower left")

plt.show()
```

legend()：标示不同图形的文本标签图例



legend()：标示不同图形的文本标签图例

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

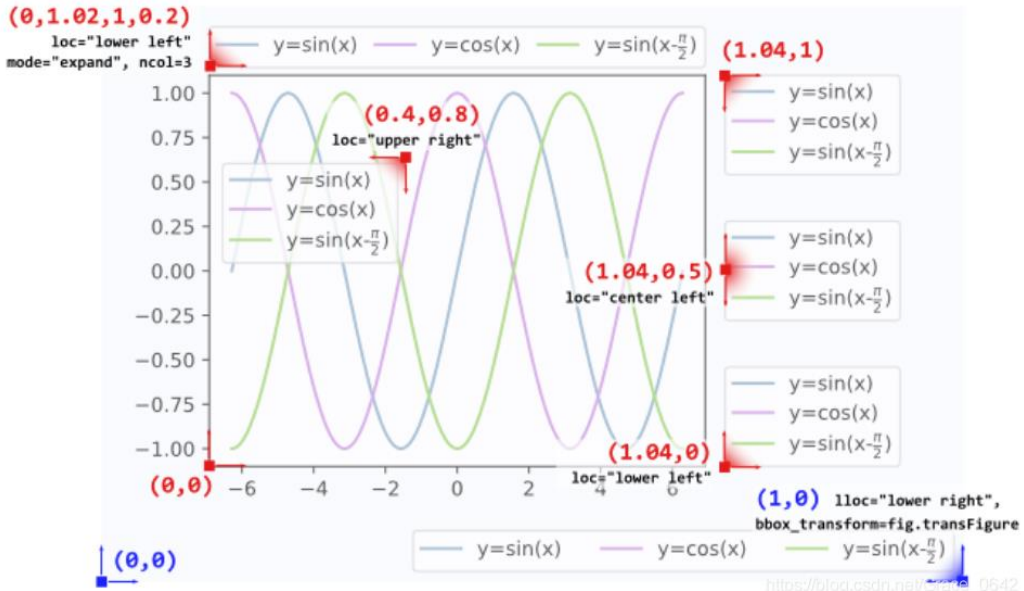
关键字参数 bbox_to_anchor 的参数值是一个四元元组，且使用 Axes 坐标系统。也就是说，第 1 个元素代表距离画布左侧的 x 轴长度的倍数的距离；第 2 个元素代表距离画布底部的 y 轴长度的倍数的距离；第 3 个元素代表 x 轴长度的倍数的线框长度；第 4 个元素代表 y 轴长度的倍数的线框宽度。代码中的语句“legend(loc="upper left",bbox_to_anchor=(0.05,0.95))”会将图例放在上方左手边拐角处的距离坐标轴左边 0.1、底部 7.6 的位置。关键字参数 shadow 是控制线框是否添加阴影的选择。关键字参数 fancybox 是控制线框的圆角或直角的选择。

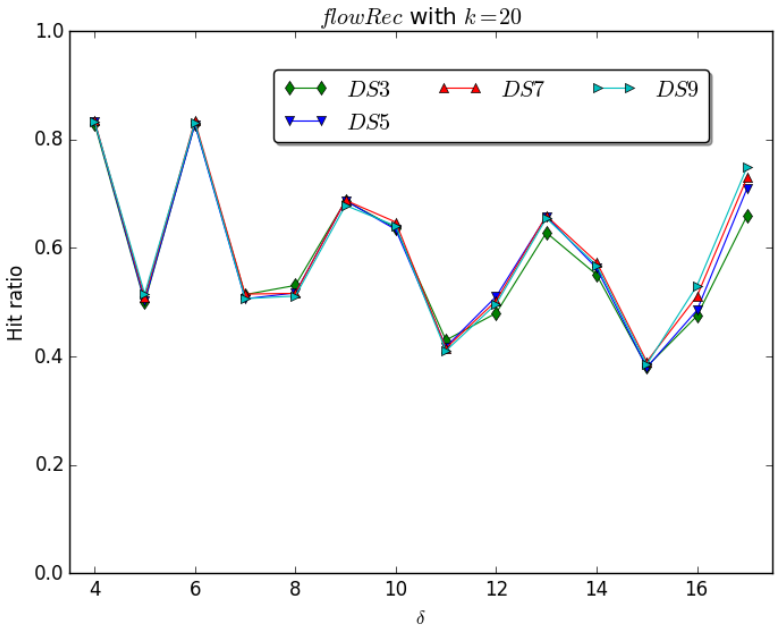
legend()：标示不同图形的文本标签图例

以方框的上中心为标定点

方框的上中心横坐标在整体的60%处，纵坐标在整体的95%处

- 直接看例子：
  - plt.legend(loc='upper center', bbox_to_anchor=(0.6,0.95), ncol=3, fancybox=True, shadow=True)





实例：请课下自己阅读代码并实践

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

from matplotlib import cm as cm

# define data
x = np.linspace(0.5,3.5,100)
y = np.sin(x)
y1 = np.random.randn(100)

# scatter figure
plt.scatter(x,y1,c="0.25",label="scatter figure")

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

实例（续）

```
# some clean up(removing chartjunk)
# turn the top spine and the right spine off
for spine in plt.gca().spines.keys():
    if spine == "top" or spine == "right":
        plt.gca().spines[spine].set_color("none")

# turn bottom tick for x-axis on
plt.gca().xaxis.set_ticks_position("bottom")
# set tick_line position of bottom

# turn left ticks for y-axis on
plt.gca().yaxis.set_ticks_position("left")
# set tick line position of left
```

实例（续）

```
# set x,yaxis limit
plt.xlim(0.0,4.0)
plt.ylim(-3.0,3.0)

# set axes labels
plt.ylabel("y_axis")
plt.xlabel("x_axis")

# set x,yaxis grid
plt.grid(True,ls=":",color="r")

# add a horizontal line across the axis
plt.axhline(y=0.0,c="r",ls="--",lw=2)

# add a vertical span across the axis
plt.axvspan(xmin=1.0,xmax=2.0,facecolor="y",alpha=.3)
```

实例（续）

```
    #set annotating info
    plt.annotate("maximum",xy=(np.pi/2,1.0),
                 xytext=((np.pi/2)+0.15,1.5),weight="bold",color="r",

arrowprops=dict(arrowstyle="->",connectionstyle="arc3",color="r"))

    plt.annotate("spines",xy=(0.75,-3),
                 xytext=(0.35,-2.25),weight="bold",color="b",

arrowprops=dict(arrowstyle="->",connectionstyle="arc3",color="b"))

    plt.annotate("",xy=(0,-2.78),
                 xytext=(0.4,-2.32),

arrowprops=dict(arrowstyle="->",connectionstyle="arc3",color="b"))
```

实例（续）

```
    plt.annotate("",xy=(3.5,-2.98),
                 xytext=(3.6,-2.70),

arrowprops=dict(arrowstyle="->",connectionstyle="arc3",color="b"))

    # set text info
    plt.text(3.6,-2.70,"'|' is tickline",weight="bold",color="b")
    plt.text(3.6,-2.95,"3.5 is ticklabel",weight="bold",color="b")

    # set title
    plt.title("structure of matplotlib")

    # set legend
    plt.legend()

    plt.show()
```

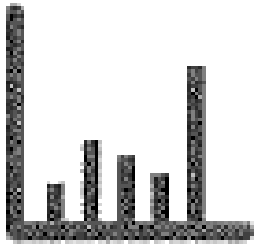
图形的保存

```
plt.savefig("D:\\FIGURE_DEMO.png")
```

```
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)
```

使用统计函数绘制简单图形

bar()：柱状图



- x：柱状图中的柱体标签值。
- y：柱状图中的柱体高度。
- align：柱体对齐方式。
- color：柱体颜色。
- tick_label：刻度标签值。
- alpha：柱体的透明度。

函数功能：在 x 轴上绘制定性数据的分布特征。

调用签名：plt.bar(x,y)。

参数说明

- x：标示在 x 轴上的定性数据的类别。
- y：每种定性数据的类别的数量。

柱状图主要是应用在定性数据的可视化场景中，或者是离散型数据的分布展示。例如，一个本科班级的学生的籍贯分布，出国旅游人士的职业分布以及下载一款 App 产品的操作系统的分布。

bar()：柱状图

align : {'center', 'edge'}, default: 'center'

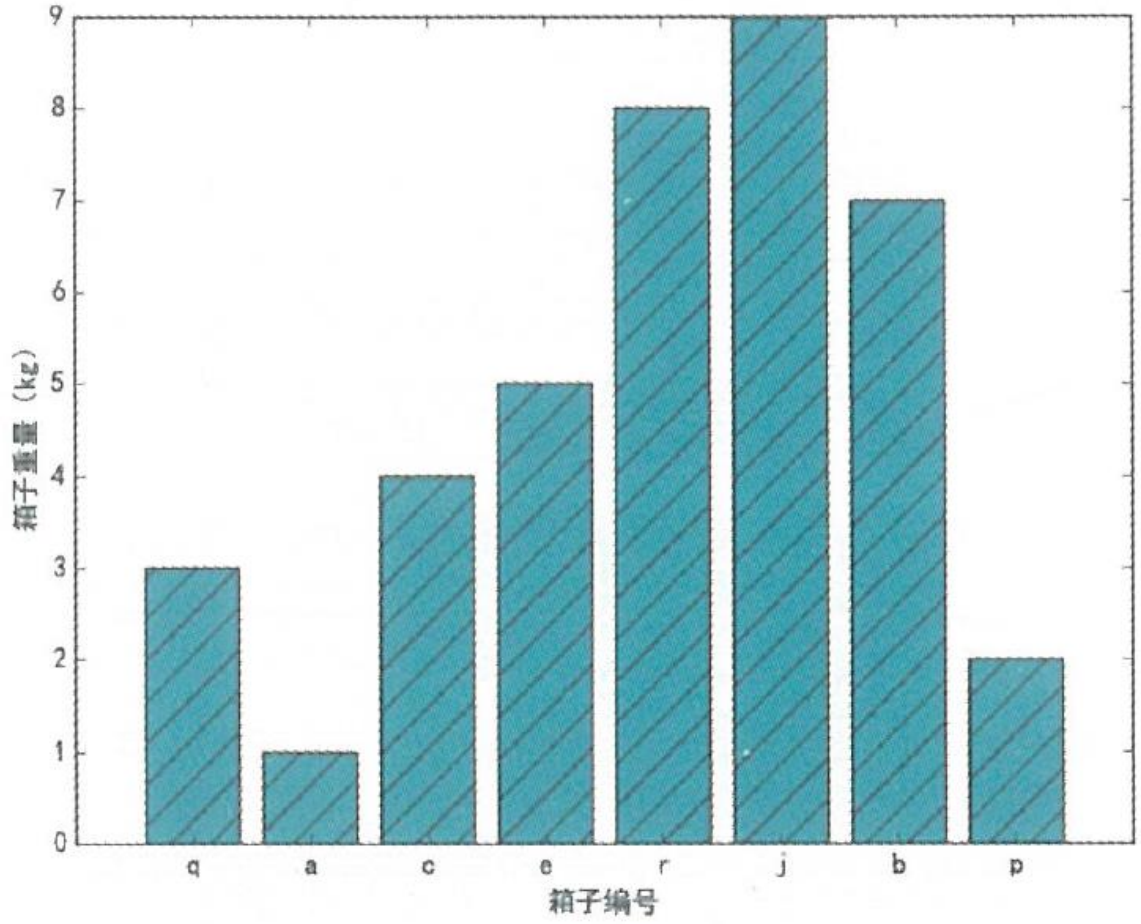
Alignment of the bars to the *x* coordinates:

- 'center': Center the base on the *x* positions.
- 'edge': Align the left edges of the bars with the *x* positions.

To align the bars on the right edge pass a negative *width* and `align='edge'`.

```
# -*- coding:utf-8 -*-

import matplotlib as mpl
import matplotlib.pyplot as plt

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

# some simple data
x = [1,2,3,4,5,6,7,8]
y = [3,1,4,5,8,9,7,2]

# create bar
plt.bar(x,y,align="center",color="c",tick_label=["q","a","c","e","r","j"
,"b","p"],hatch="/")

# set x,y_axis label
plt.xlabel("箱子编号")
plt.ylabel("箱子重量（kg）")

plt.show()
```

bar()：柱状图

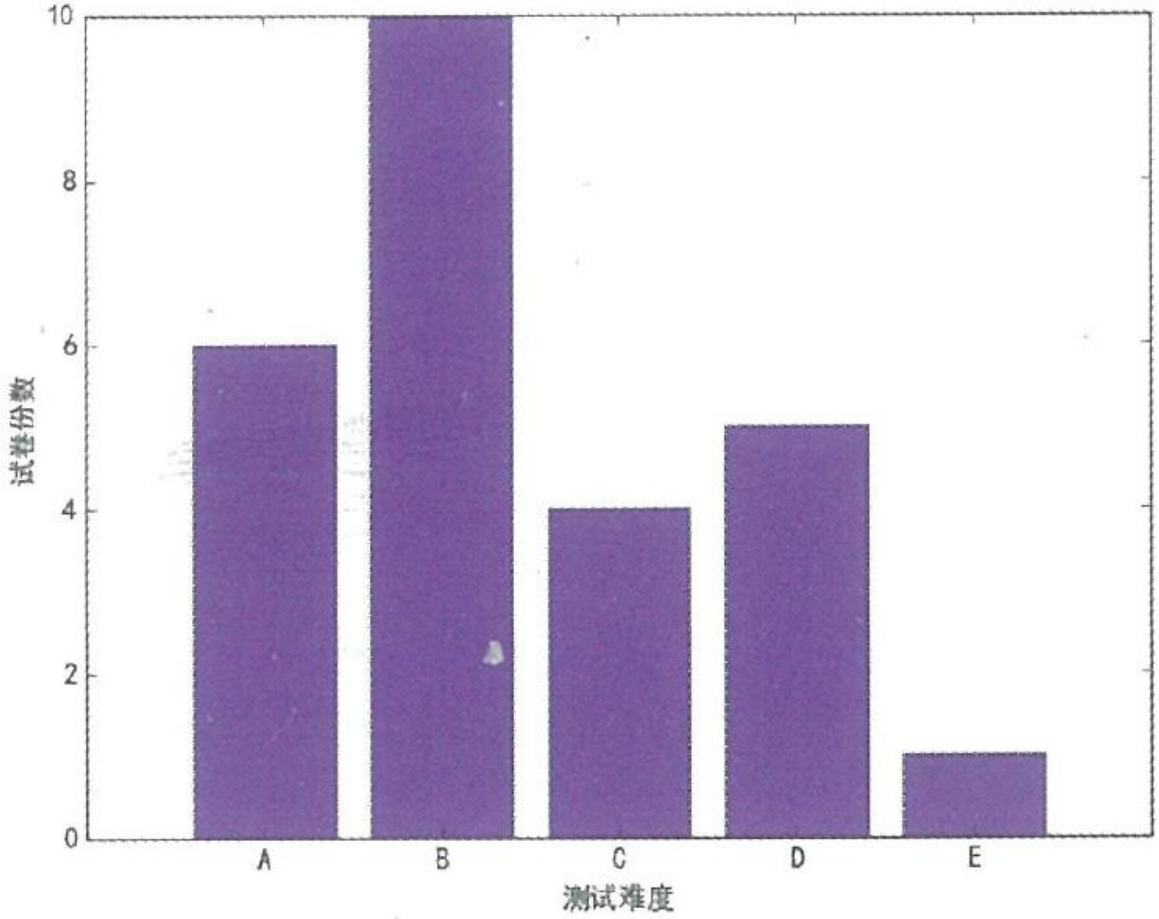


bar()：柱状图

```
# -*- coding:utf-8 -*-

import matplotlib as mpl
import matplotlib.pyplot as plt

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

# some simple data
x = [1,2,3,4,5]
y = [6,10,4,5,1]

# create bar
plt.bar(x,y,align="center",color="b",tick_label=["A","B","C","D","E"],al
ha=0.6)

# set x,y_axis label
plt.xlabel("测试难度")
plt.ylabel("试卷份数")

# set yaxis grid
plt.grid(True,axis="y",ls=":",color="r",alpha=0.3)

plt.show()
```

bar()：柱状图

啊其实这里是有平行于x轴的网格线的！只不过看不清 ☹

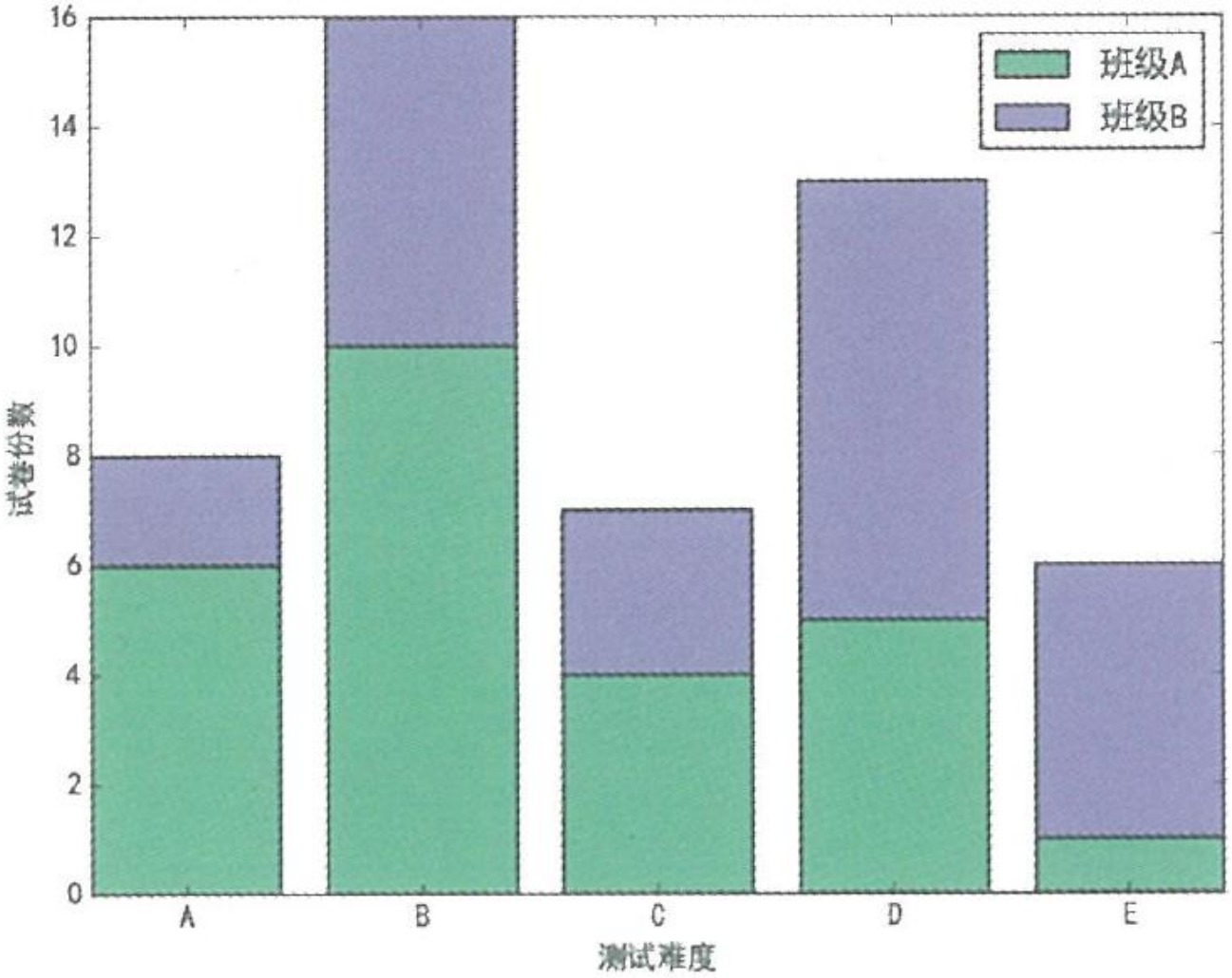


bar()：堆积柱状图

```
import matplotlib as mpl
import matplotlib.pyplot as plt

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

# some simple data
x = [1,2,3,4,5]
y = [6,10,4,5,1]
y1 = [2,6,3,8,5]

# create bar
plt.bar(x,y,align="center",color="#66c2a5",tick_label=["A","B","C","D","E"],label="班级 A")
plt.bar(x,y1,align="center",bottom=y,color="#8da0cb",label="班级 B")

# set x,y_axis label
plt.xlabel("测试难度")
plt.ylabel("试卷份数")

plt.legend()

plt.show()
```

Y1的柱子基于Y的柱子上，bottom默认值为0

bar()：堆积柱状图



【补充】颜色的不同表示形式

- 颜色的名字和十六进制的表示是对应的，比如：

```
cnames = {
    'aliceblue':            '#F0F8FF',
    'antiquewhite':         '#FAEBD7',
    'aqua':                 '#00FFFF',
    'aquamarine':           '#7FFFD4',
    'azure':                '#F0FFFF',
    'beige':                '#F5F5DC',
    'bisque':               '#FFE4C4',
    'black':                '#000000',
    'blanchedalmond':       '#FFEBCD',
    'blue':                 '#0000FF',
```

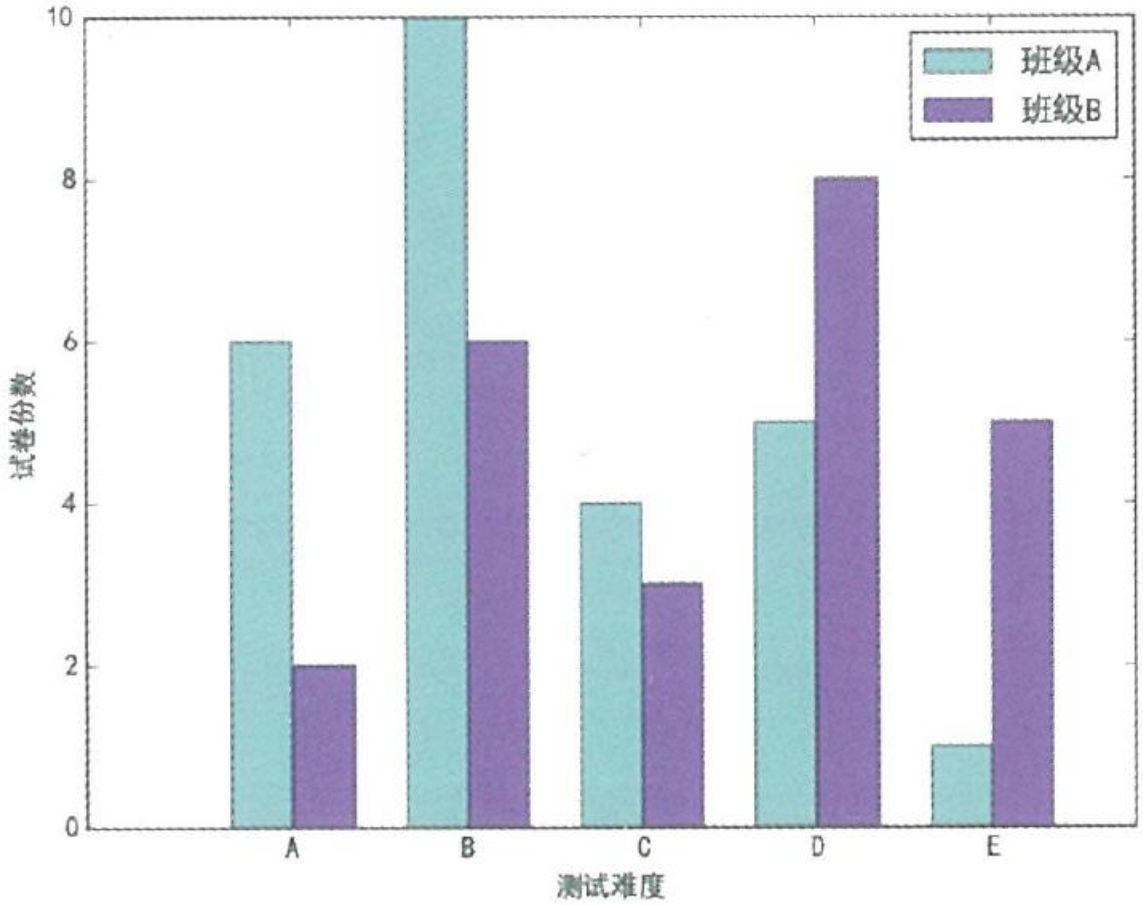
bar()：多数据并列柱状图

```
# -*- coding:utf-8 -*-

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

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

# some simple data
x = np.arange(5)
y = [6,10,4,5,1]
y1 = [2,6,3,8,5]

bar_width = 0.35
tick_label=["A","B","C","D","E"]
```

```
    # create bar
    plt.bar(x,y,bar_width,color="c",align="center",label="班级A",alpha=0.5)
    plt.bar(x+bar_width,y1,bar_width,color="b",align="center",label="  班  级
B",alpha=0.5)

    # set x,y_axis label
    plt.xlabel("测试难度")
    plt.ylabel("试卷份数")

    # set xaxis ticks and ticklabels
    plt.xticks(x+bar_width/2,tick_label)

    plt.legend()

    plt.show()
```

这行代码表达什么意思？

bar()：多数据并列柱状图

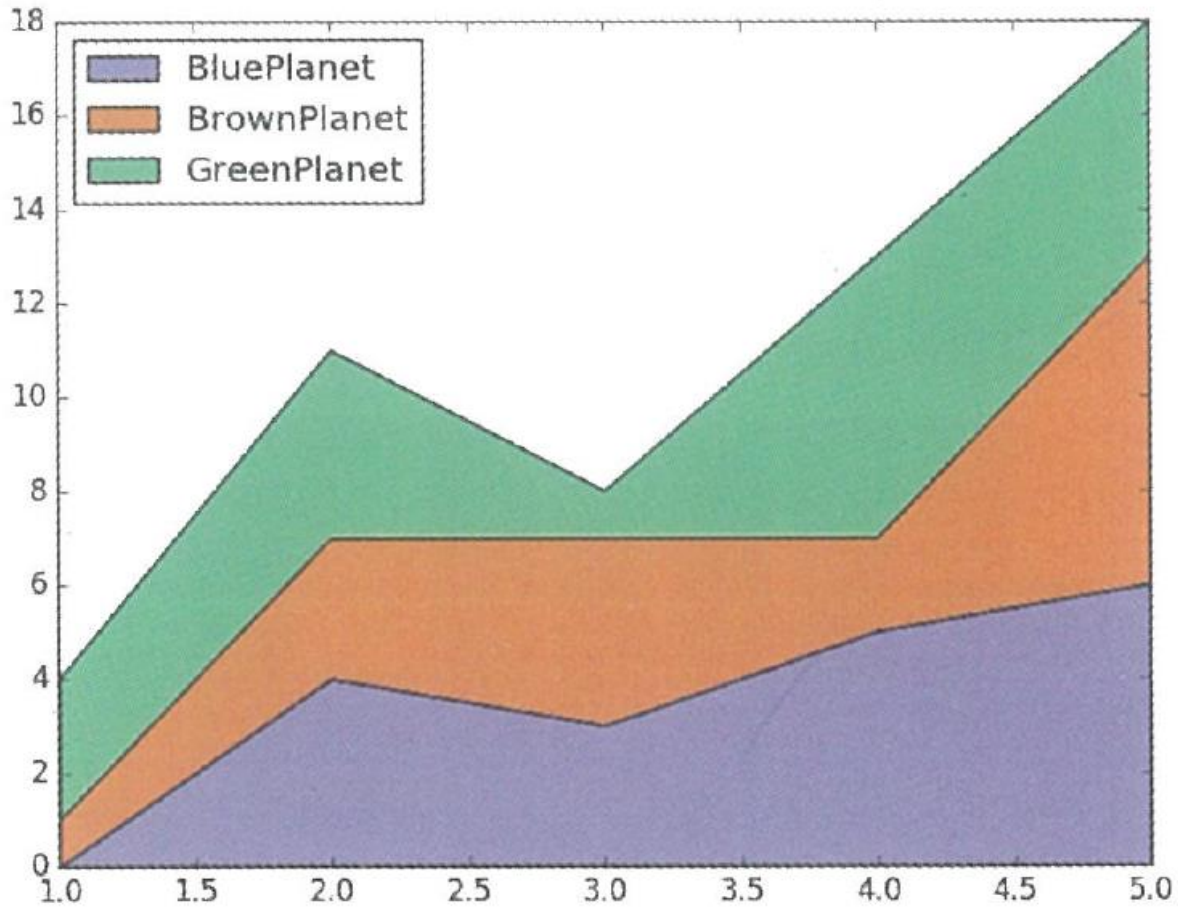


stackplot()：堆积折线图

堆积折线图是通过绘制不同数据集的折线图而生成的。堆积折线图是按照垂直方向上彼此堆叠且又不相互覆盖的排列顺序，绘制若干条折线图而形成的组合图形。

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

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()
```

stackplot()：堆积折线图



stackplot()：堆积折线图

x : (N,) array-like

y : (M, N) array-like

The data is assumed to be unstacked. Each of the following calls is legal:

```
stackplot(x, y)           # where y has shape (M, N)
stackplot(x, y1, y2, y3)  # where y1, y2, y3, y4 have length N
```

barh()：条形图

函数功能：在 y 轴上绘制定性数据的分布特征。

调用签名：plt.barh(x,y)。

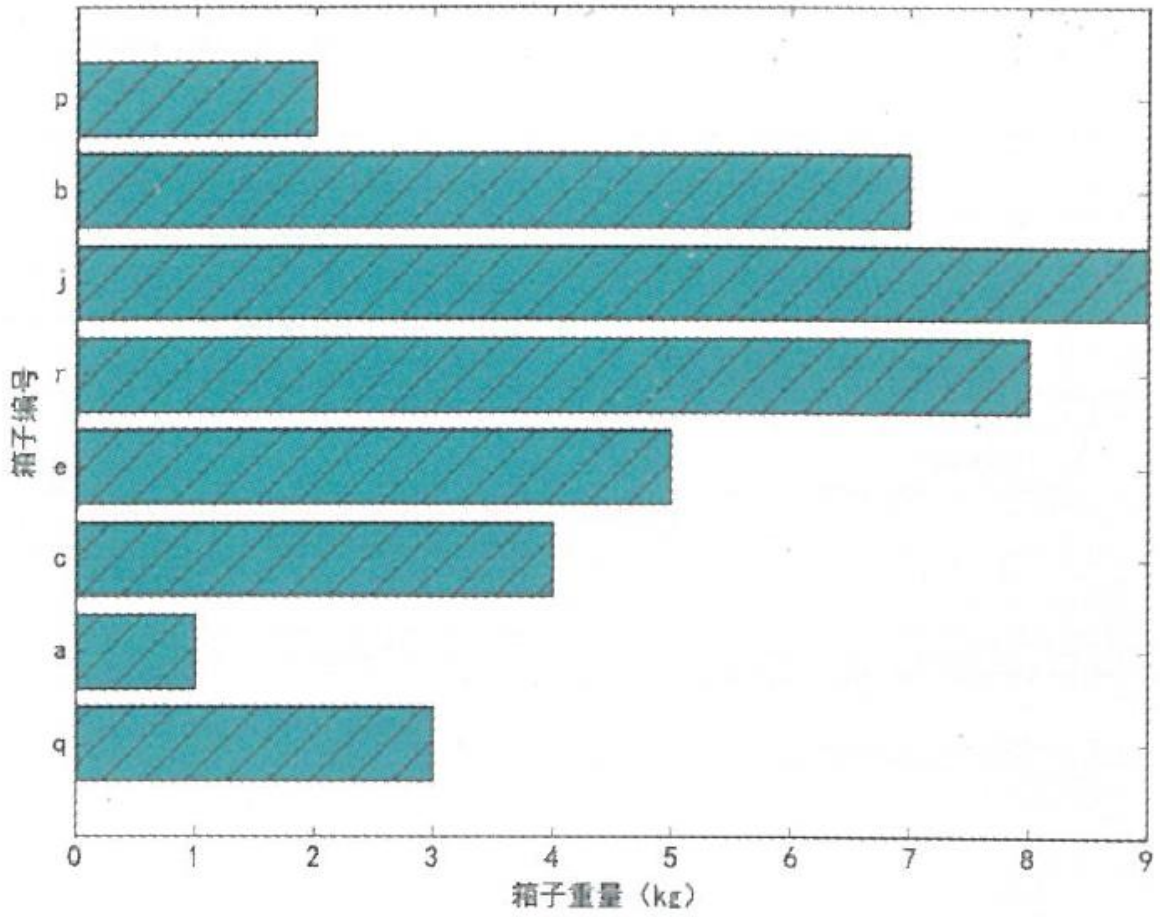
参数说明

- 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。

barh()：条形图

```
# -*- coding:utf-8 -*-

import matplotlib as mpl
import matplotlib.pyplot as plt

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

# some simple data
x = [1,2,3,4,5,6,7,8]
y = [3,1,4,5,8,9,7,2]

# create bar
plt.barh(x,y,align="center",color="c",tick_label=["q","a","c","e","r","j
","b","p"],hatch="/")

# set x,y_axis label
plt.xlabel("箱子重量（kg）")
plt.ylabel("箱子编号")

plt.show()
```

barh()：条形图



step()：阶梯图

阶梯图在可视化效果上正如图形的名字那样形象，就如同山间的台阶时而上升时而下降，从图形本身而言，很像折线图。也用采是反映数据的趋势变化或是周期规律的。阶梯图经常使用在时间序列数据的可视化任务中，凸显时序数据的波动周期和规律。

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

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()
```

where ： 设置阶梯所在位置，取值范围为{'pre', 'post', 'mid'}，默认值为'pre'

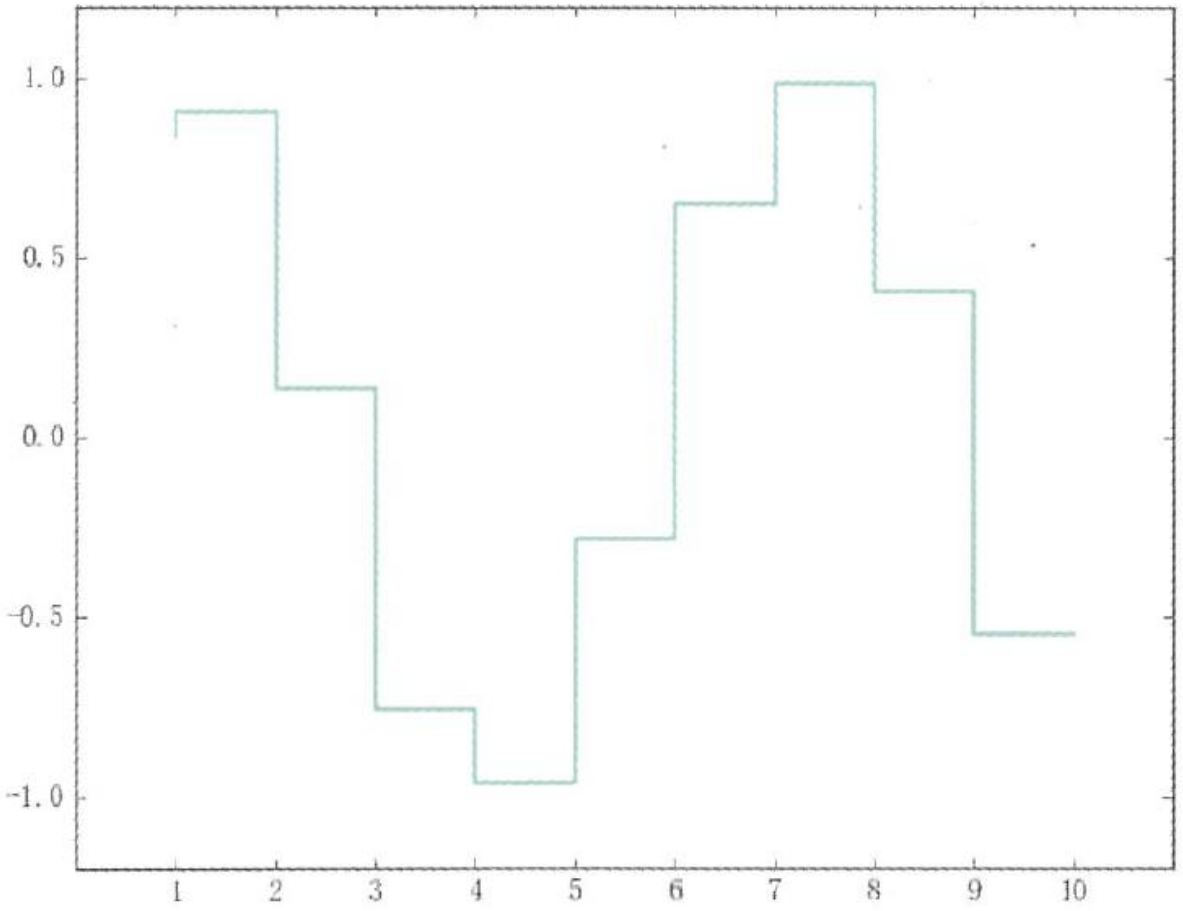
step()：阶梯图

where : {'pre', 'post', 'mid'}, default: 'pre'

Define where the steps should be placed:

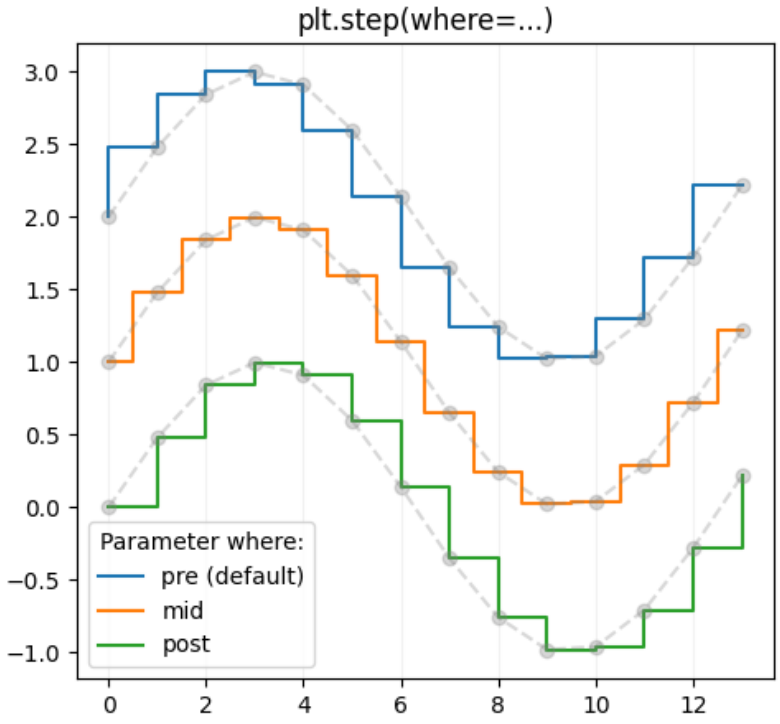
- 'pre': The y value is continued constantly to the left from every *x* position, i.e. the interval `(x[i-1], x[i]]` has the value `y[i]`.
- 'post': The y value is continued constantly to the right from every *x* position, i.e. the interval `[x[i], x[i+1])` has the value `y[i]`.
- 'mid': Steps occur half-way between the *x* positions.

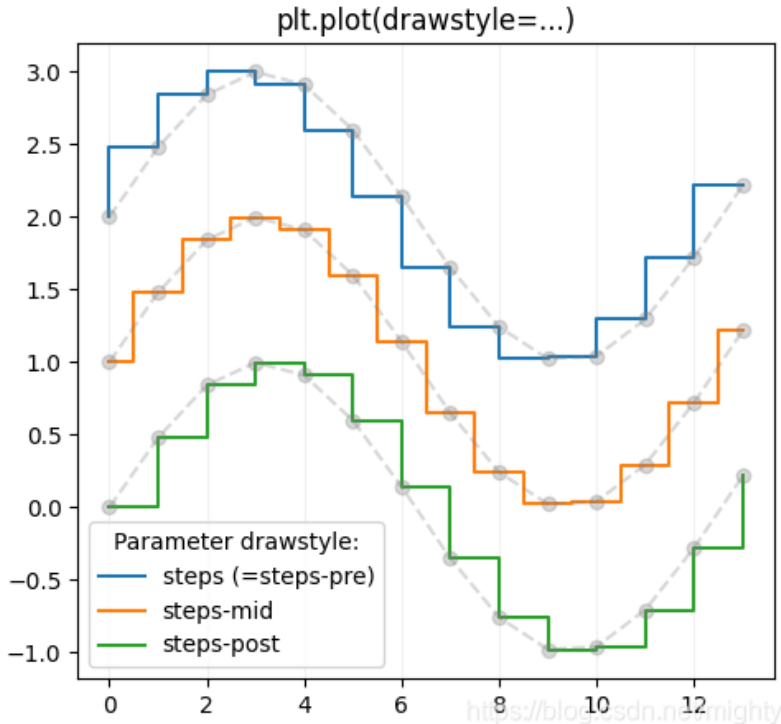
step()：阶梯图



step()与plot()

plt.plot()中的drawstyle默认为default





https://matplotlib.org/stable/gallery/lines_bars_and_markers/step_demo.html#sphx-glr-gallery-lines-bars-and-markers-step-demo-py

hist()：直方图

函数功能：在 x 轴上绘制定量数据的分布特征。

调用签名：plt.hist(x)。

参数说明

- x：在 x 轴上绘制箱体的定量数据输入值。

hist(x,bins=bins,color= "b",histtype="bar",label="score",rwidth=10)

- x：连续型数据输入值。
- bins：用于确定柱体的个数或是柱体边缘范围。
- color：柱体的颜色。
- histtype：柱体类型。
- label：图例内容。
- rwidth：柱体的相对宽度，取值范围是[0.0,1.0]。

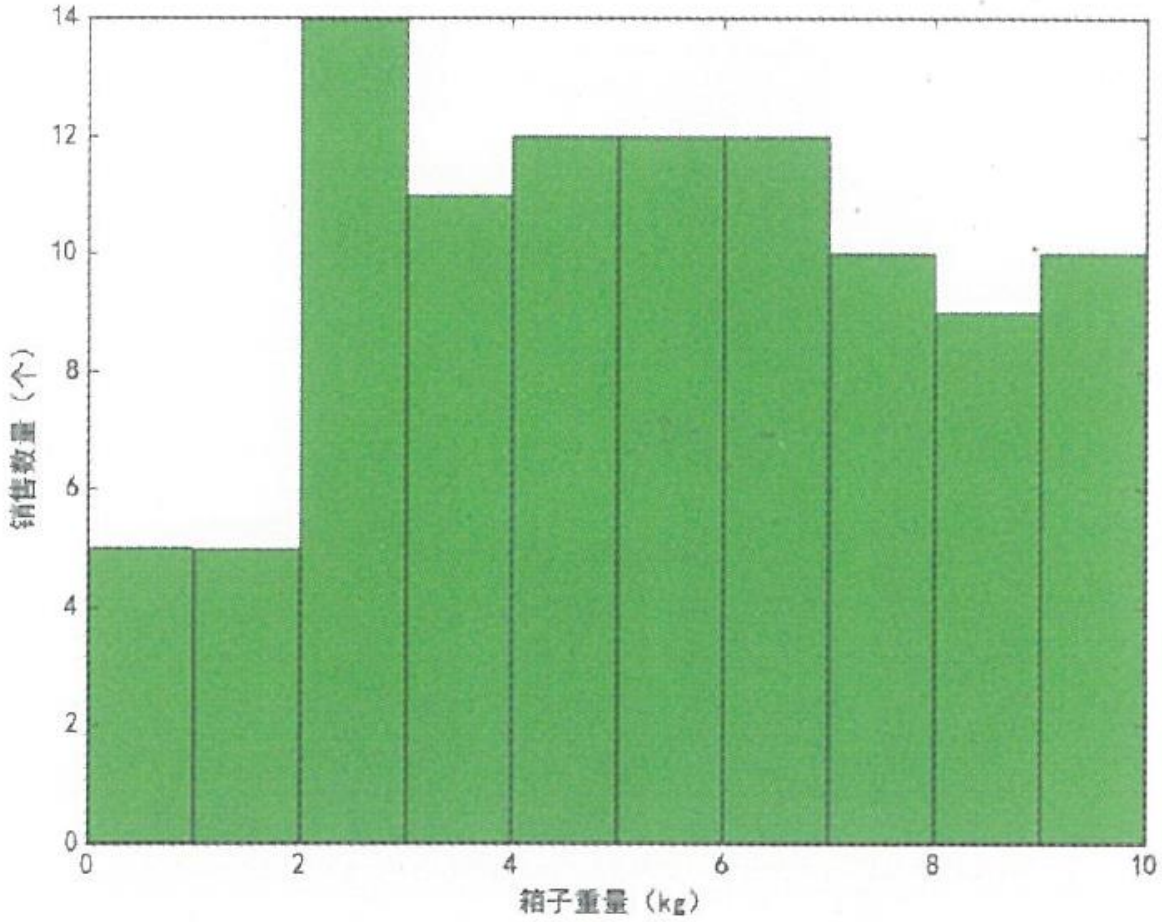
hist()：直方图

```
# -*- coding:utf-8 -*-

import matplotlib as mpl

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

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

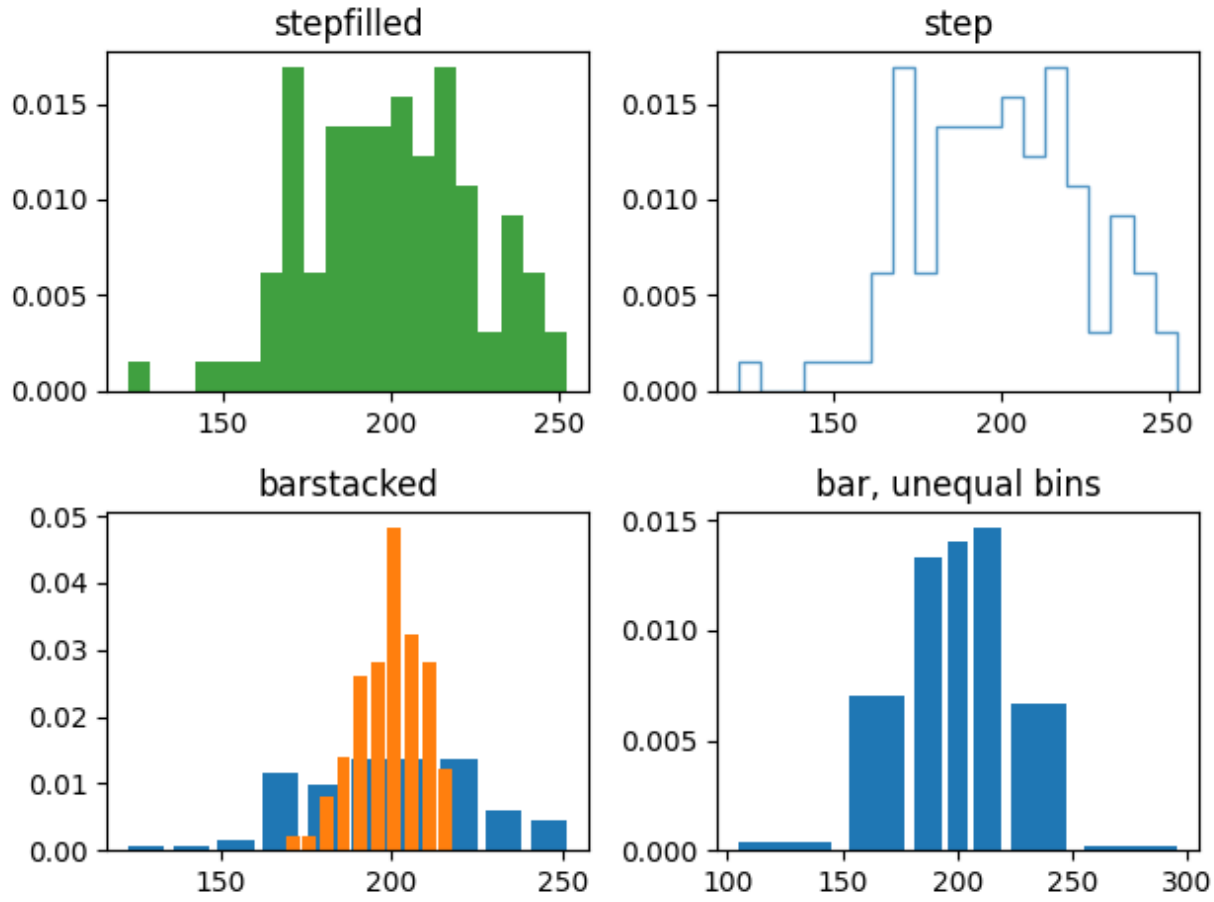
```
# set test scores
boxWeight = np.random.randint(0,10,100)

x = boxWeight

# plot histogram
bins = range(0,11,1)

plt.hist(x,bins=bins,
         color="g",
         histtype="bar",
         rwidth=1,
         alpha=0.6)

# set x,y-axis label
plt.xlabel("箱子重量（kg）")
plt.ylabel("销售数量（个）")

plt.show()
```

histtype: 可选{‘bar’，‘barstacked’，‘step’，‘stepfilled’}之一，默认为bar，推荐使用默认配置，step使用的是梯状，stepfilled则会对梯状内部进行填充，效果与bar类似

hist()：直方图



hist()中的histtype()



直方图 vs. 柱状图 (回顾《数科导》)

?

直方图 vs. 柱状图 (回顾《数科导》)

- 直方图描述了**连续型**数据的分布，柱状图描述了**离散型**数据的分布。
- 直方图的柱体之间**没有空隙**，但柱状图的柱体之间**有空隙**。

pie()：饼图

函数功能：绘制定性数据的不同类别的百分比。

调用签名：plt.pie(x)。

参数说明

- x：定性数据的不同类别的百分比。

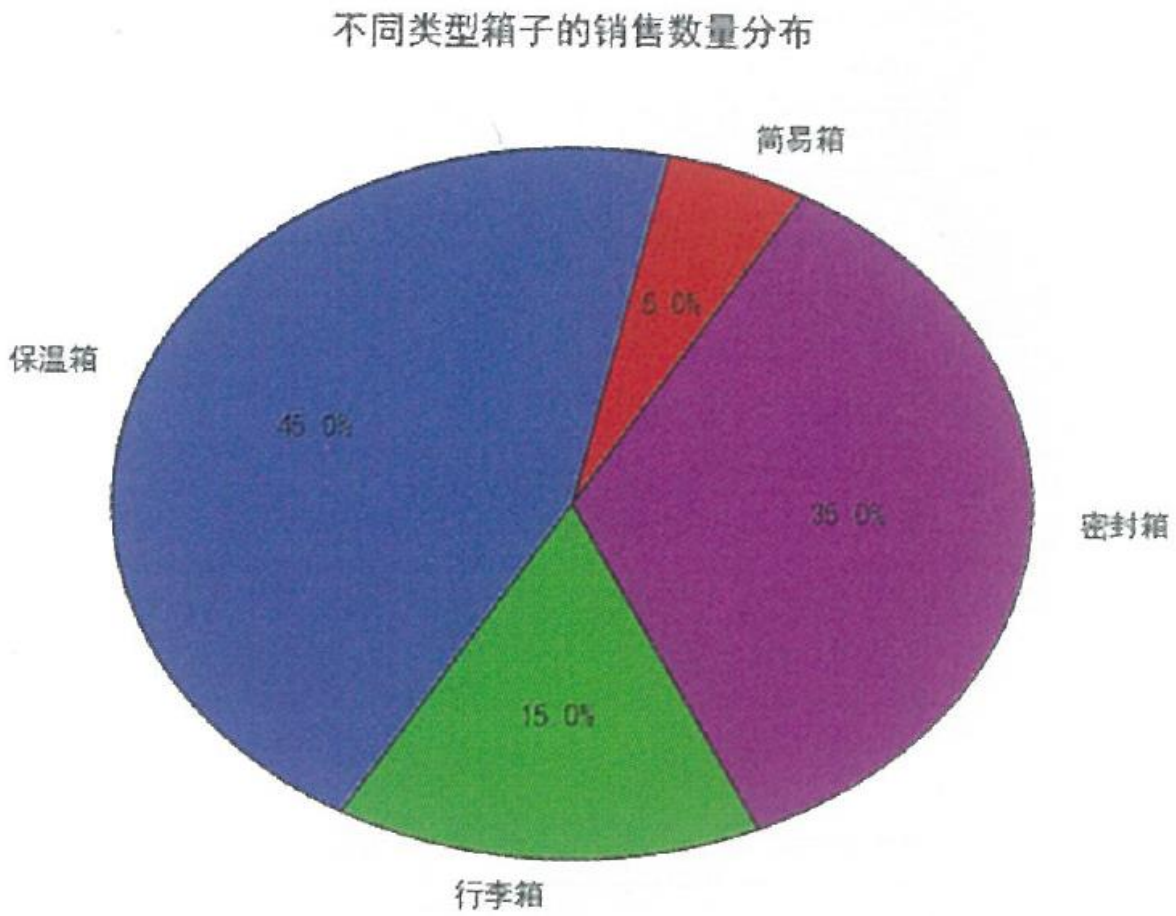
- students：饼片代表的百分比。
- explode：饼片边缘偏离半径的百分比。
- labels：标记每份饼片的文本标签内容。
- autopct：饼片文本标签内容对应的数值百分比样式。
- startangle：从 x 轴作为起始位置，第一个饼片逆时针旋转的角度。
- shadow：是否绘制饼片的阴影。
- colors：饼片的颜色。

pie()：饼图

```
# -*- coding:utf-8 -*-

import matplotlib as mpl
import matplotlib.pyplot as plt

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

kinds = "简易箱","保温箱","行李箱","密封箱"

colors = ["#e41a1c","#377eb8","#4daf4a","#984ea3"]

soldNums = [0.05,0.45,0.15,0.35]

#pie chart
plt.pie(soldNums,
        labels=kinds,
        autopct="%3.1f%%",
        startangle=60,
        colors=colors)

plt.title("不同类型箱子的销售数量占比")

plt.show()
```

startangle：饼块起始角度。浮点数。默认值为0，即从x轴开始。角度逆时针旋转。

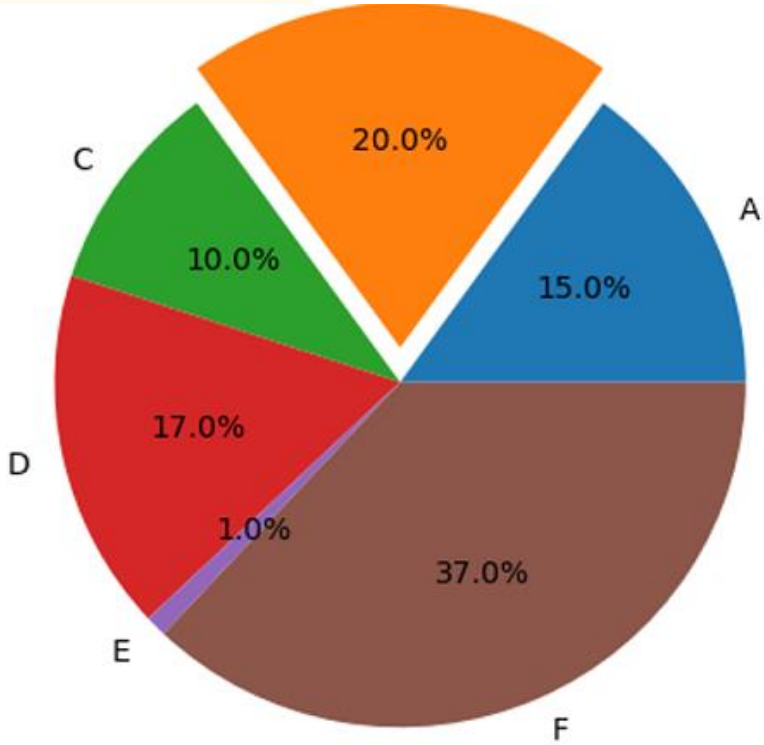
pie()：饼图



pie()：饼图

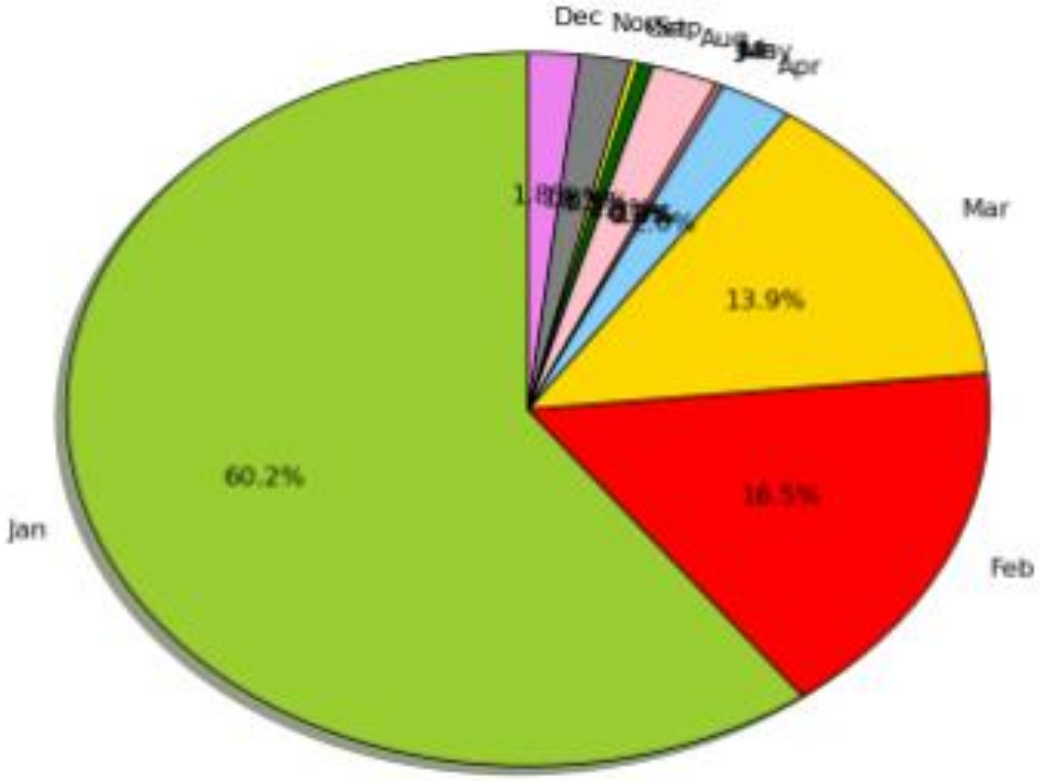
```
labels = 'A', 'B', 'C', 'D', 'E', 'F'
sizes = [15, 20, 10, 17, 1, 37]
explode = (0, 0.1, 0, 0, 0, 0)


plt.figure()
plt.pie(sizes, labels=labels, explode=explode, autopct='%1.1f%%')
# Equal aspect ratio ensures that
# pie is drawn as a circle.
plt.axis('equal')

plt.show()
```

让饼图变圆！



pie()：饼图

- 如果饼图的分块过多，会怎么样？



pie()：饼图

- https://www.itranslater.com/qa/details/2583582768730997760

pie()：内嵌环形饼图【高年级】

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

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

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

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

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

pie()：内嵌环形饼图【高年级】

- pctdistance：饼块内标签与圆心的距离。浮点数。默认值为0.6，autopct不为None该参数生效。
- wedgeprops：饼块属性。

`pie()` 的返回值为三元组。

- `patches`： `matplotlib.patches.Wedge` 对象序列。类型为列表。
- `texts`：外标签 `Text` 对象列表。类型为列表。
- `autotexts`：只有 `autopct` 属性不为 `None` 才会返回值，饼块内标签 `Text` 对象列表。类型为列表。

```
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=inner_colors,
                                    textprops=dict(color="w"),
                                    wedgeprops=dict(width=0.3,edgecolor="w"))
```

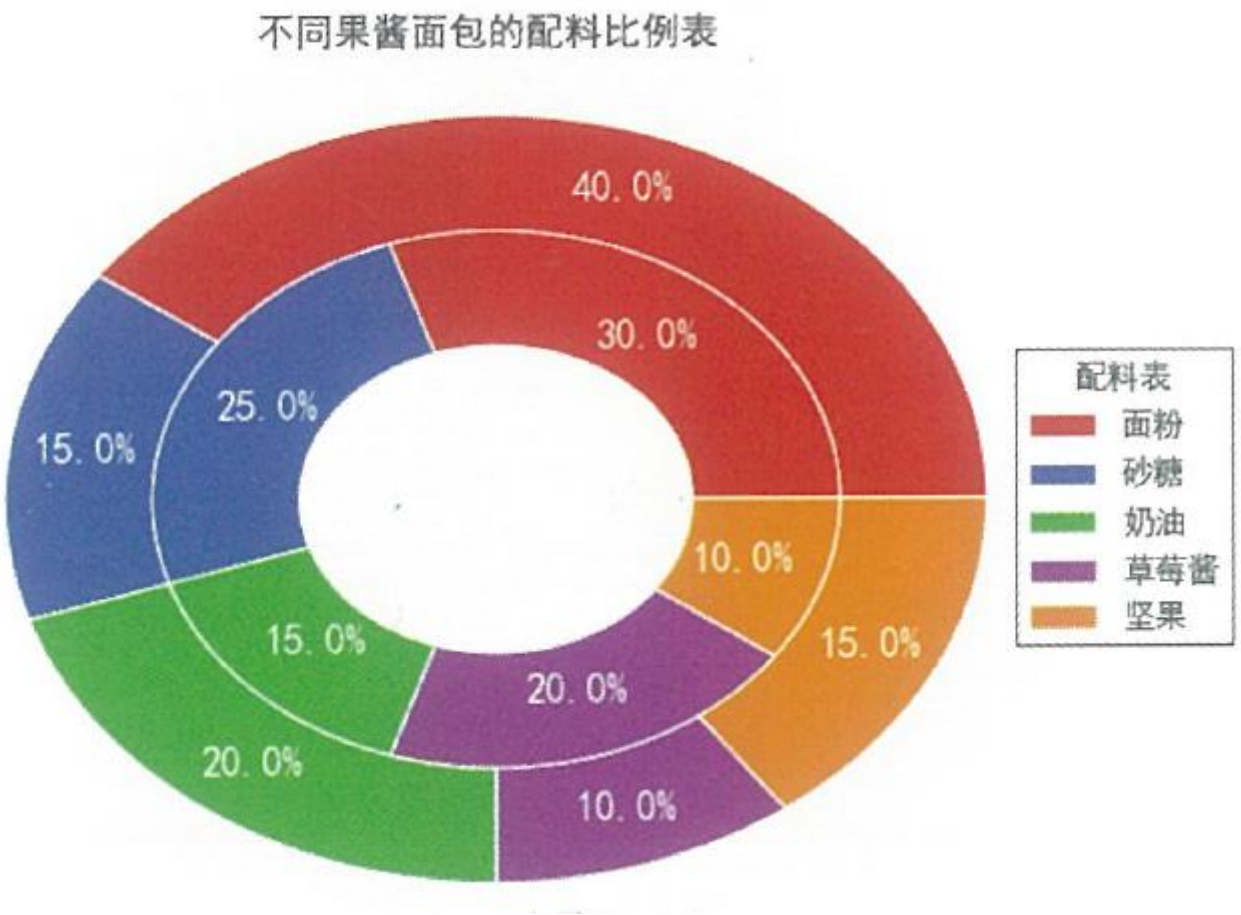
pie()：内嵌环形饼图【高年级】

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

Setp()用于设置属性的值。

```
plt.setp(autotexts1,size=15,weight="bold")
plt.setp(autotexts2,size=15,weight="bold")
plt.setp(texts1,size=12)

plt.title("不同果酱面包配料比例表")

plt.show()
```

pie()：内嵌环形饼图【高年级】



polar()：极线图

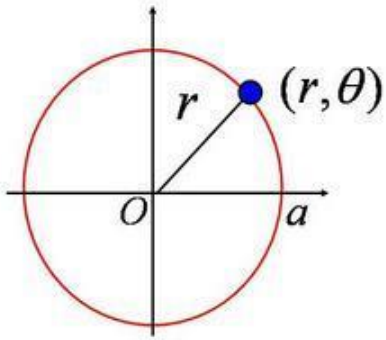
函数功能：在极坐标轴上绘制折线图。

调用签名：plt.polar(theta,r)。

参数说明

- 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。

	直角坐标	极坐标
圆	$x^2+y^2=a^2$	$r=a$



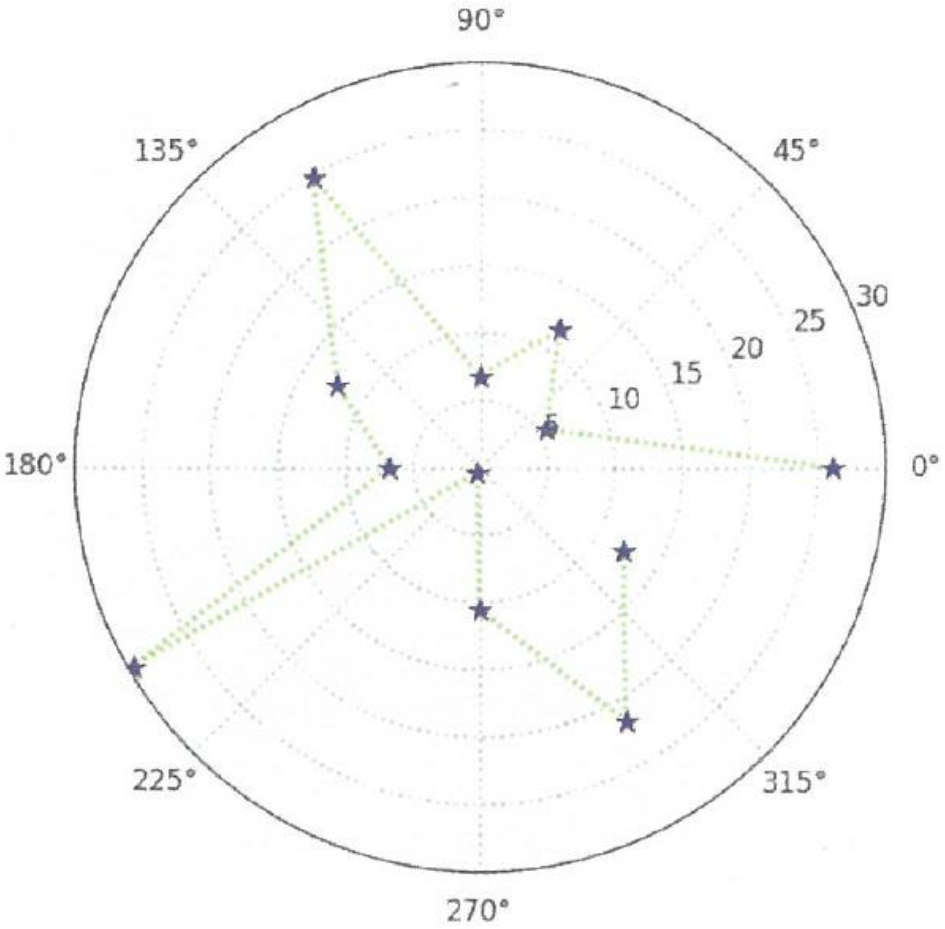
polar()：极线图

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

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()
```

黄绿色

星星的颜色为黑色

polar()：极线图



scatter()：气泡图

函数功能：二维数据借助气泡大小展示三维数据。

调用签名：plt.scatter(x,y)。

参数说明

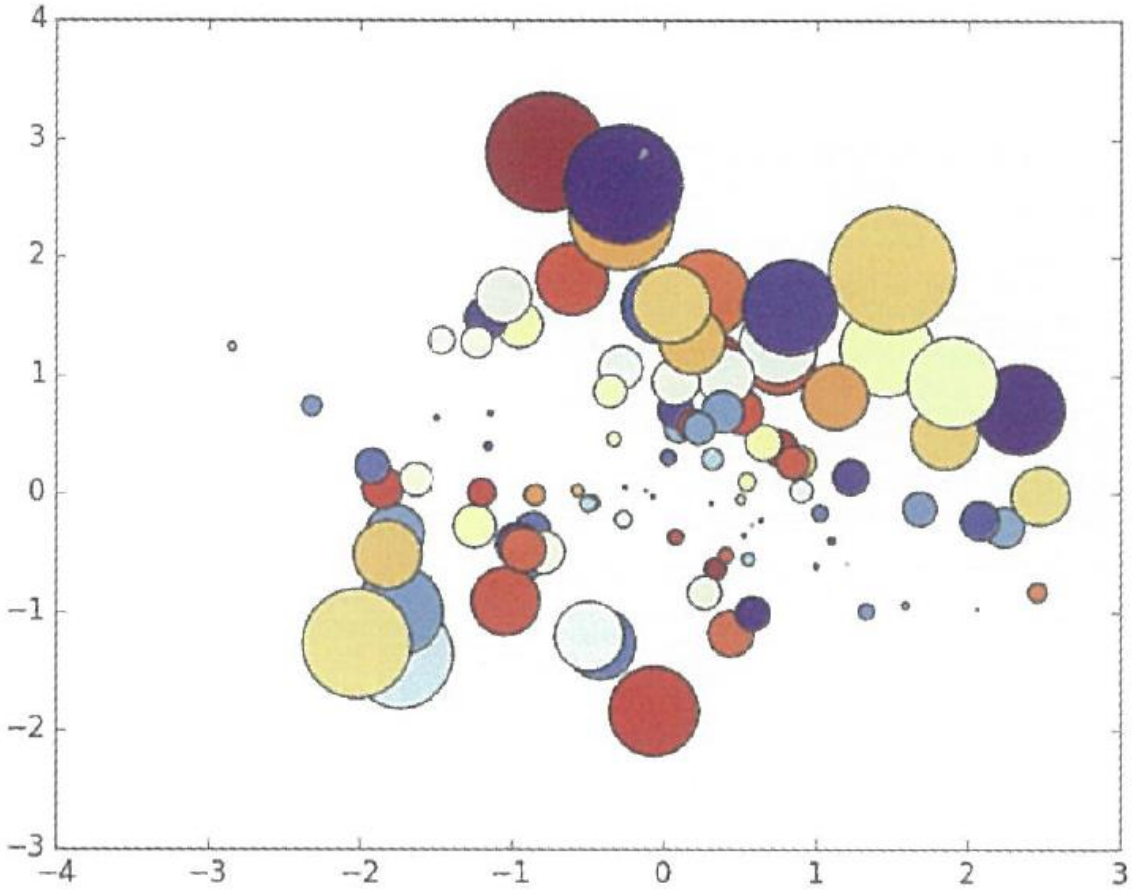
- x：x 轴上的数值。
- y：y 轴上的数值。
- s：散点标记的大小。
- c：散点标记的颜色。
- cmap：将浮点数映射成颜色的颜色映射表。

scatter()：气泡图

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

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

# colormap:RdYlBu
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()
```

Matplotlib内部嵌入的配色方案（colormap）

scatter()：气泡图

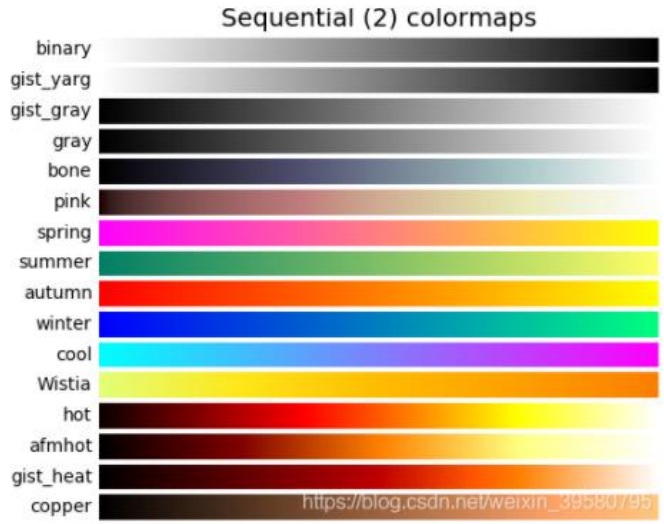


Matplotlib内部嵌入的配色方案（colormap）

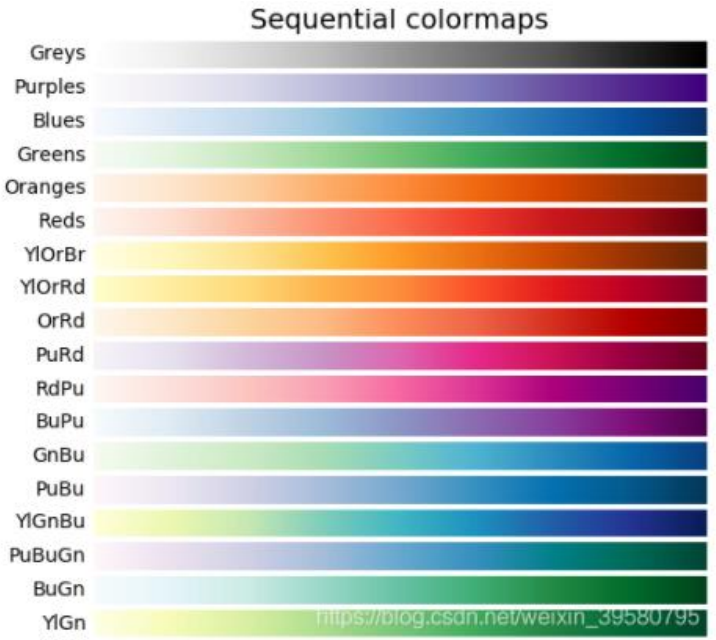
- Sequential colormaps：连续化色图
  - 在两种色调之间近似平滑变化，通常是从低饱和度（如白色）到高饱和度（如明亮的蓝色）。
  - 适用大多数科学数据，可直观地看出数据从低到高的变化。

以中间值颜色命名（如viridis松石绿）

以风格命名

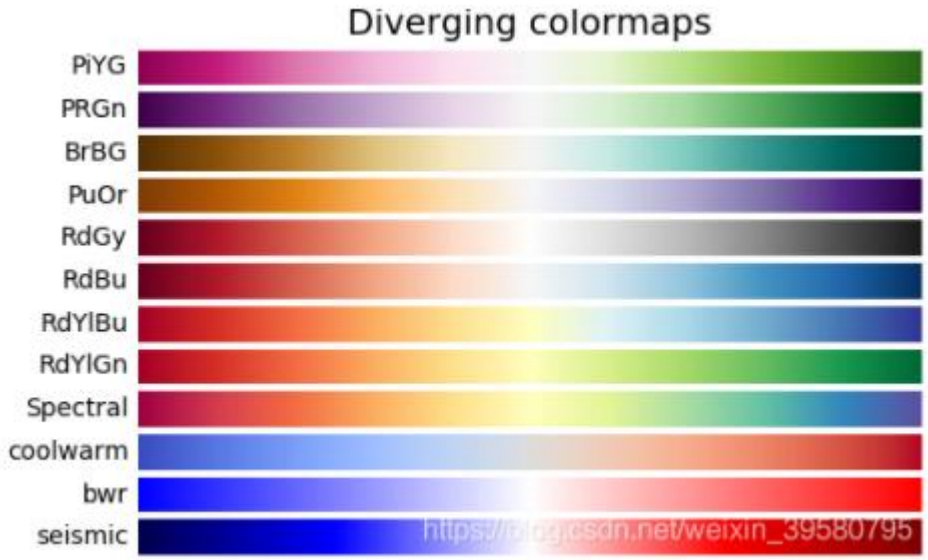


以色系名称命名，由低饱和度到高饱和度过渡（如*YlOrRd* = yellow-orange-red）



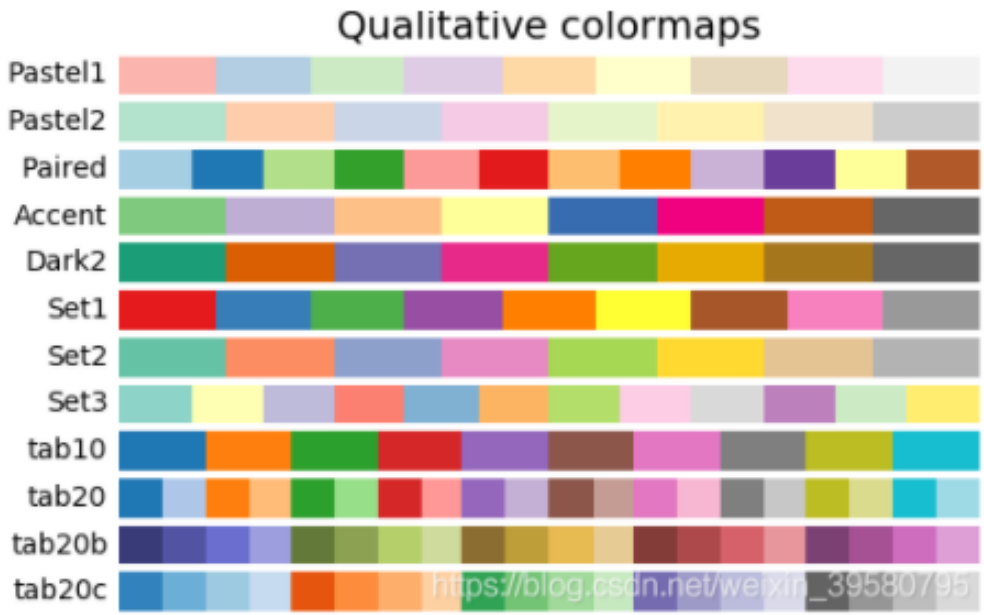
Matplotlib内部嵌入的配色方案（colormap）

- Diverging colormaps：两端发散的色图
  - 具有中间值（通常是浅色），并在高值和低值处平滑变化为两种不同的色调。
  - 适用于数据的中间值很大的情况（例如0，因此正值和负值分别表示为颜色图的不同颜色）。



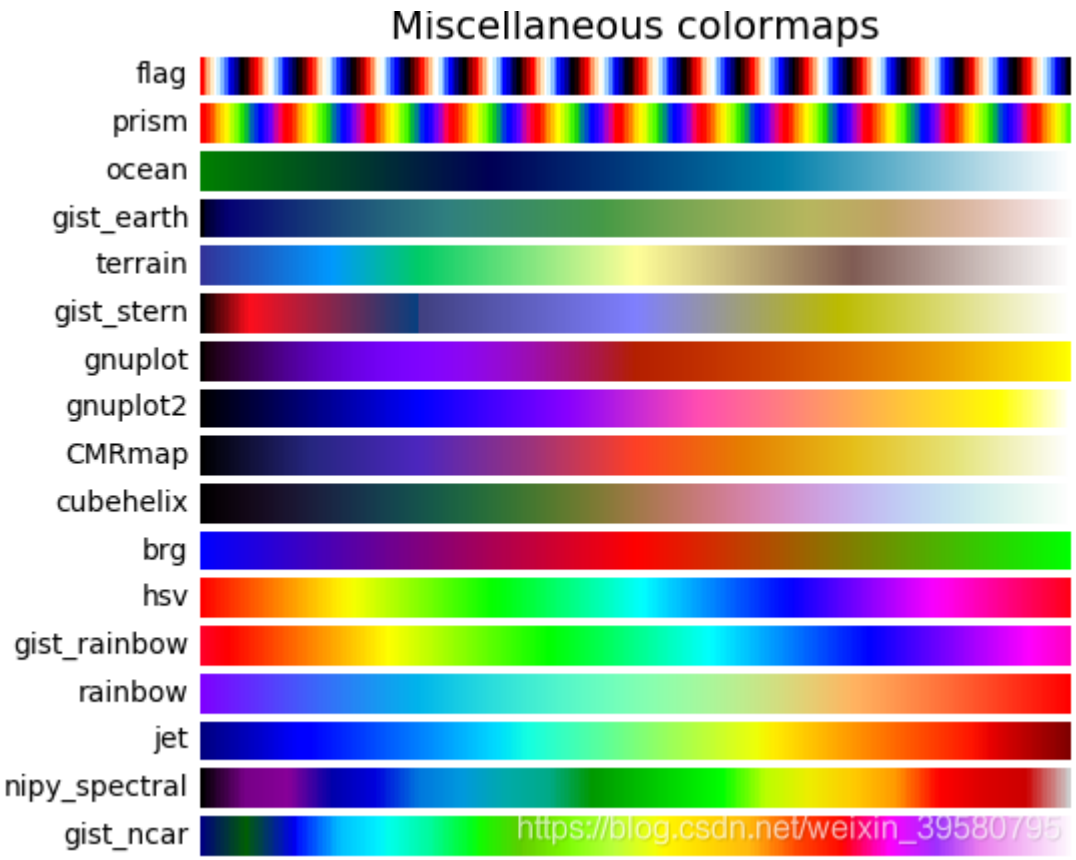
Matplotlib内部嵌入的配色方案（colormap）

- Qualitative colormaps：离散化色图
  - 离散的颜色组合。
  - 在深色背景上绘制一系列线条时，可以在定性色图中选择一组离散的颜色，例如：color_list = plt.cm.Set3(np.linspace(0, 1, 12))。



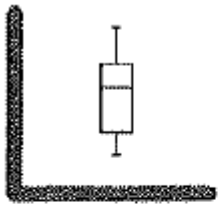
Matplotlib内部嵌入的配色方案（colormap）

- Miscellaneous colormaps：其它色图



boxplot()：箱线图

默认whis=1.5

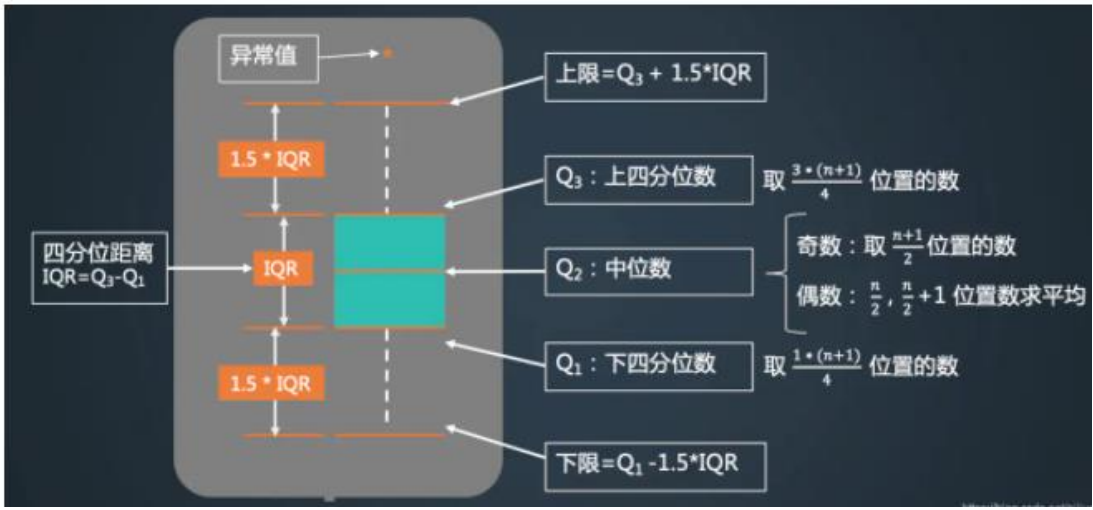


函数功能：绘制箱线图。

调用签名：plt.boxplot(x)。

参数说明

- x：绘制箱线图的输入数据
- testList：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。
- sym：离群值的标记样式。
- labels：绘制每一个数据集的刻度标签。
- patch_artist：是否给箱体添加颜色。

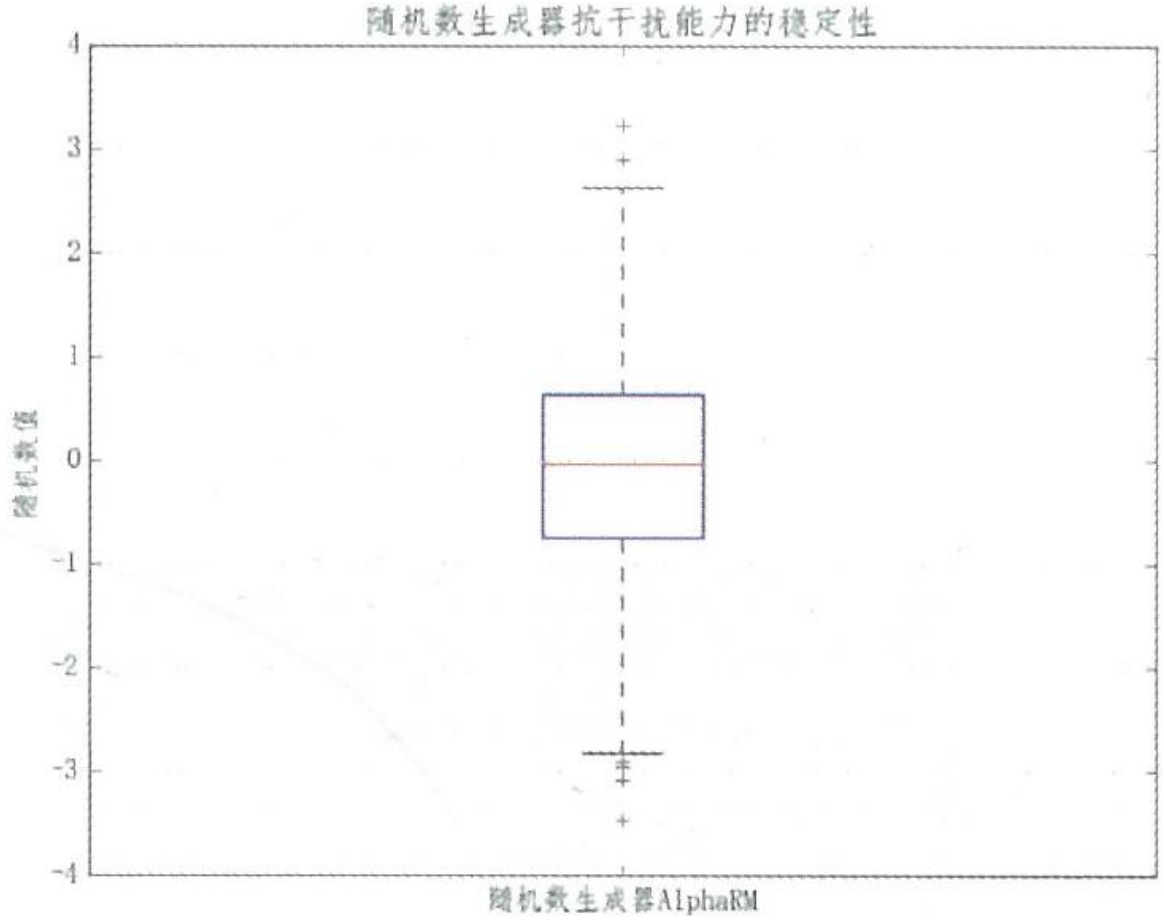


boxplot()：箱线图

```
# -*- coding:utf-8 -*-

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

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

x = np.random.randn(1000)

plt.boxplot(x)

plt.xticks([1],["随机数生成器 AlphaRM"])
plt.ylabel("随机数值")
plt.title("随机数生成器抗干扰能力的稳定性")

plt.grid(axis="y",ls=":",lw=1,color="gray",alpha=0.4)

plt.show()
```

参数1 ticks：The list of tick locations
参数2 labels：The labels to place at the given *ticks* locations

boxplot()：箱线图



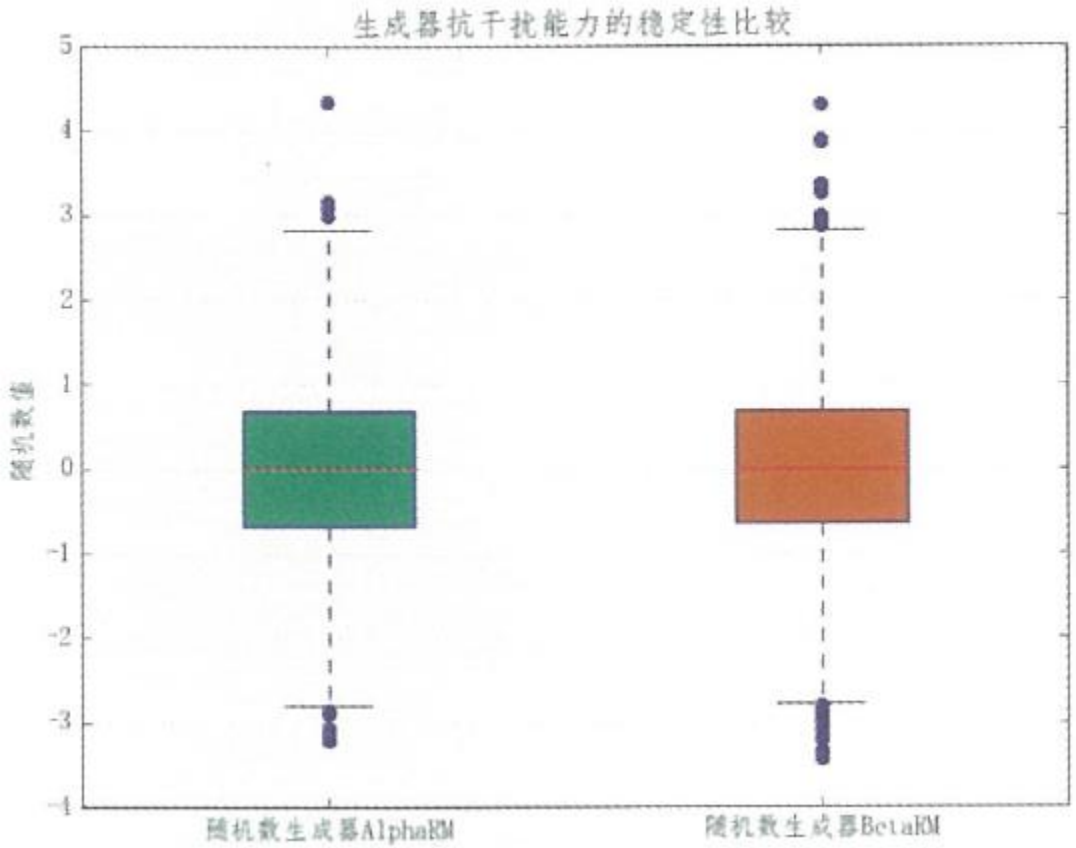
boxplot()：箱线图

```
# -*- coding:utf-8 -*-

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

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

testA = np.random.randn(5000)
testB = np.random.randn(5000)

testList = [testA,testB]
labels = ["随机数生成器 AlphaRM","随机数生成器 BetaRM"]
colors = ["#1b9e77","#d95f02"]

whis = 1.6
width = 0.35
```

```
bplot = plt.boxplot(testList,
                    whis=whis,
                    widths=width,
                    sym="o",
                    labels=labels,
                    patch_artist=True)

for patch,color in zip(bplot["boxes"],colors):
    patch.set_facecolor(color)
```

```
plt.ylabel("随机数值")
plt.title("生成器抗干扰能力的稳定性比较")

plt.grid(axis="y",ls=":",lw=1,color="gray",alpha=0.4)

plt.show()
```

boxplot()：箱线图


生成器抗干扰能力的稳定性比较
随机数值
5
4
3
2
1
0
-1
-2
-3
-4
随机数生成器AlphaRM
随机数生成器BetaRM

【小结】这些图如何画？

- 散点图、气泡图
- 折线图（含堆积折线图）
- 柱状图（含堆叠柱状图）
- 直方图
- 阶梯图
- 雷达图（极线图）
- 箱线图

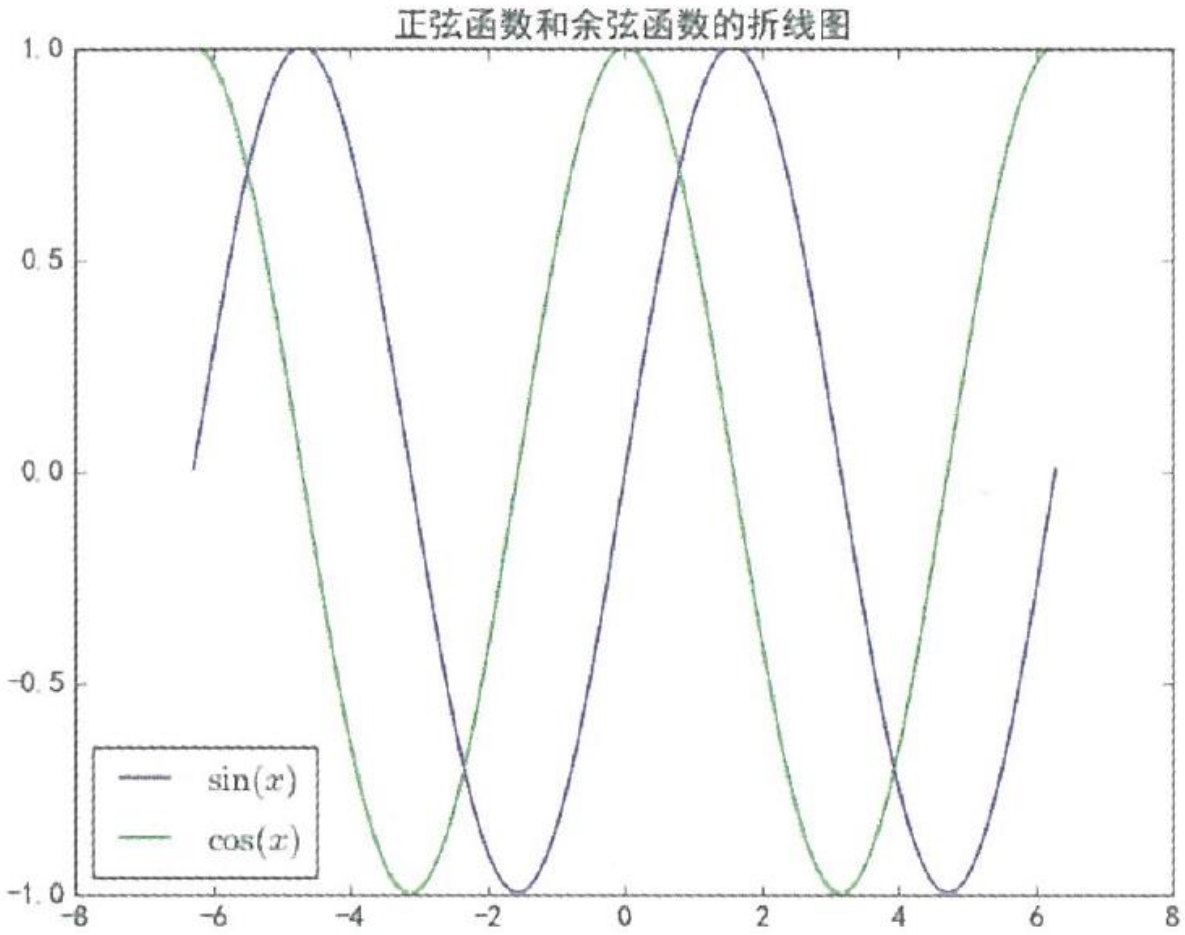
完善统计图形

添加图例和标题

```
# -*- coding:utf-8 -*-

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

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

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

plt.plot(x,y,label=r"$\sin(x)$")
plt.plot(x,y1,label=r"$\cos(x)$")

plt.legend(loc="lower left")

plt.title("正弦函数和余弦函数的折线图")

plt.show()
```

防止中文或者是负号无法显示

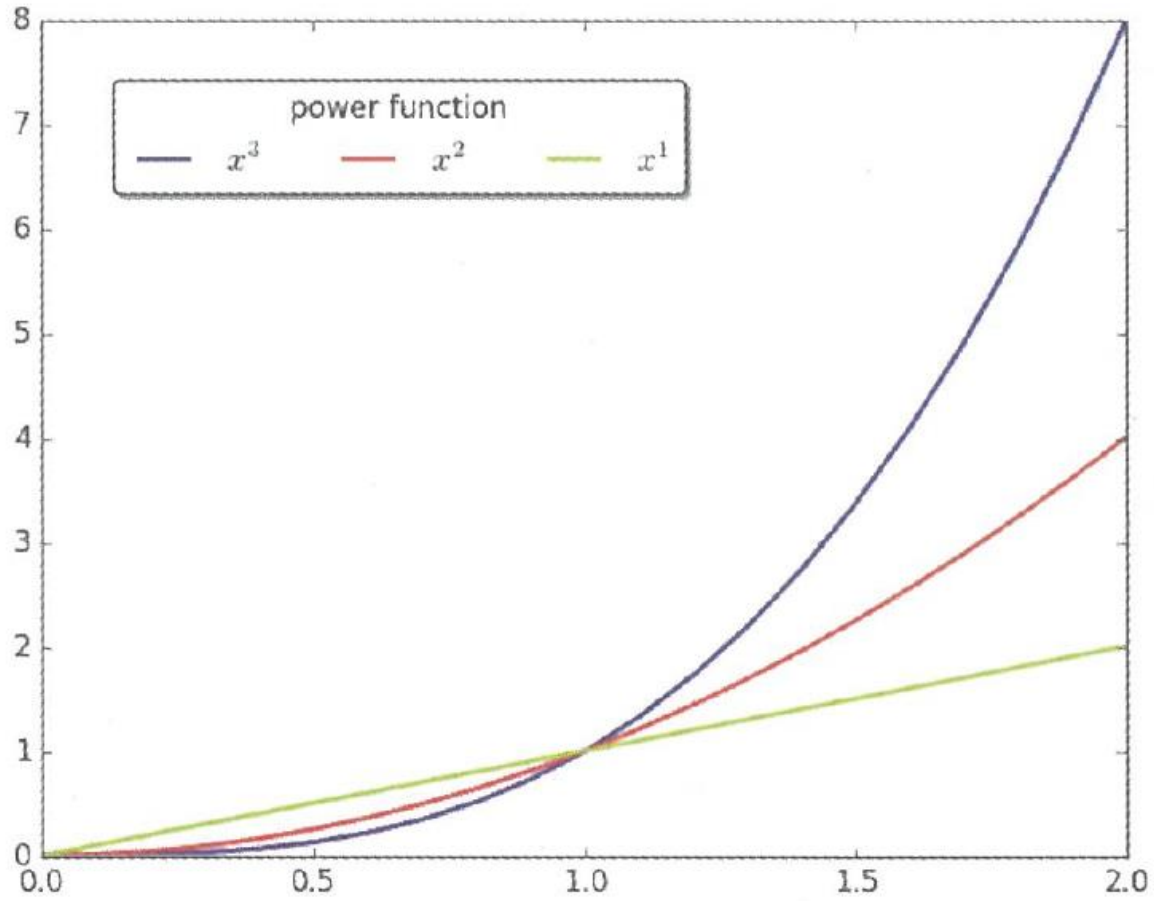
添加图例和标题



添加图例和标题

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

x = np.arange(0,2.1,0.1)
y = np.power(x,3)
y1 = np.power(x,2)
y2 = np.power(x,1)

plt.plot(x,y,ls="-",lw=2,label="$x^{3}$")
plt.plot(x,y1,ls="-",lw=2,c="r",label="$x^{2}$")
plt.plot(x,y2,ls="-",lw=2,c="y",label="$x^{1}$")

plt.legend(loc="upper left",bbox_to_anchor=(0.05,0.95),ncol=3,
           title="power function",shadow=True,fancybox=True)

plt.show()
```

添加图例和标题

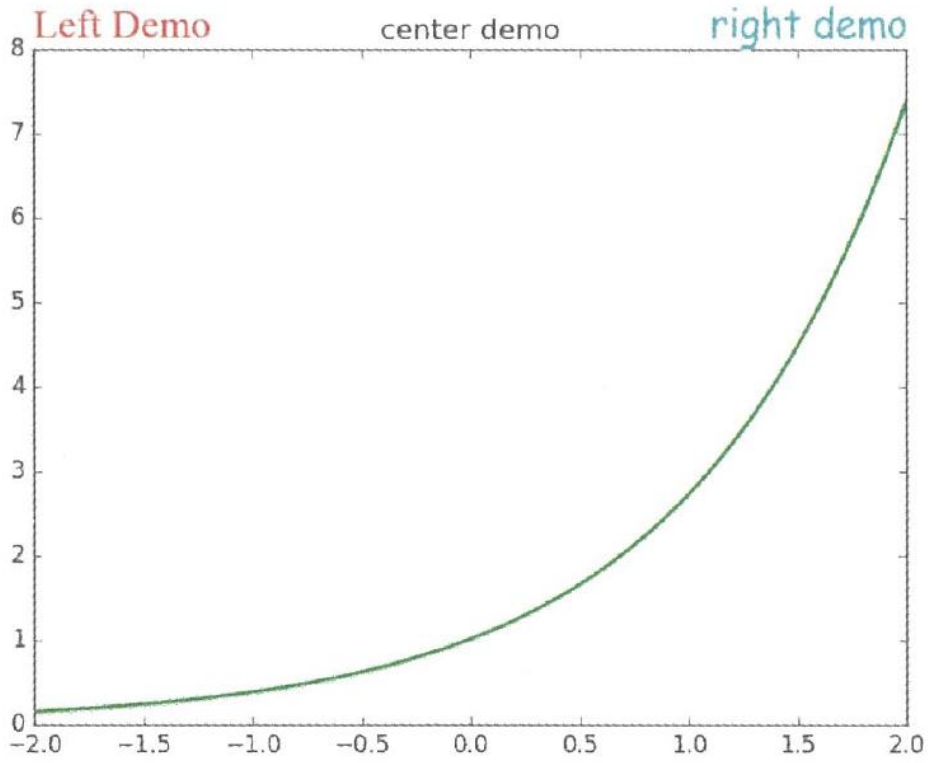


添加图例和标题

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

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()
```

添加图例和标题

标题函数 title()的关键字参数主要集中在标题位置参数和标题文本格式参数，标题位置参数值有“left”“center”和“right”。标题文本格式参数主要是字体类别（family）、字体大小（size）、字体颜色（color）、字体风格（style）等，这些文本格式参数可以放在关键字参数 fontdict 的字典中存储，也可以分别作为标题函数 title()的关键字参数。



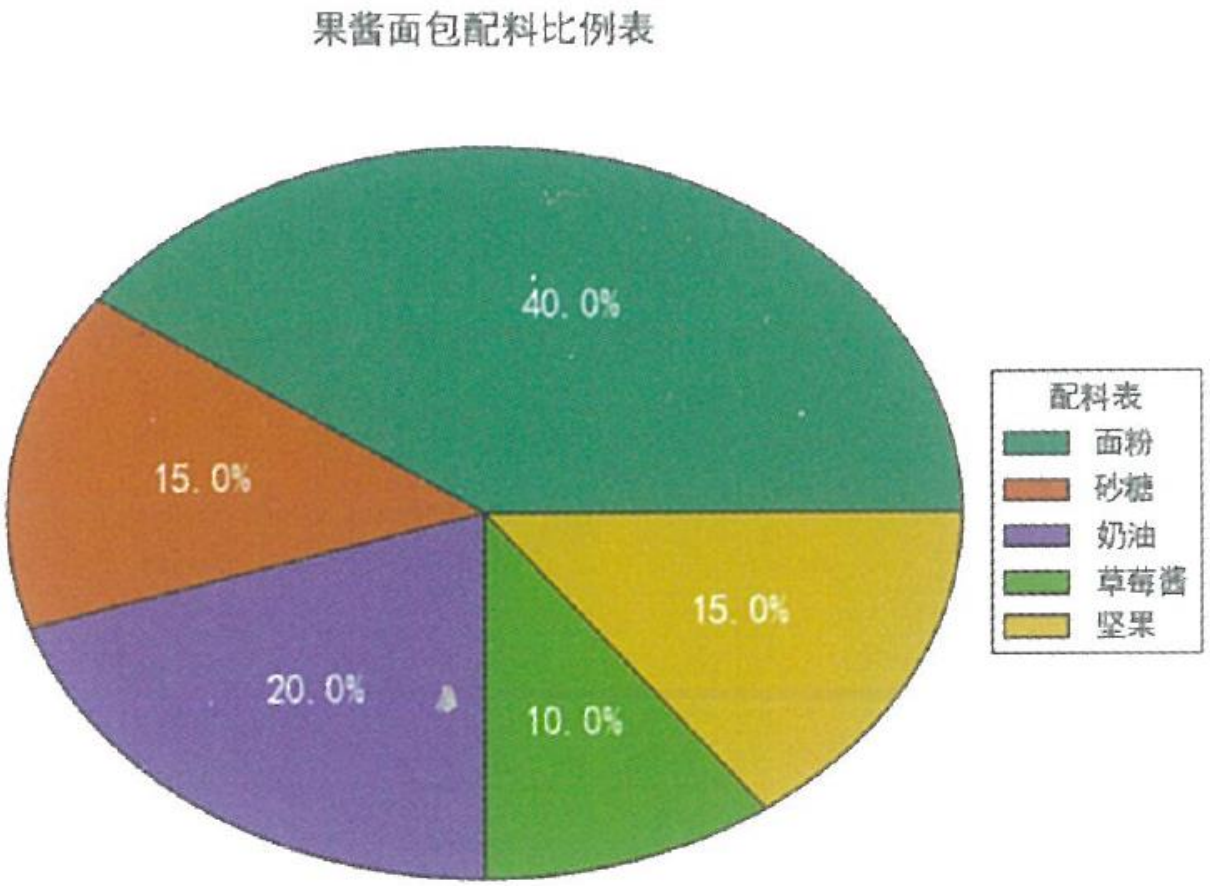
添加图例和标题

```
# -*- coding:utf-8 -*-

import matplotlib as mpl
import matplotlib.pyplot as plt

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

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

weight = [40,15,20,10,15]

colors = ["#1b9e77","#d95f02","#7570b3","#66a61e","#e6ab02"]

wedges, texts, autotexts = plt.pie(weight,
                                   autopct="%3.1f%%",
                                   textprops=dict(color="w"),
                                   colors=colors)
```

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

```
plt.setp(autotexts, size=15, weight="bold")
plt.setp(texts,size=12)

plt.title("果酱面包配料比例表")
```

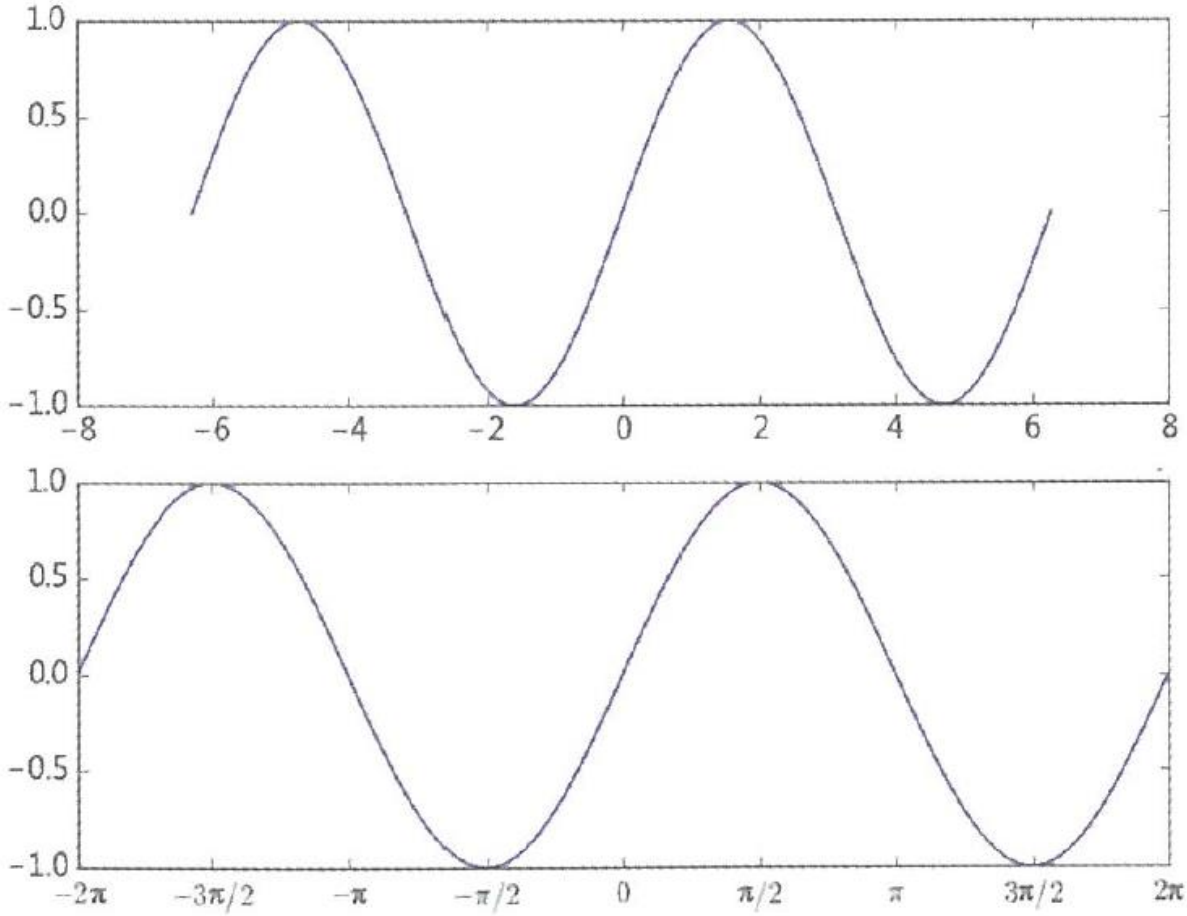
添加图例和标题



调整刻度范围和刻度标签

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

x = np.linspace(-2*np.pi,2*np.pi,200)
y = np.sin(x)

# set subplot(211)
plt.subplot(211)

# plot figure
plt.plot(x,y)

# set subplot(212)
plt.subplot(212)

# set xlimit
plt.xlim(-2*np.pi,2*np.pi)

    # set ticks
    plt.xticks([-2*np.pi,-3*np.pi/2,-1*np.pi,-1*(np.pi)/2,0,(np.pi)/2,np.pi,
3*np.pi/2,2*np.pi],

[r"$-2\pi$",r"$-3\pi/2$",r"$-\pi$",r"$-\pi/2$",r"$0$",r"$\pi/2$",r"$\pi$",r"
$3\pi/2$",r"$2\pi$"])

# plot figure
plt.plot(x,y)

plt.show()
```

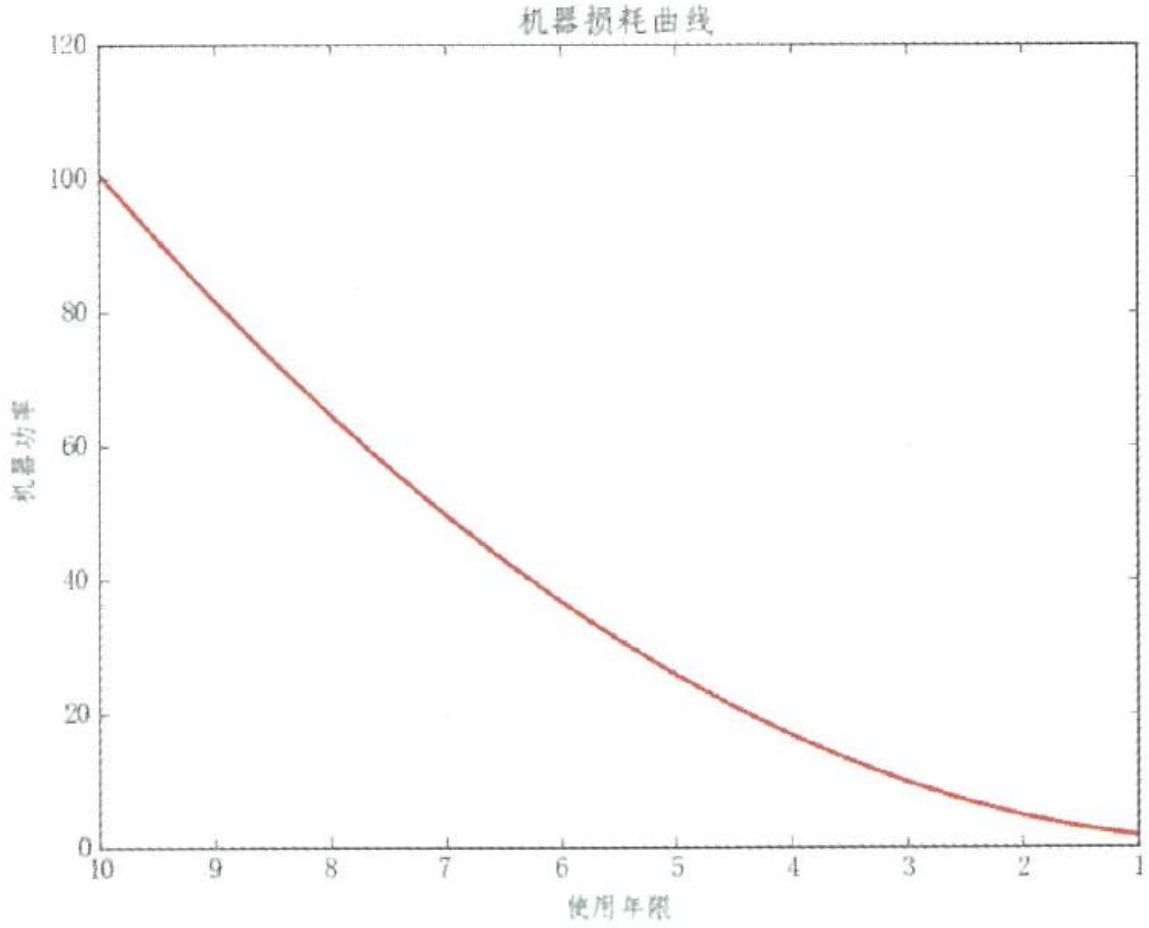
调整刻度范围和刻度标签



调整刻度范围和刻度标签

```
# -*- coding:utf-8 -*-

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

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

time = np.arange(1,11,0.5)
machinePower = np.power(time,2)+0.7

plt.plot(time,machinePower,
        linestyle="-",
        linewidth=2,
        color="r")

plt.xlim(10,1)

plt.xlabel("使用年限")
plt.ylabel("机器功率")

plt.title("机器损耗曲线")

plt.grid(ls=":",lw=1,color="gray",alpha=0.5)

plt.show()
```

调整刻度范围和刻度标签



向统计图形添加表格

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

labels = "A 难度水平","B 难度水平","C 难度水平","D 难度水平"

students = [0.35,0.15,0.20,0.30]

explode = (0.1,0.1,0.1,0.1)

colors = ["#377eb8","#e41a1c","#4daf4a","#984ea3"]

# exploded pie chart
plt.pie(students,
        explode=explode,
        labels=labels,
        autopct="%1.1f%%",
        startangle=45,
        shadow=True,
        colors=colors)

plt.title("选择不同难度测试试卷的学生占比")
```

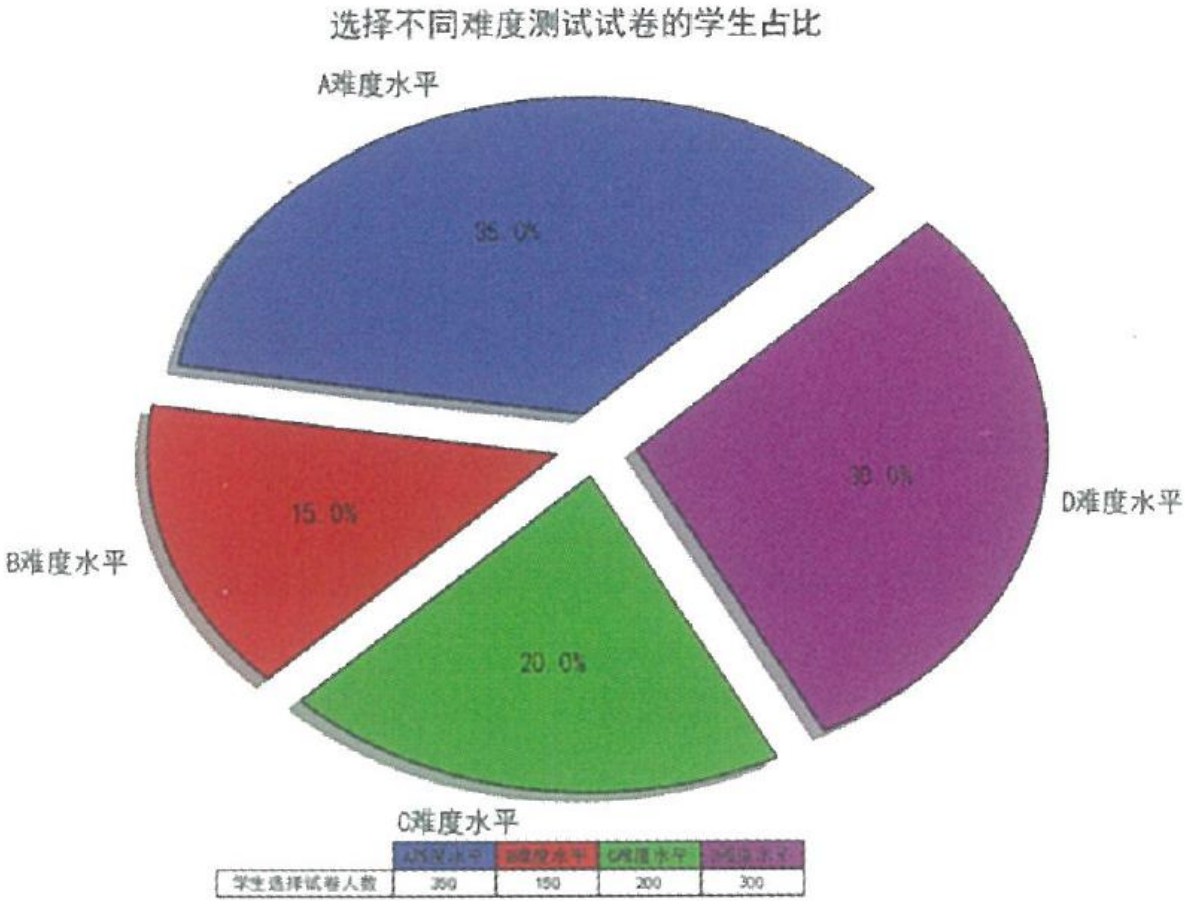
饼片边缘偏离中心多远？

向统计图形添加表格

```
# add table to pie figure
colLabels = ["A 难度水平","B 难度水平","C 难度水平","D 难度水平"]
rowLabels = ["学生选择试卷人数"]
studentValues = [[350,150,200,300]]
colColors = ["#377eb8","#e41a1c","#4daf4a","#984ea3"]

plt.table(cellText=studentValues,
          cellLoc="center",
          colWidths=[0.1]*4,
          colLabels=colLabels,
          colColours=colColors,
          rowLabels=rowLabels,
          rowLoc="center",
          loc="bottom")

plt.show()
```

- cellText：表格的数值，将源数据按照行进行分组，每组数据放在列表里存储，所有组数据再放在列表里储存。
- cellLoc：表格中的数据对齐位置，可以左对齐、居中和右对齐。
- 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，可以左对齐、居中和右对齐。
- loc：表格在画布中的位置。

向统计图形添加表格


选择不同难度测试试卷的学生占比
A难度水平
35.0%
B难度水平
15.0%
C难度水平
20.0%
D难度水平
30.0%
学生选择试卷人数
300
150
200
300

设置坐标轴的刻度样式

- 刻度定位器
- 刻度格式器

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.ticker import AutoMinorLocator, MultipleLocator, FuncFormatter
```

设置坐标轴的刻度样式

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

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

# set x y-major_tick_locator
ax.xaxis.set_major_locator(MultipleLocator(1.0))
ax.yaxis.set_major_locator(MultipleLocator(1.0))

# set x,y-minor_tick_locator
ax.xaxis.set_minor_locator(AutoMinorLocator(4))
ax.yaxis.set_minor_locator(AutoMinorLocator(4))
```

在x轴的1倍处分别设置主刻度线

红色下划线部分获得y轴的实例

每一份主刻度线区间等分4份，作为次要刻度线

设置坐标轴的刻度样式

```
# set x-minor_tick_formatter

def minor_tick(x, pos):  # n % n = 0; m % n = m(m<n)
    if not x % 1.0:
        return ""
    return "%.2f" % x

ax.xaxis.set_minor_formatter(FuncFormatter(minor_tick))
```

次要刻度线显示位置的精度

```
# change the appearance of ticks and tick labels
ax.tick_params("y",which='major',
               length=15,width=2.0,
               colors="r")
ax.tick_params(which='minor',
               length=5,width=1.0,
               labelsize=10, labelcolor='0.25')
```

主要刻度样式的设置

次要刻度样式的设置

- which：设置主刻度的样式。
- length：设置主刻度线的长度。
- width：设置主刻度线的宽度。
- colors：设置主刻度线和主刻度标签的颜色。

次要刻度样式的设置方法的具体执行语句是“ax.tick_params(which='minor', length=5,width=1.0,labelsize=10, labelcolor='0.25')”，执行语句中实例方法 tick_params()的关键字参数的含义如下。

- which：设置次要刻度的样式。
- length：设置次要刻度线的长度。
- width：设置次要刻度线的宽度。
- labelsize：设置次要刻度标签的大小。
- labelcolor：设置次要刻度标签的颜色。

设置坐标轴的刻度样式

```
# set x,y_axis_limit
ax.set_xlim(0,4)
ax.set_ylim(0,2)

# plot subplot
ax.plot(x,y, c=(0.25, 0.25, 1.00), lw=2, zorder=10) # pair 0
#ax.plot(x,y, c=(0.25, 0.25, 1.00), lw=2, zorder=0) # pair 1

# set grid
ax.grid(linestyle="-", linewidth=0.5, color='r', zorder=0) # pair 0
#ax.grid(linestyle="-", linewidth=0.5, color='r', zorder=10) # pair 1
#ax.grid(linestyle="--", linewidth=0.5, color='.25', zorder=0) # only one
```

设置坐标轴的刻度样式

画布和绘图区域

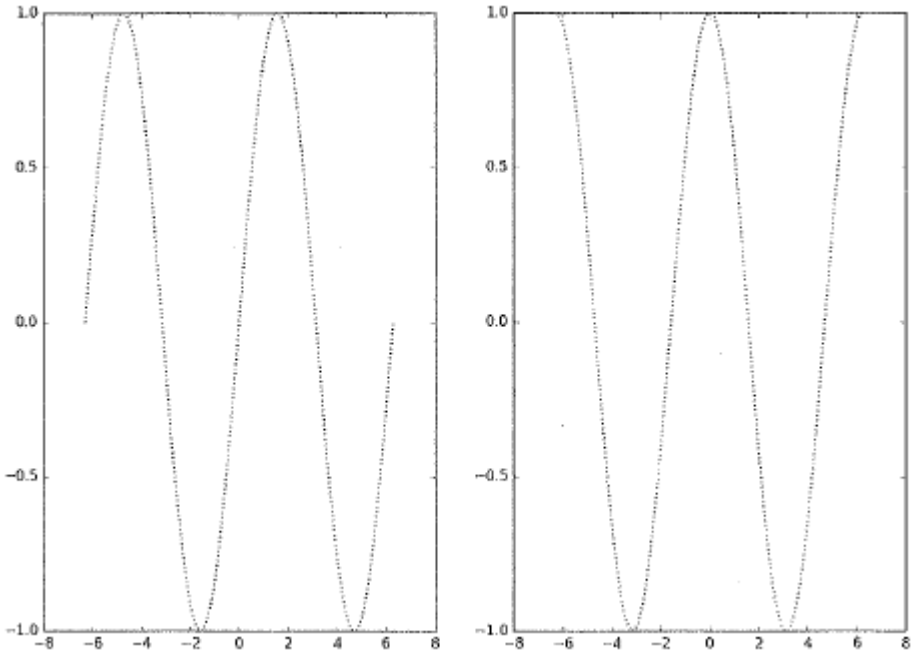
subplot()：绘制网格区域中的几何形状相同的子区布局

```
subplot(numRows,numCols,plotNum)
```

如果子区函数 subplot()的三个参数分别是整数 C、整数 R 和整数 P，即 subplot(C,R,P)，那么这三个整数就表示在 C 行、R 列的网格布局上，子区 subplot()会被放置在第 P 个位置上，即为将被创建的子区编号，子区编号从 1 开始，起始于左上角，序号依次向右递增。也就是说，每行的子区位置都是从左向右进行升序计数的，即 subplot(2,3,4)是第 2 行的第 1 个子区。

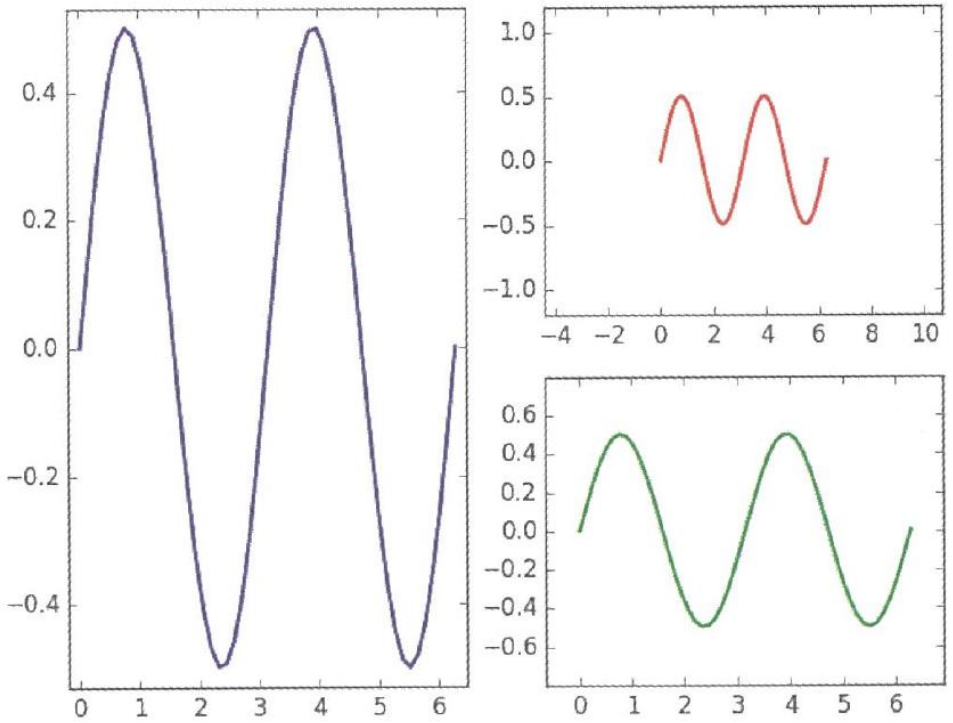
subplot()

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

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

# set figure #1
plt.subplot(121)

plt.plot(x,y)

# set figure #2
plt.subplot(122)

plt.plot(x,y1)

plt.show()
```



subplot()

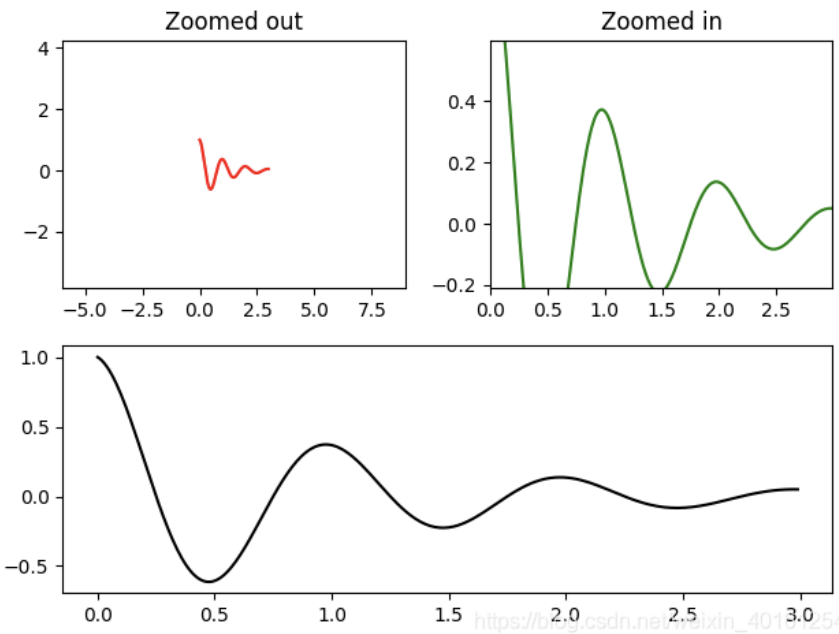
```
import matplotlib.pyplot as plt
import numpy as np

fig = plt.figure()

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

ax1 = fig.add_subplot(121)
ax1.margins(0.03)
ax1.plot(x,y,ls="-",lw=2,color="b")

ax2 = fig.add_subplot(222)
ax2.margins(0.7,0.7)
ax2.plot(x,y,ls="-",lw=2,color="r")

ax3 = fig.add_subplot(224)
ax3.margins(x=0.1,y=0.3)
ax3.plot(x,y,ls="-",lw=2,color="g")

plt.show()
```



subplot()

```
import matplotlib.pyplot
import numpy as np

fig = plt.figure()

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

ax1 = fig.add_subplot(121)
ax1.margins(0.03)
ax1.plot(x,y,ls="-",lw=2,color="b")

ax2 = fig.add_subplot(222)
ax2.margins(0.7,0.7)
ax2.plot(x,y,ls="-",lw=2,color="r")

ax3 = fig.add_subplot(224)
ax3.margins(x=0.1,y=0.3)
ax3.plot(x,y,ls="-",lw=2,color="g")

plt.show()
```

```
ax1 = plt.subplot(212)
ax1.margins(0.05)           # Default margin is 0.05, value 0 means fit
ax1.plot(t1, f(t1), 'k')

ax2 = plt.subplot(221)
ax2.margins(2, 2)           # Values >0.0 zoom out
ax2.plot(t1, f(t1), 'r')
ax2.set_title('Zoomed out')

ax3 = plt.subplot(222)
ax3.margins(x=0, y=-0.25)   # Values in (-0.5, 0.0) zooms in to center
ax3.plot(t1, f(t1), 'g')
ax3.set_title('Zoomed in')
```



Matplotlib部分课堂练习

- Matplotlib-A：答案相对固定的练习
- Matplotlib-B：答案相对开放的练习

谢谢！