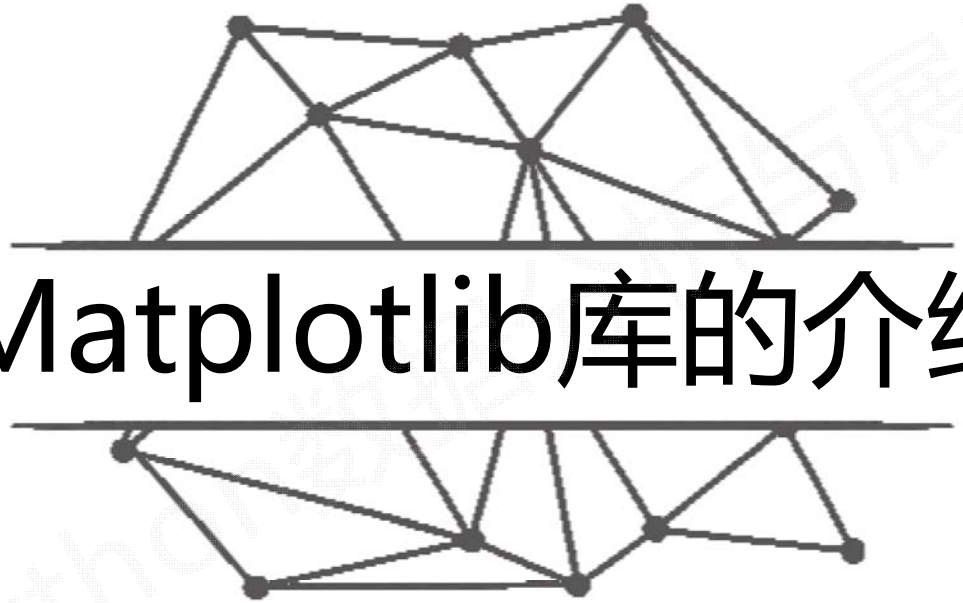
Matplotlib库入门



Matplotlib库的介绍

matplotlib

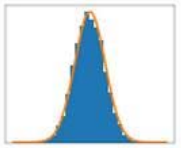
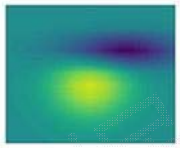
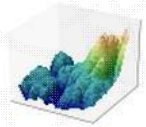
Fork me on GitHub

home | examples | gallery | pyplot | docs »

modules | index

Introduction

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shell, the jupyter notebook, web application servers, and four graphical user interface toolkits.

Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code. For a sampling, see the screenshots, thumbnail gallery, and examples directory

For simple plotting the `pyplot` module provides a MATLAB-like interface, particularly when combined with `IPython`. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Installation

Visit the Matplotlib installation instructions.

Documentation

This is the documentation for Matplotlib version 2.0.0.

Other versions are available:

- 2.0.0 Latest stable version.
- 2.x Latest git master (unstable)
- 1.5.3 Previous stable version.
- 1.4.3 Older stable version.

Depsy 100th percentile

Travis-CI: build passing

Support matplotlib

Support NumFOCUS

Quick search

Go

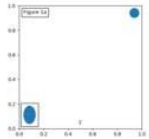
Python优秀的数据

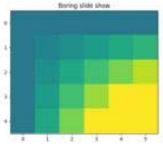
可视化第三方库

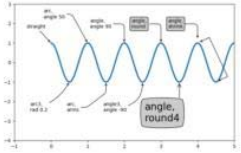
Matplotlib库的效果

http://matplotlib.org/gallery.html

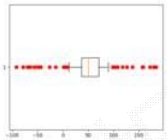

alignment_test

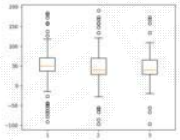

anchored_artists

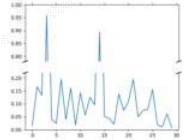

animation_demo

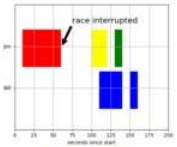


annotation_demo2

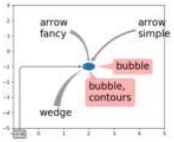

boxplot_demo


boxplot_demo

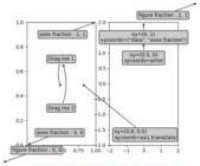

broken_axis

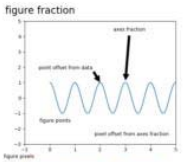


broken_barh

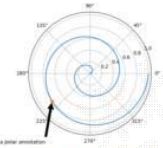


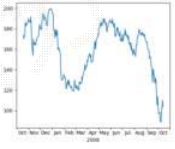
annotation_demo2

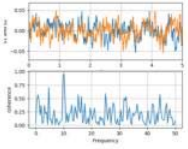

annotation_demo3

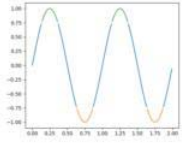


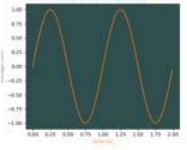
annotation_demo

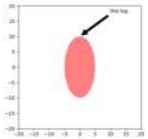

annotation_demo

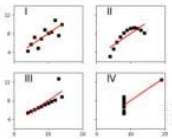

centered_ticklabels

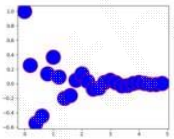

cohere_demo


color_by_yvalue

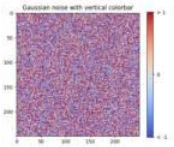

color_demo

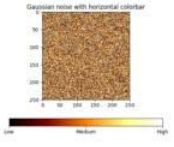

annotation_demo

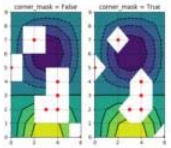

anscombe

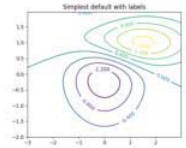

arctest

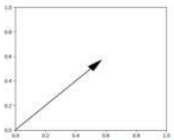
arrow_demo

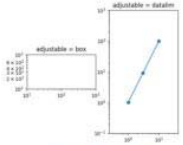

colorbar_tick_labelling_demo

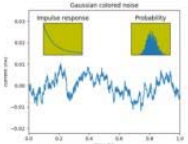

colorbar_tick_labelling_demo

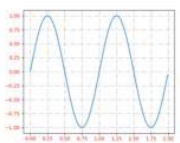

contour_corner_mask

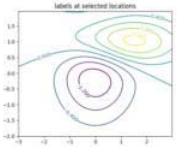

contour_demo

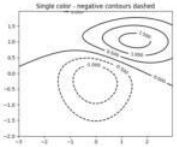

arrow_simple_demo

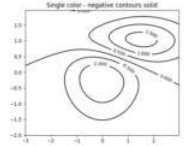

aspect_loglog

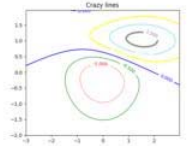

axes_demo


axes_props


contour_demo


contour_demo


contour_demo


contour_demo

Matplotlib库的使用

Matplotlib库由各种可视化类构成，内部结构复杂，受Matlab启发

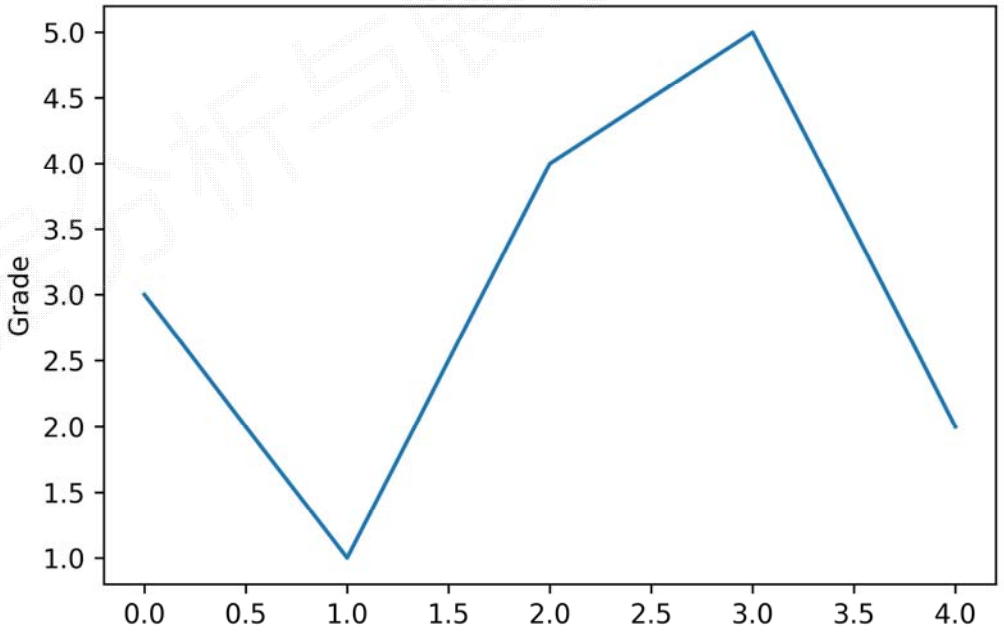
matplotlib.pyplot是绘制各类可视化图形的命令子库，相当于快捷方式

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

引入模块的别名

Matplotlib库小测

```
import matplotlib.pyplot as plt

plt.plot([3, 1, 4, 5, 2])
plt.ylabel("Grade")
plt.show()
```

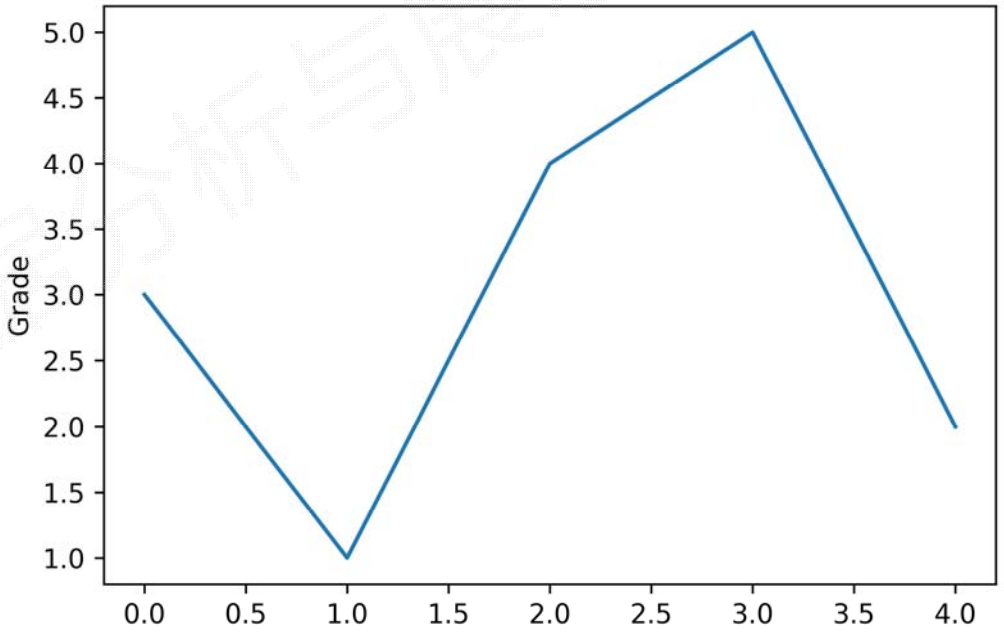


plt.plot()只有一个输入列表或数组时，参数被当作Y轴，X轴以索引自动生成

Matplotlib库小测

```
import matplotlib.pyplot as plt
plt.plot([3, 1, 4, 5, 2])
plt.ylabel("Grade")
plt.savefig('test', dpi=600)  #PNG文件
plt.show()
```

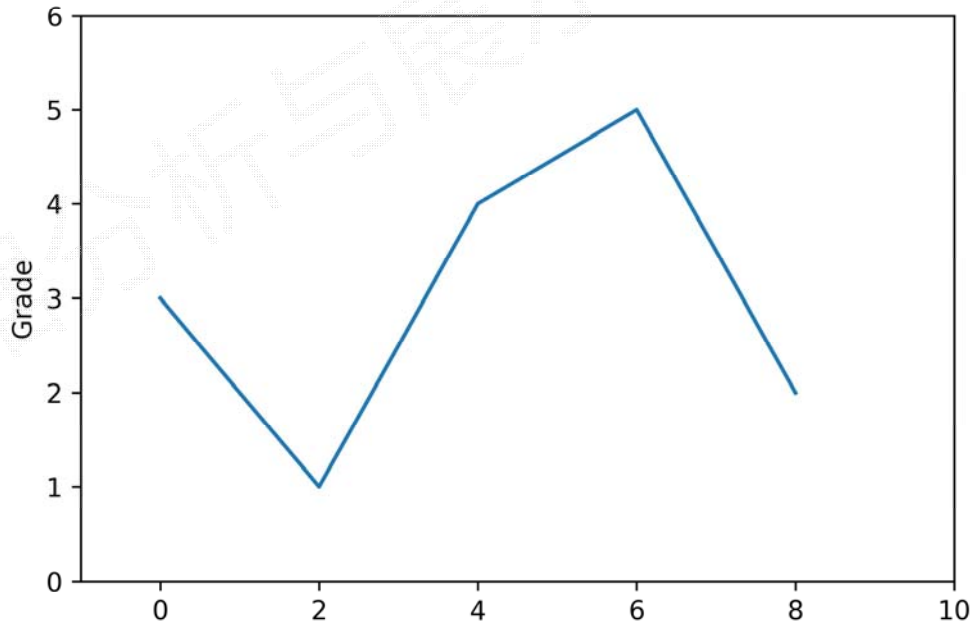
存成文件



plt.savefig()将输出图形存储为文件，默认PNG格式，可以通过dpi修改输出质量

Matplotlib库小测综合

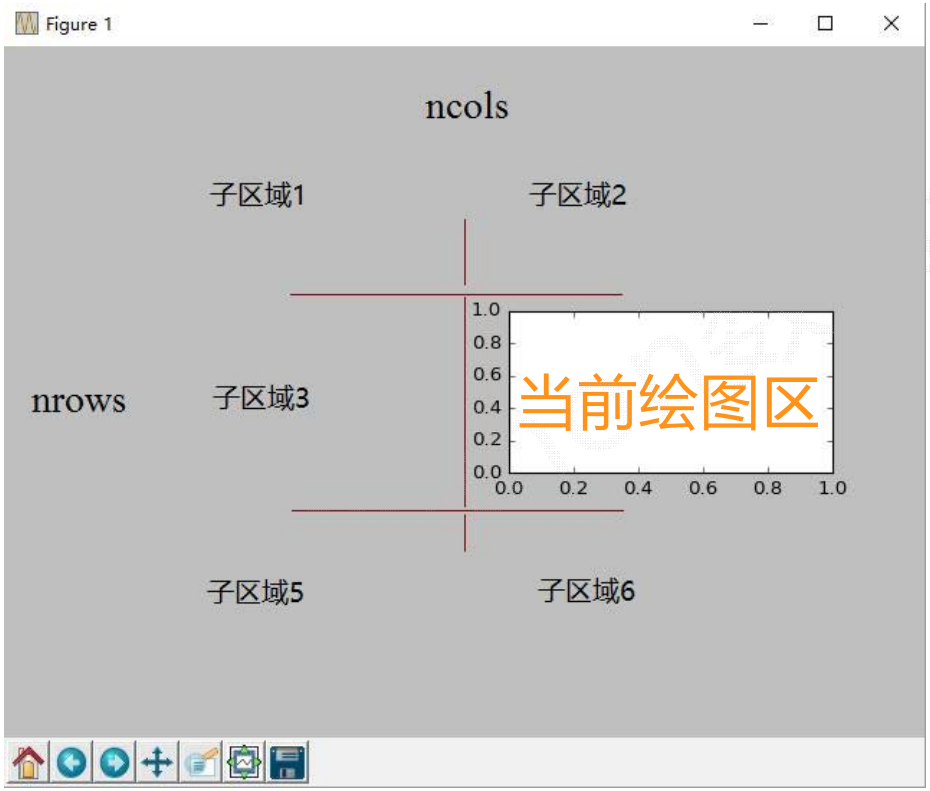
```
import matplotlib.pyplot as plt
plt.plot([0, 2, 4, 6, 8], [3, 1, 4, 5, 2])
plt.ylabel("Grade")
plt.axis([-1, 10, 0, 6])
plt.show()
```



plt.plot(x,y)当有两个以上参数时，按照X轴和Y轴顺序绘制数据点

pyplot的绘图区域

```
plt.subplot(nrows, ncols, plot_number)
```

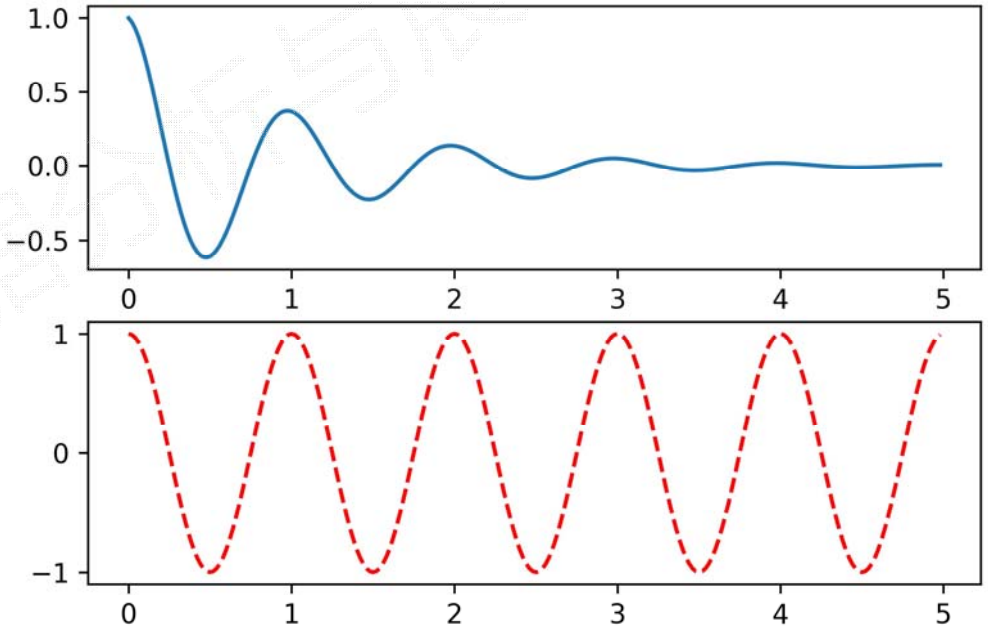


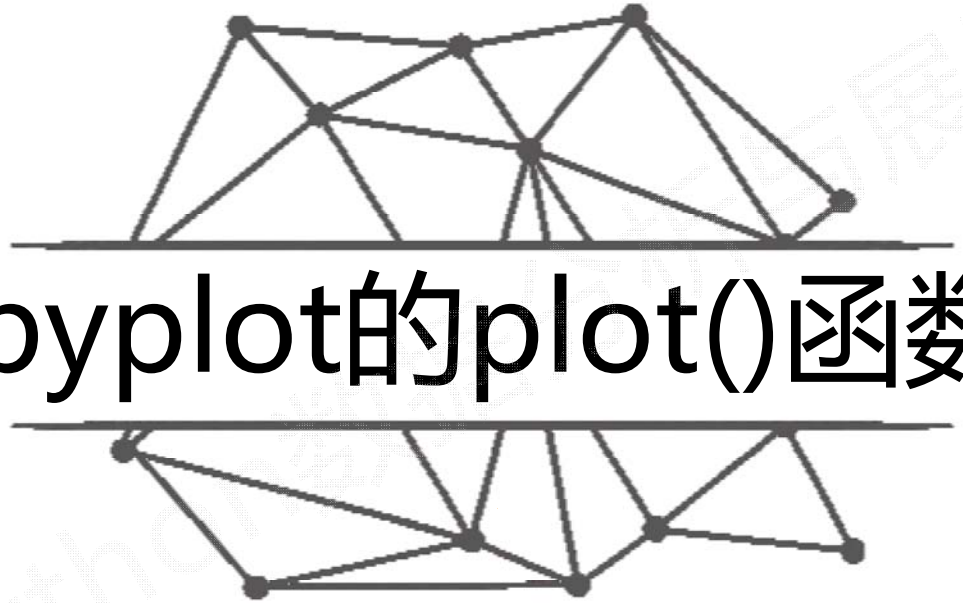

```
plt.subplot(3,2,4)
plt.subplot(324)
```

在全局绘图区域中创建一个分区体系，并定位到一个子绘图区域

pyplot的绘图区域

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

def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)

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

plt.subplot(211)
plt.plot(a, f(a))

plt.subplot(2,1,2)
plt.plot(a, np.cos(2*np.pi*t2), 'r--')
plt.show()
```





pyplot的plot()函数

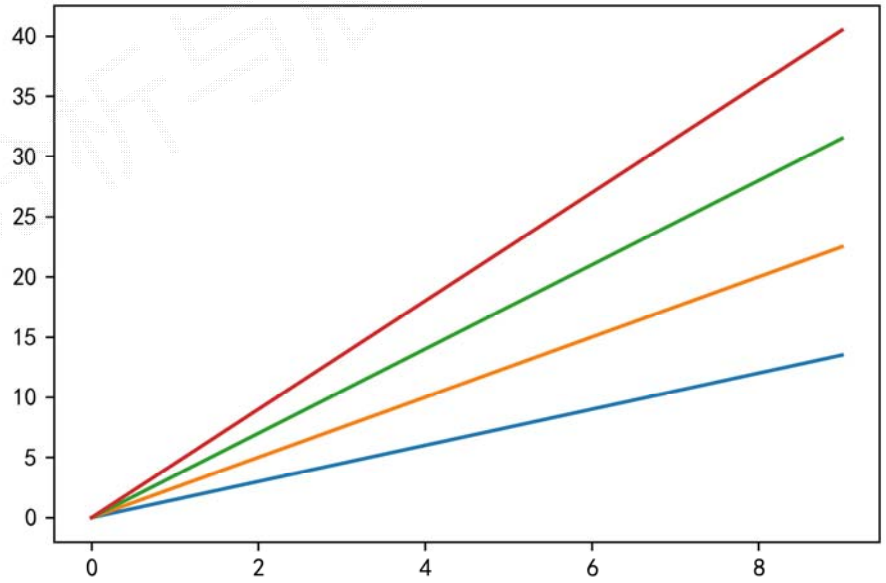
```
plt.plot(x, y, format_string, **kwargs)
```

- x : X轴数据，列表或数组，可选
- y : Y轴数据，列表或数组
- format_string: 控制曲线的格式字符串，可选
- **kwargs : 第二组或更多(x,y,format_string)

当绘制多条曲线时，各条曲线的x不能省略

plt.plot(x, y, format_string, **kwargs)

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

a = np.arange(10)
plt.plot(a, a*1.5, a, a*2.5, a, a*3.5, a, a*4.5)
plt.show()
```

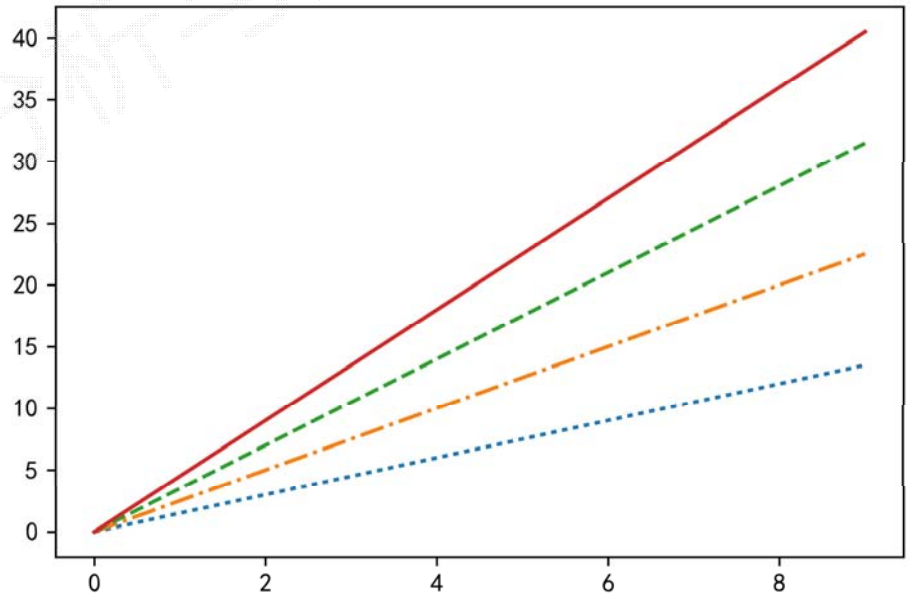


- **format_string**：控制曲线的格式字符串，可选由颜色字符、风格字符和标记字符组成

颜色字符	说明	颜色字符	说明
'b'	蓝色	'm'	洋红色 magenta
'g'	绿色	'y'	黄色
'r'	红色	'k'	黑色
'c'	青绿色 cyan	'w'	白色
'#008000'	RGB某颜色	'0.8'	灰度值字符串

- `format_string`：控制曲线的格式字符串，可选由颜色字符、风格字符和标记字符组成

风格字符	说明
'-'	实线
'--'	破折线
'-.'	点划线
':'	虚线
'' ' '	无线条

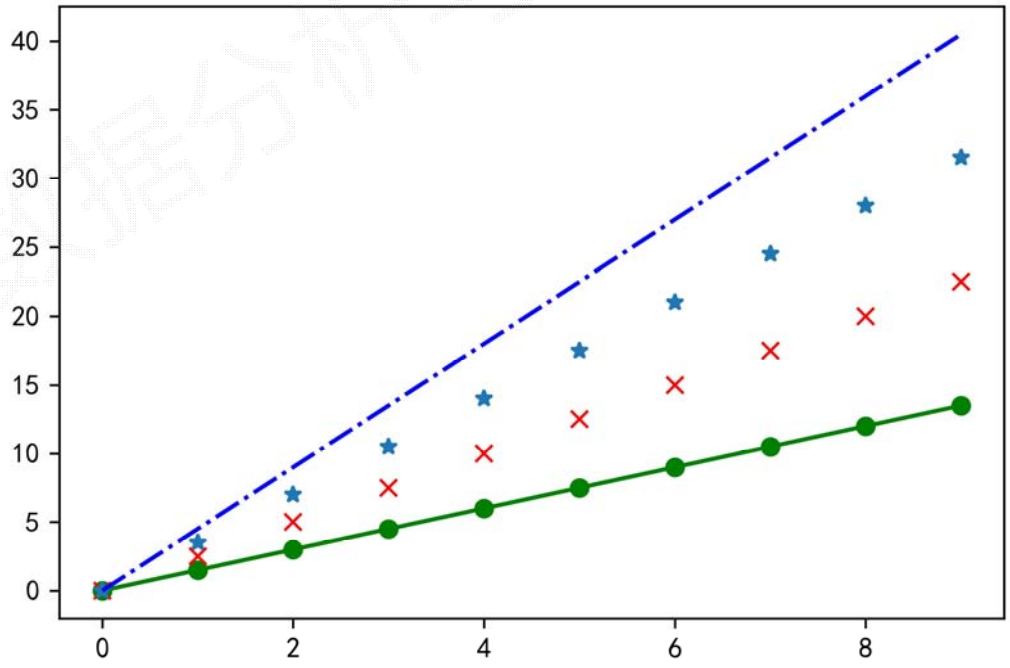


- `format_string`: 控制曲线的格式字符串，可选

标记字符	说明	标记字符	说明	标记字符	说明
'.'	点标记	'1'	下花三角标记	'h'	竖六边形标记
','	像素标记(极小点)	'2'	上花三角标记	'H'	横六边形标记
'o'	实心圈标记	'3'	左花三角标记	'+'	十字标记
'v'	倒三角标记	'4'	右花三角标记	'x'	x标记
'^'	上三角标记	's'	实心方形标记	'D'	菱形标记
'>'	右三角标记	'p'	实心五角标记	'd'	瘦菱形标记
'<'	左三角标记	'*'	星形标记	'|'	垂直线标记

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

a = np.arange(10)
plt.plot(a, a*1.5, 'go-', a, a*2.5, 'rx', a, a*3.5, '*', a, a*4.5, 'b-.')
plt.show()
```

颜色字符、风格字符和标记字符可以组合使用



```
plt.plot(x, y, format_string, **kwargs)
```

- **kwargs : 第二组或更多(x,y,format_string)

  color : 控制颜色，color='green'

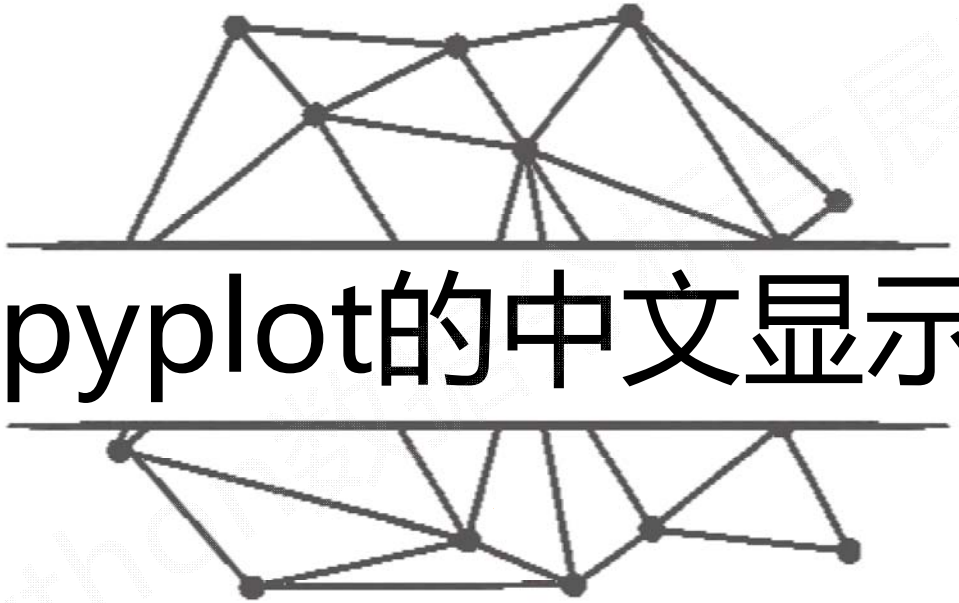
  linestyle : 线条风格，linestyle='dashed'

  marker : 标记风格，marker='o'

  markerfacecolor: 标记颜色，markerfacecolor='blue'

  markersize : 标记尺寸，markersize=20

  ……

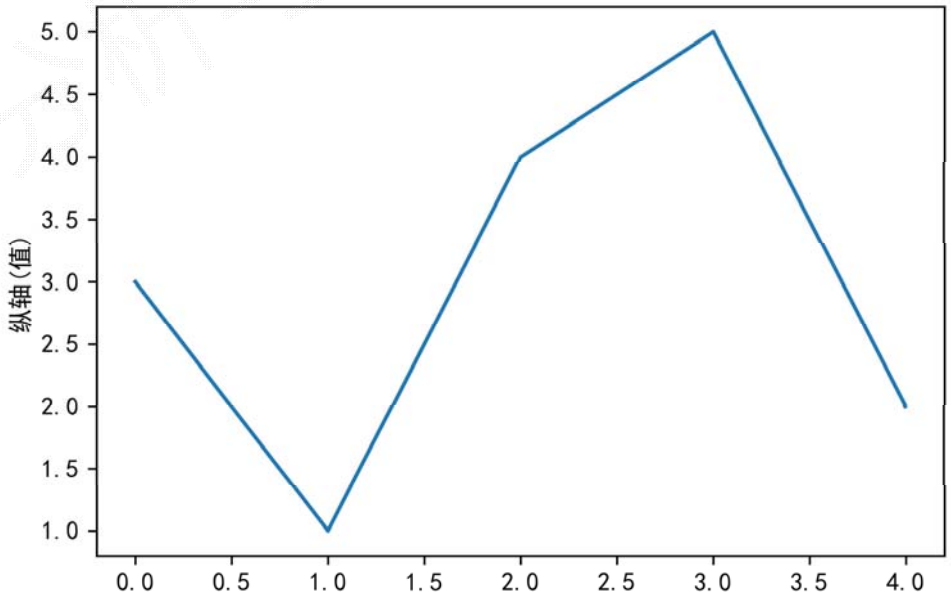


pyplot的中文显示

pyplot的中文显示：第一种方法

pyplot并不默认支持中文显示，需要rcParams修改字体实现

```
import matplotlib.pyplot as plt
import matplotlib

matplotlib.rcParams['font.family']='SimHei'
plt.plot([3, 1, 4, 5, 2])
plt.ylabel("纵轴(值)")
plt.savefig('test', dpi=600)
plt.show()
```



'SimHei'是黑体

rcParams的属性

属性	说明
'font.family'	用于显示字体的名字
'font.style'	字体风格，正常'normal'或 斜体'italic'
'font.size'	字体大小，整数字号或者'large'、'x-small'

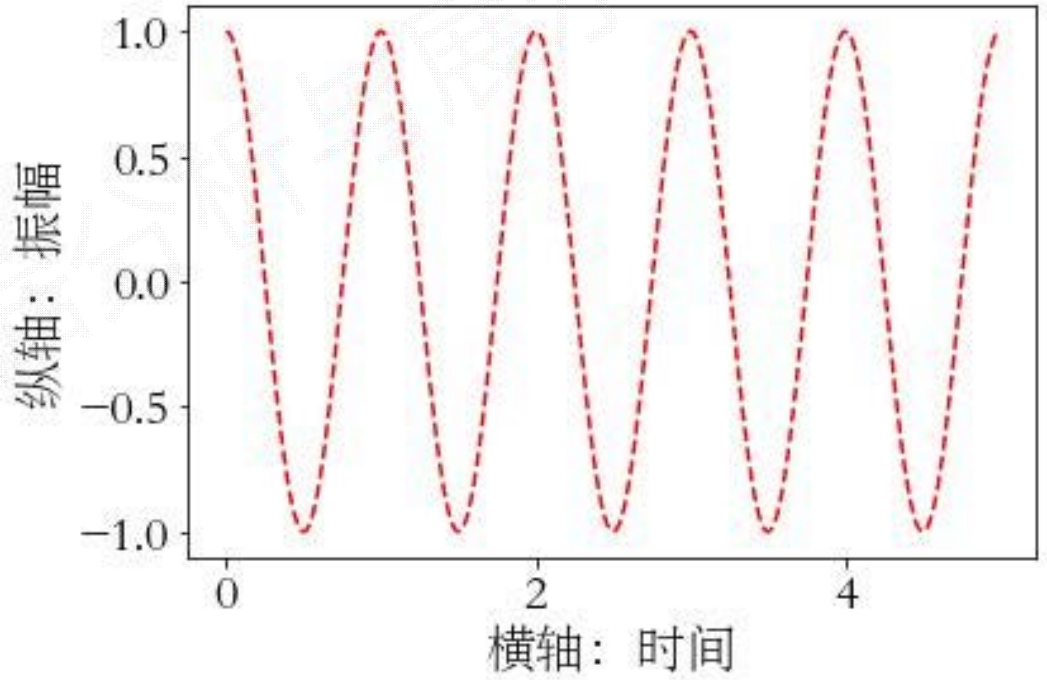
中文字体的种类

```
rcParams['font.family']
```

中文字体	说明
'SimHei'	中文黑体
'Kaiti'	中文楷体
'LiSu'	中文隶书
'FangSong'	中文仿宋
'YouYuan'	中文幼圆
'STSong'	华文宋体

实例

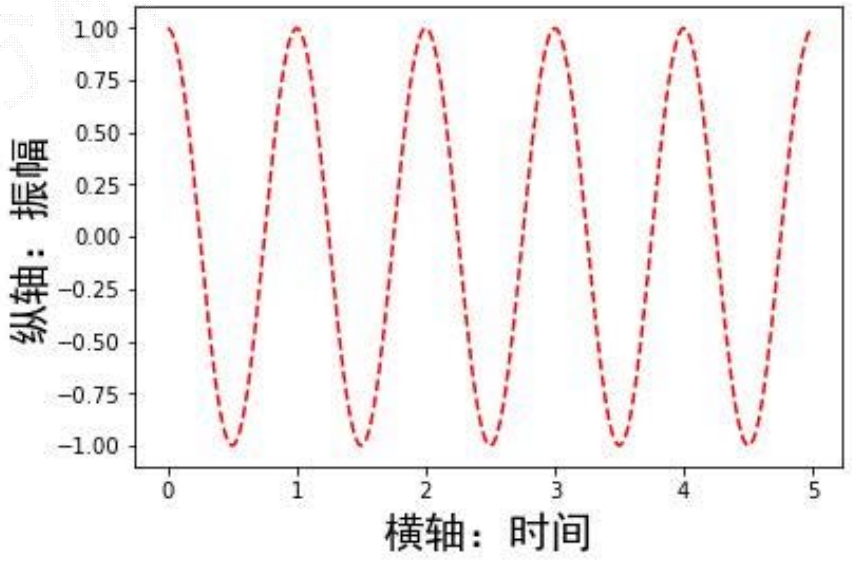
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib

matplotlib.rcParams['font.family']='STSong'
matplotlib.rcParams['font.size']=20

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

plt.xlabel('横轴：时间')
plt.ylabel('纵轴：振幅')
plt.plot(a, np.cos(2*np.pi*a), 'r--')
plt.show()
```

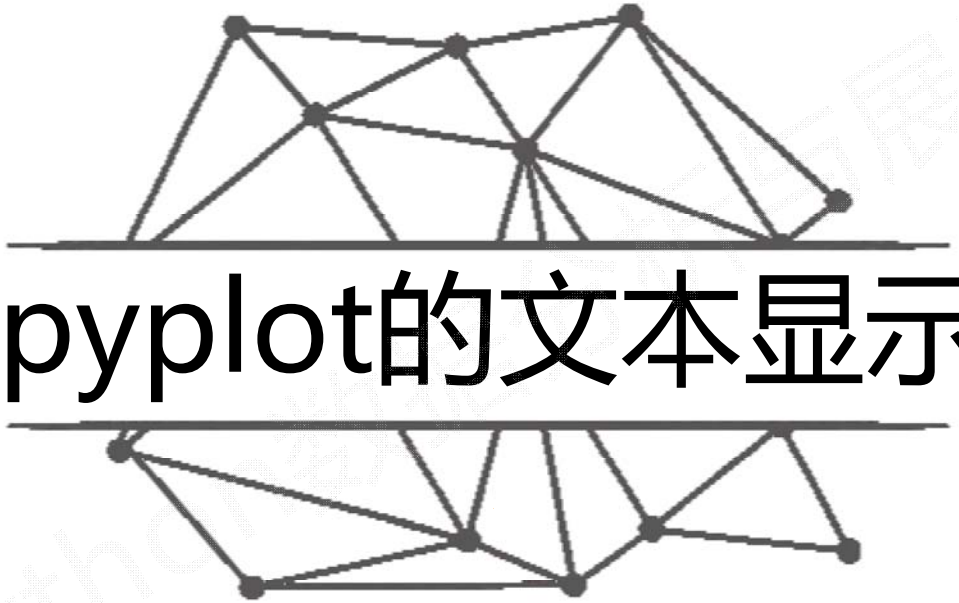


pyplot的中文显示：第二种方法

在有中文输出的地方，增加一个属性：fontproperties

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

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

plt.xlabel('横轴：时间', fontproperties='SimHei', fontsize=20)
plt.ylabel('纵轴：振幅', fontproperties='SimHei', fontsize=20)
plt.plot(a, np.cos(2*np.pi*a), 'r--')
plt.show()
```



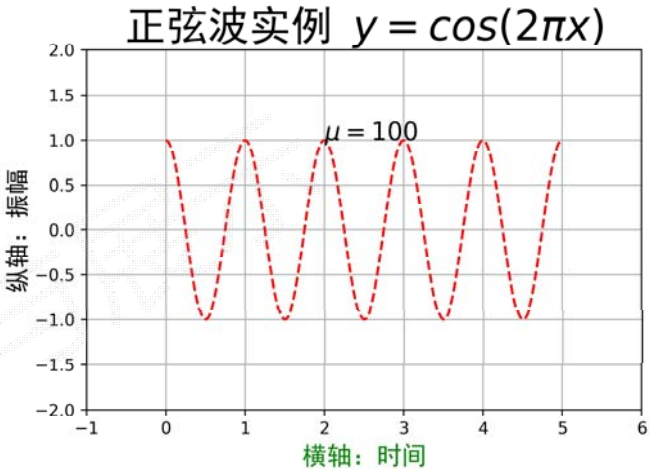


pyplot的文本显示

pyplot的文本显示函数

函数	说明
plt.xlabel()	对X轴增加文本标签
plt.ylabel()	对Y轴增加文本标签
plt.title()	对图形整体增加文本标签
plt.text()	在任意位置增加文本
plt.annotate()	在图形中增加带箭头的注解

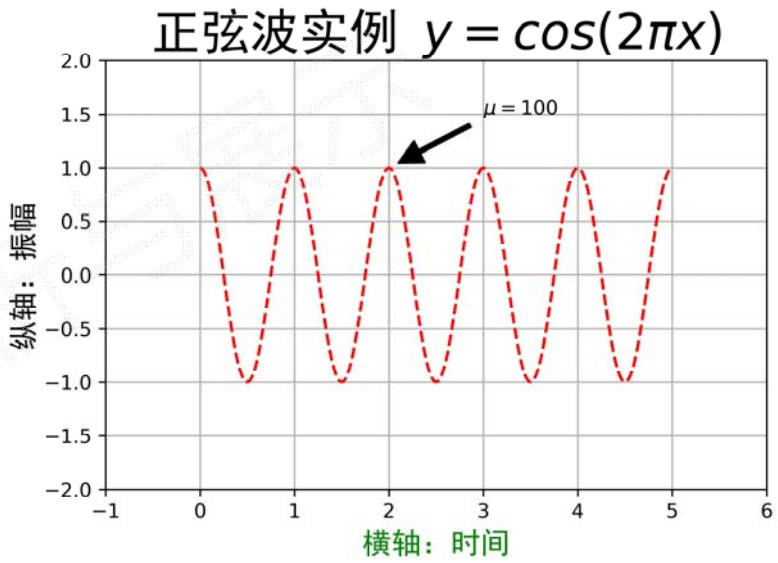
实例



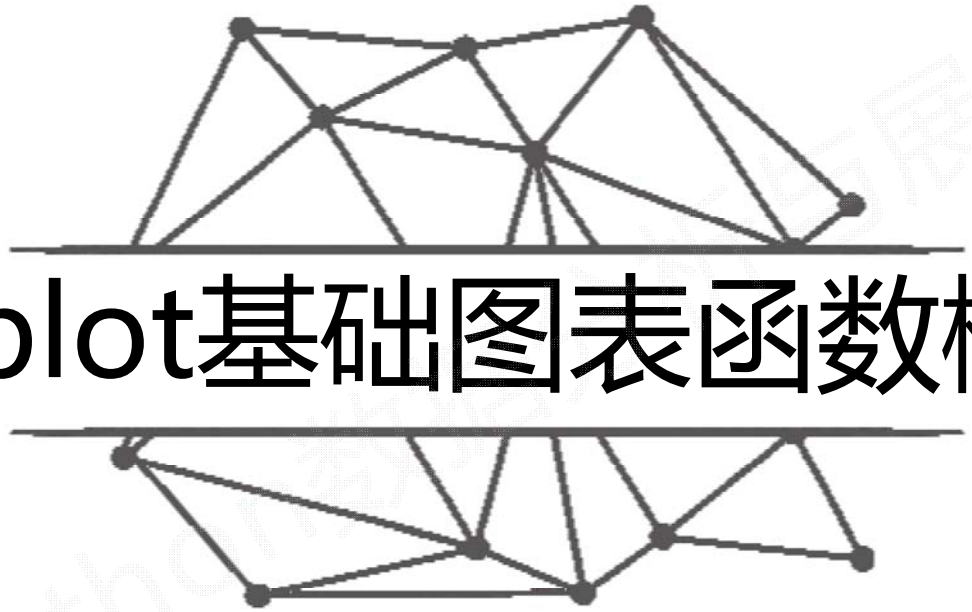

```
import numpy as np
import matplotlib.pyplot as plt

a = np.arange(0.0, 5.0, 0.02)
plt.plot(a, np.cos(2*np.pi*a), 'r--')

plt.xlabel('横轴：时间', fontproperties='SimHei', fontsize=15, color='green')
plt.ylabel('纵轴：振幅', fontproperties='SimHei', fontsize=15)
plt.title(r'正弦波实例 $y=cos(2\pi x)$', fontproperties='SimHei', fontsize=25)
plt.text(2, 1, r'$\mu=100$', fontsize=15)

plt.axis([-1, 6, -2, 2])
plt.grid(True)
plt.show()
```

Latex

```
plt.annotate(s, xy=arrow_crd, xytext=text_crd, arrowprops=dict)
```




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

a = np.arange(0.0, 5.0, 0.02)
plt.plot(a, np.cos(2*np.pi*a), 'r--')

plt.xlabel('横轴：时间', fontproperties='SimHei', fontsize=25, color='green')
plt.ylabel('纵轴：振幅', fontproperties='SimHei', fontsize=25)
plt.title(r'正弦波实例 $y=cos(2\pi x)$', fontproperties='SimHei', fontsize=25)
plt.annotate(r'$\mu=100$', xy=(2, 1), xytext=(3, 1.5),
             arrowprops=dict(facecolor='black', shrink=0.1, width=2))

plt.axis([-1, 6, -2, 2])
plt.grid(True)
plt.show()
```



pyplot基础图表函数概述

pyplot的基础图标函数

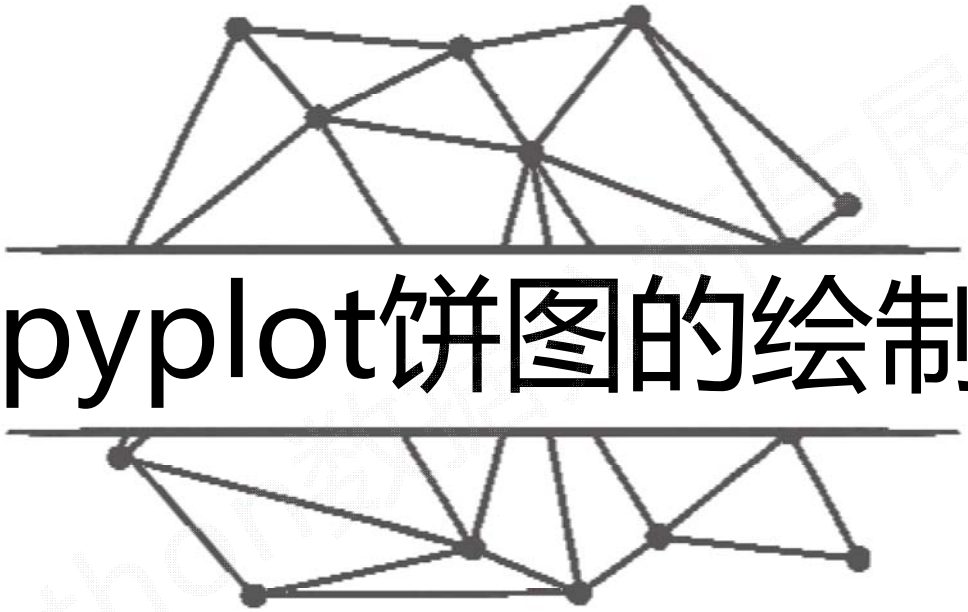
函数	说明
plt.plot(x,y,fmt,…)	绘制一个坐标图
plt.boxplot(data,notch,position)	绘制一个箱形图
plt.bar(left,height,width,bottom)	绘制一个条形图
plt.barh(width,bottom,left,height)	绘制一个横向条形图
plt.polar(theta, r)	绘制极坐标图
plt.pie(data, explode)	绘制饼图

pyplot的基础图标函数

函数	说明
plt.psd(x,NFFT=256,pad_to,Fs)	绘制功率谱密度图
plt.specgram(x,NFFT=256,pad_to,F)	绘制谱图
plt.cohere(x,y,NFFT=256,Fs)	绘制X-Y的相关性函数
plt.scatter(x,y)	绘制散点图，其中，x和y长度相同
plt.step(x,y,where)	绘制步阶图
plt.hist(x,bins,normed)	绘制直方图

pyplot的基础图标函数

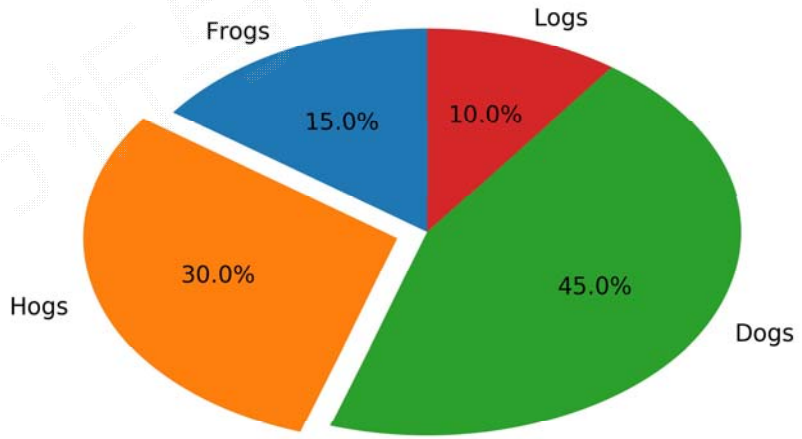
函数	说明
plt.contour(X,Y,Z,N)	绘制等值图
plt.vlines()	绘制垂直图
plt.stem(x,y,linefmt,markerfmt)	绘制柴火图
plt.plot_date()	绘制数据日期



pyplot饼图的绘制

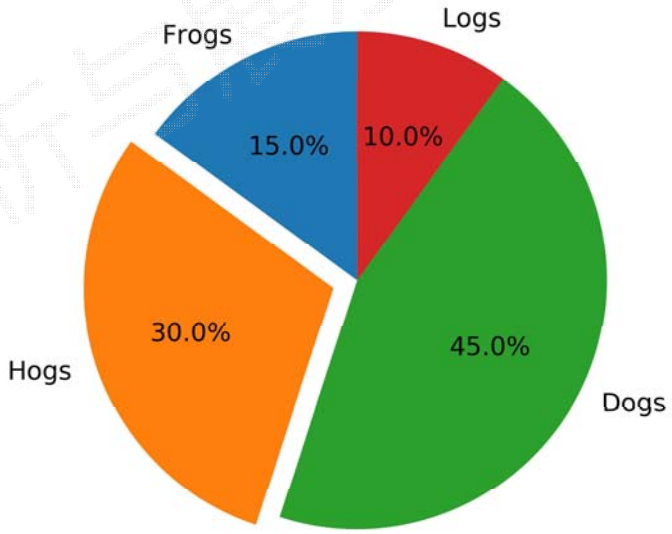
plt.pie()

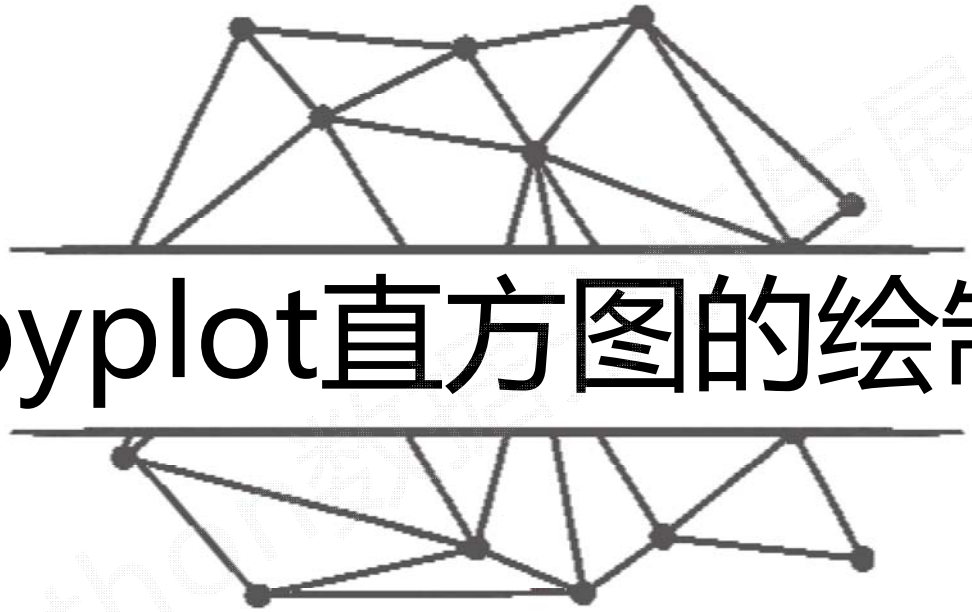
```
import matplotlib.pyplot as plt

labels  = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes   = [15, 30, 45, 10]
explode = (0, 0.1, 0, 0)

plt.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=False, startangle=90)

plt.show()
```



plt.pie()

```
import matplotlib.pyplot as plt

labels  = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes   = [15, 30, 45, 10]
explode = (0, 0.1, 0, 0)

plt.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=False, startangle=90)

plt.axis('equal')
plt.show()
```

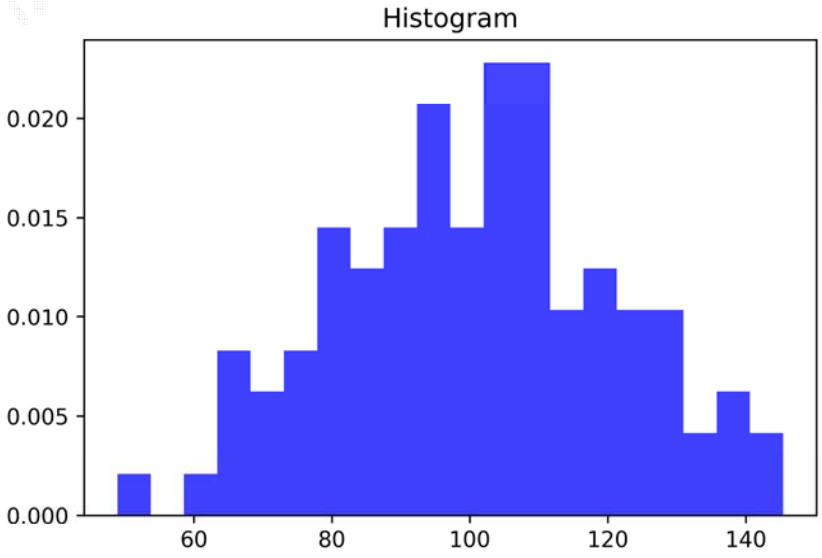




pyplot直方图的绘制

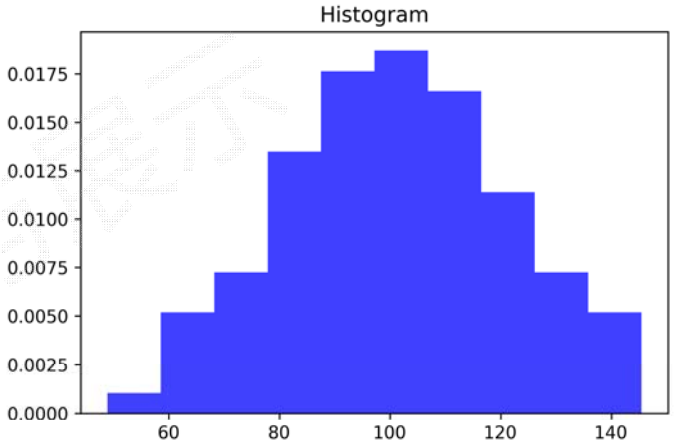
plt.hist()

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

np.random.seed(0)
mu, sigma = 100, 20    # 均值和标准差
a = np.random.normal(mu, sigma, size=100)

plt.hist(a, 20, normed=1, histtype='stepfilled',  facecolor='b', alpha=0.75)
plt.title('Histogram')

plt.show()
```



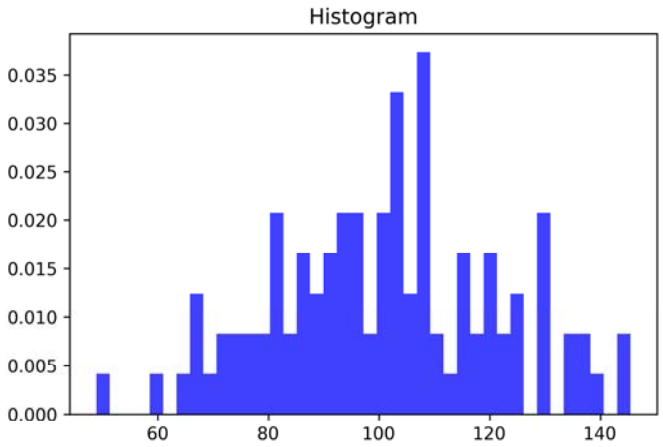
plt.hist()

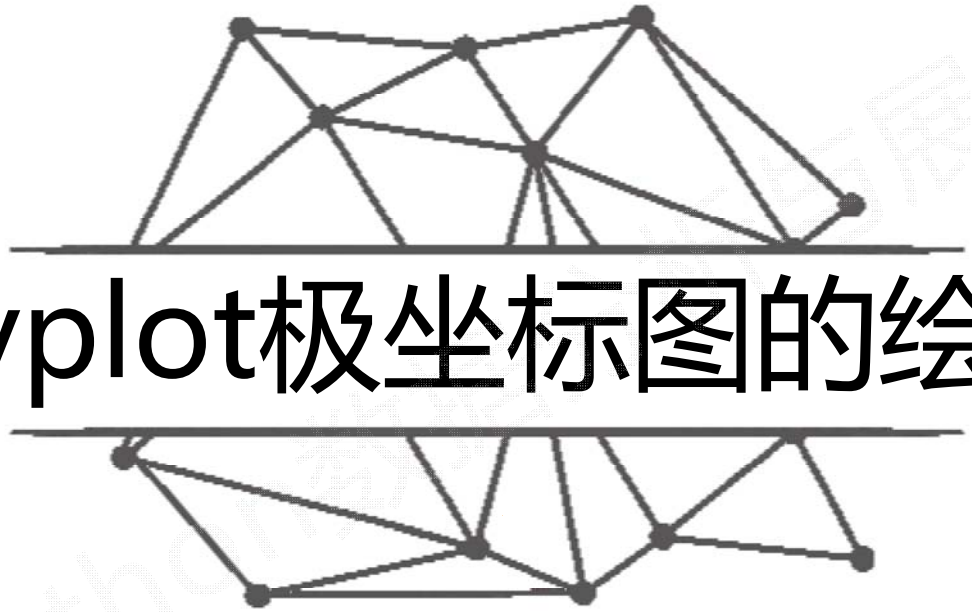
```python
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(0)
mu, sigma = 100, 20    # 均值和标准差
a = np.random.normal(mu, sigma, size=100)

plt.hist(a, 10, normed=1, histtype='stepfilled',  facecolor='b', alpha=0.75)
plt.title('Histogram')

plt.show()
```



bin: 直方图的个数

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

np.random.seed(0)
mu, sigma = 100, 20    # 均值和标准差
a = np.random.normal(mu, sigma, size=100)

plt.hist(a, 40, normed=1, histtype='stepfilled',  facecolor='b', alpha=0.75)
plt.title('Histogram')

plt.show()
```





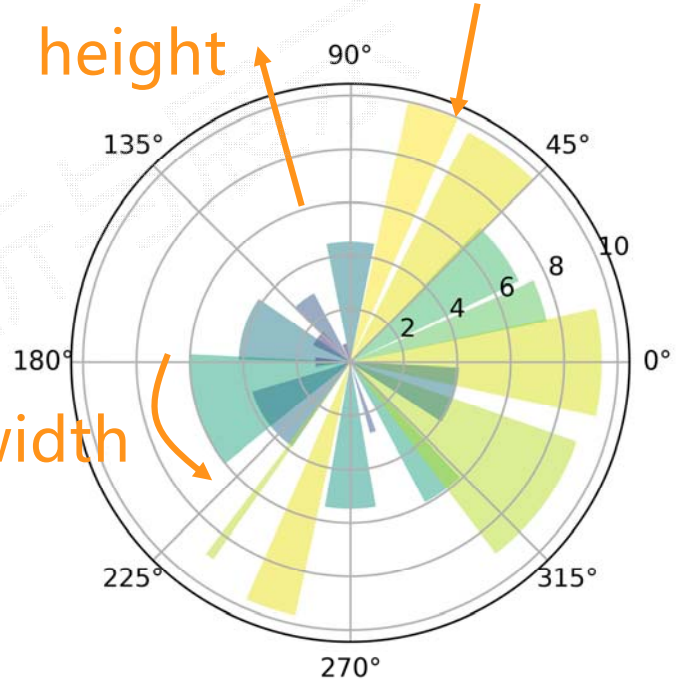
pyplot极坐标图的绘制

面向对象绘制极坐标

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

N = 20
theta = np.linspace(0.0, 2 * np.pi, N, endpoint=False)
radii = 10 * np.random.rand(N)
width = np.pi / 4 * np.random.rand(N)

ax = plt.subplot(111, projection='polar')
bars = ax.bar(theta, radii, width=width, bottom=0.0)

for r, bar in zip(radii, bars):
    bar.set_facecolor(plt.cm.viridis(r / 10.))
    bar.set_alpha(0.5)

plt.show()
```

left

height

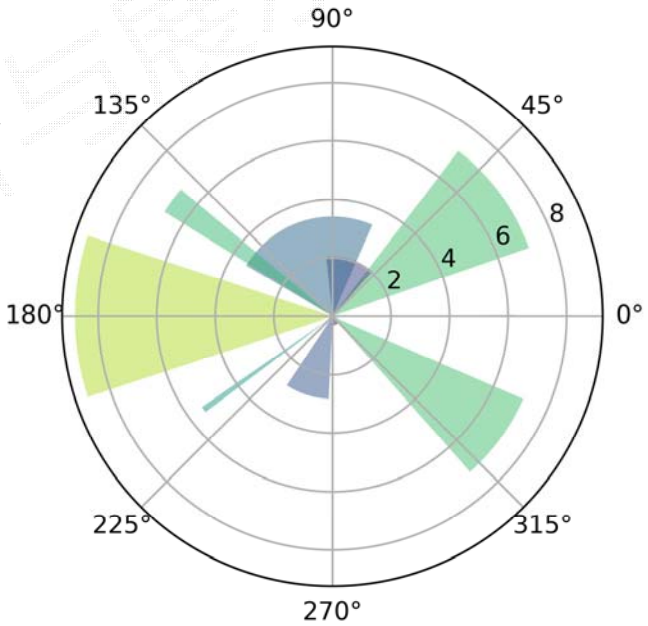


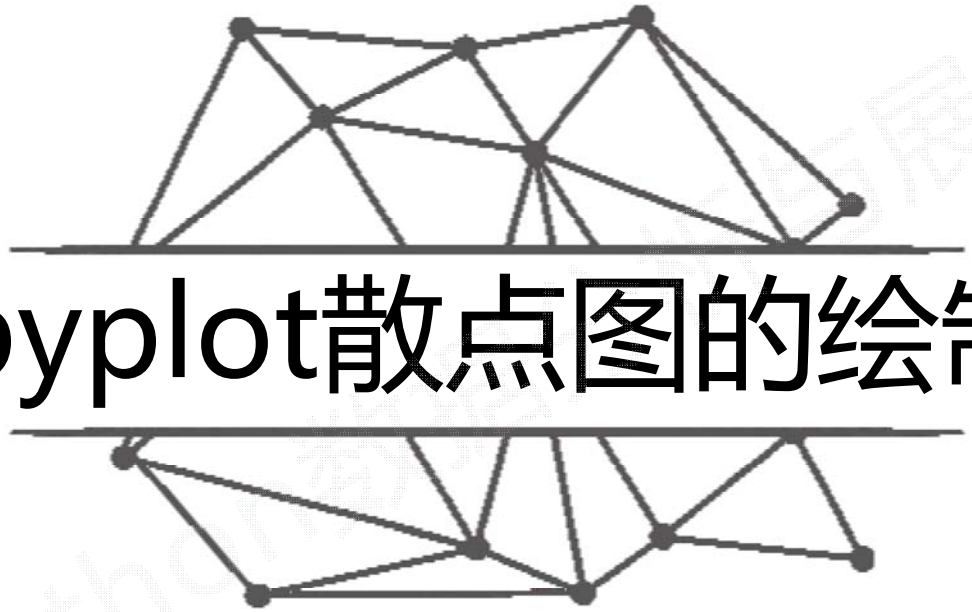
width

left, height, width

面向对象绘制方式

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

N = 10
theta = np.linspace(0.0, 2 * np.pi, N, endpoint=False)
radii = 10 * np.random.rand(N)
width = np.pi / 2 * np.random.rand(N)

ax = plt.subplot(111, projection='polar')
bars = ax.bar(theta, radii, width=width, bottom=0.0)

for r, bar in zip(radii, bars):
    bar.set_facecolor(plt.cm.viridis(r / 10.))
    bar.set_alpha(0.5)

plt.show()
```

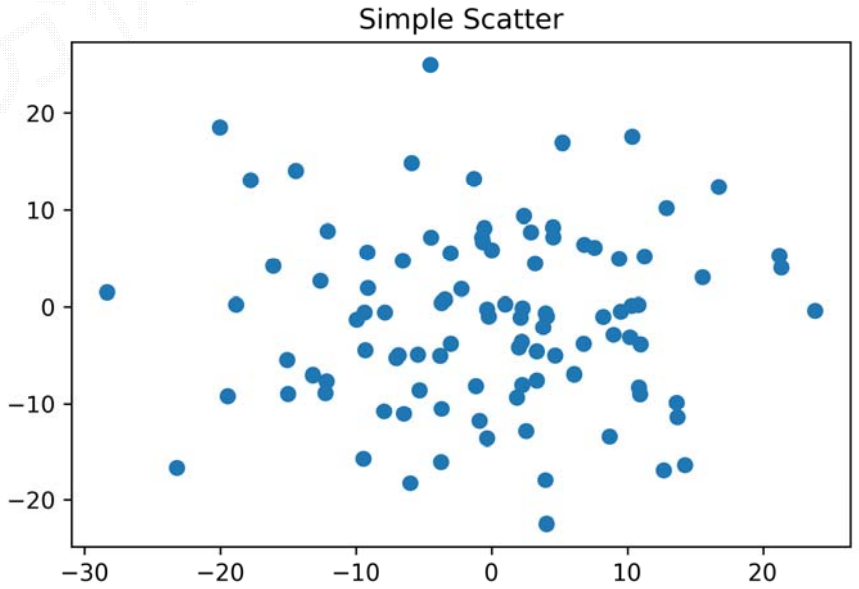


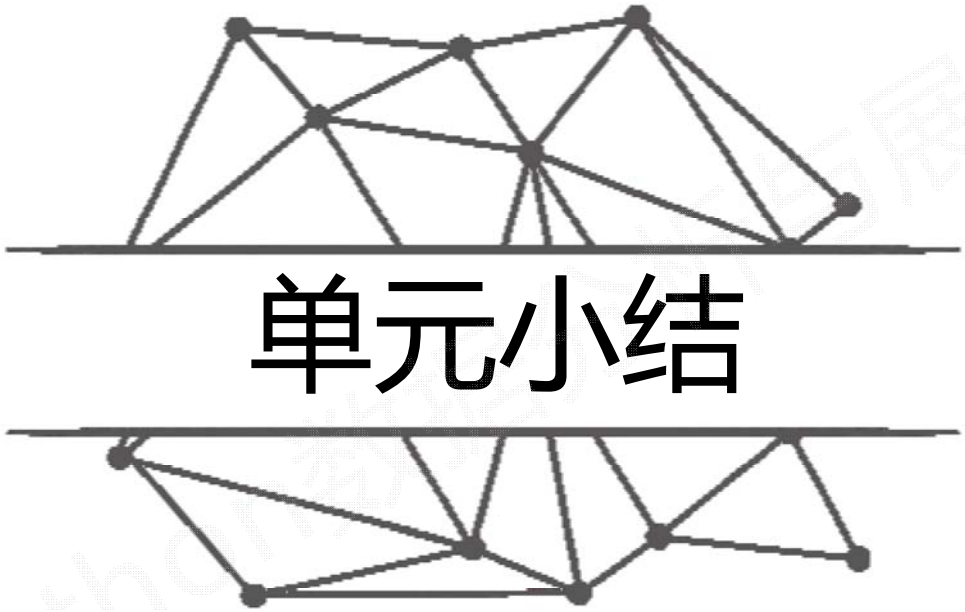


pyplot散点图的绘制

面向对象绘制散点图

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

fig, ax = plt.subplots()
ax.plot(10*np.random.randn(100), 10*np.random.randn(100), 'o')
ax.set_title('Simple Scatter')

plt.show()
```





单元小结

Matplotlib基础绘图函数示例