Matplotlib数据可视化

主讲：万永权

主要内容

- 6.1 Matplotlib 基本介绍
- 6.2 Matplotlib绘图基础
- 6.3 rc参数设置
- 6.4 pyplot中的常用绘图

参考资源

官方API文档

- https://www.matplotlib.org.cn/
- https://matplotlib.org/stable/api/index

Matplotlib数据可视化基础

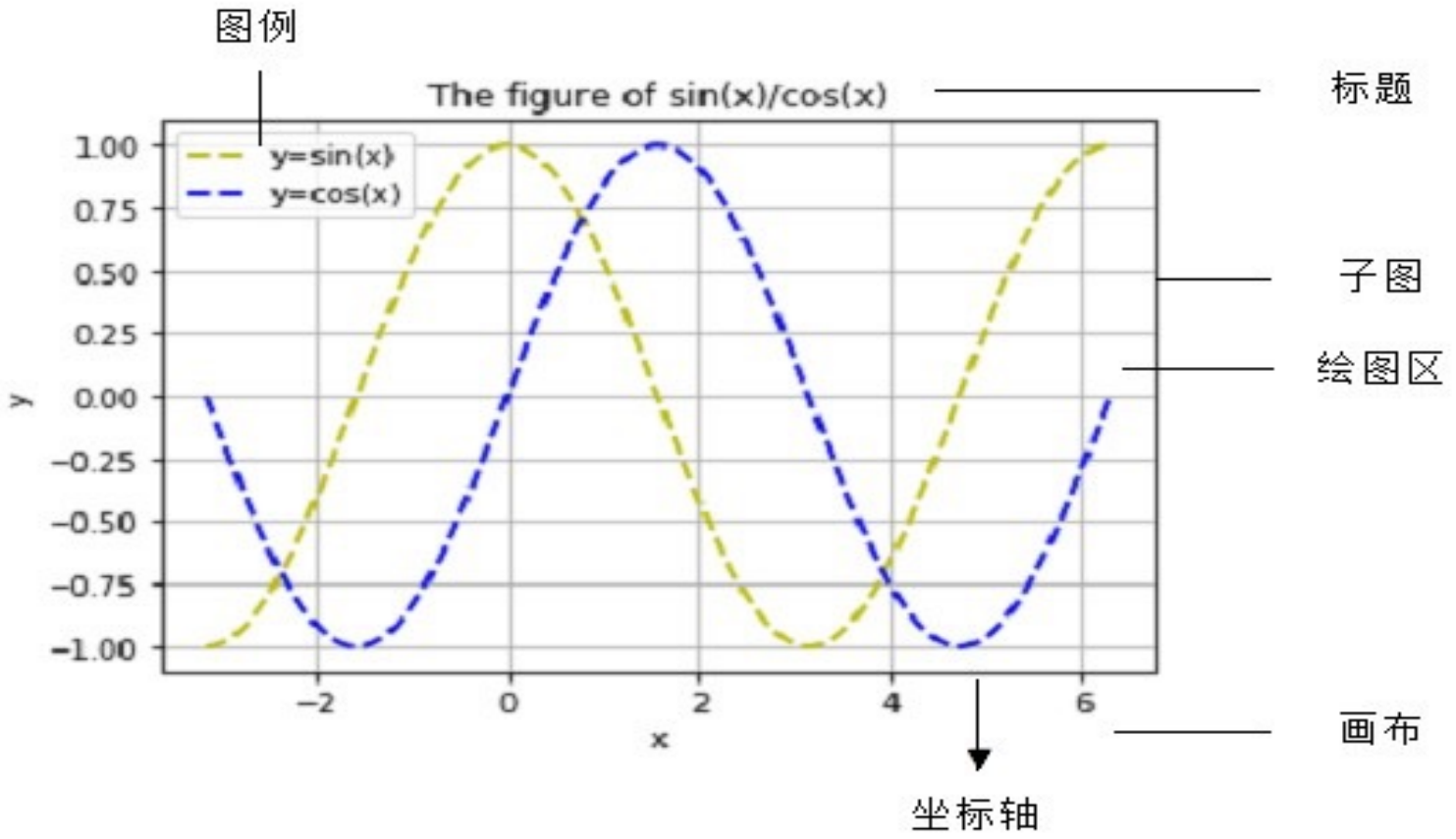
- Matplotlib是Python的一个基本2D绘图库，它提供了很多参数，可以通过参数控制样式、属性等，生成跨平台的出版质量级别的图形。
- 使用Matplotlib，能让复杂的工作变得容易，可以生成直方图、条形图、散点图、曲线图等。
- Matplotlib可用于Python scripts、Python、IPython、Jupyter notebook、web应用服务器等。

Matplotlib绘图基础

图6-1 Matplotlib中的常用绘图及分组

1. 图表的基本结构

- 图表的结构一般包括：画布、图表标题、绘图区、x轴（水平轴）和y轴（垂直轴）、图例等基本元素。



2. matplotlib.pyplot

Matplotlib模块中比较常用的是pyplot子模块，内部包含了绘制图形所需要的功能函数。

pyplot模块的常用函数

函数	描述
figure	创建一个空白画布，可以指定画布的大小和像素
add_subplot	创建子图，可以指定子图的行数，列数和标号
subplots	建立一系列子图，返回fig,ax一个fig序列对象，建立一个axis序列
title	设置图表标题，可以指定标题的名称、颜色、字体等参数
xlabel	设置x轴名称，可以指定名称、颜色、字体等参数
ylabel	设置y轴名称，可以指定名称、颜色、字体等参数
xlim	指定x轴的刻度范围
ylim	指定y轴的刻度范围
legend	指定图例，及图例的大小、位置、标签
savefig	保存图形
show	显示图形

Matplotlib的图像都位于figure对象中，用plt.figure创建一个新的画布（空画布不能直接绘图）。如果不显式调用figure()函数，也会默认创建一个画布供子图使用。

在画布上添加plot子图用add_subplot方法，add_subplot 函数的使用方法如下：

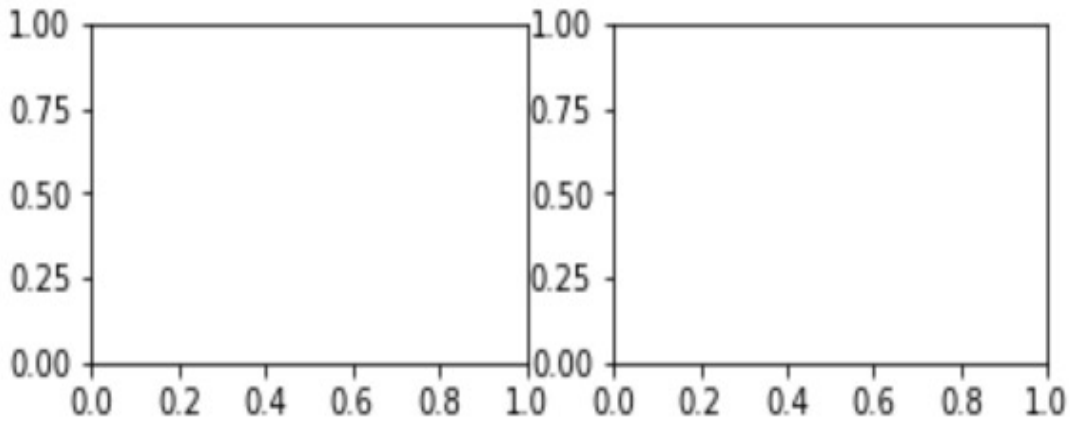
<子图对象>=<figure对象>.add_subplot(nrows, ncols, index)

参数含义：

- nrows：子图划分成的行数
- ncols：子图划分成的列数
- index：当前子图的序号，编号从1开始

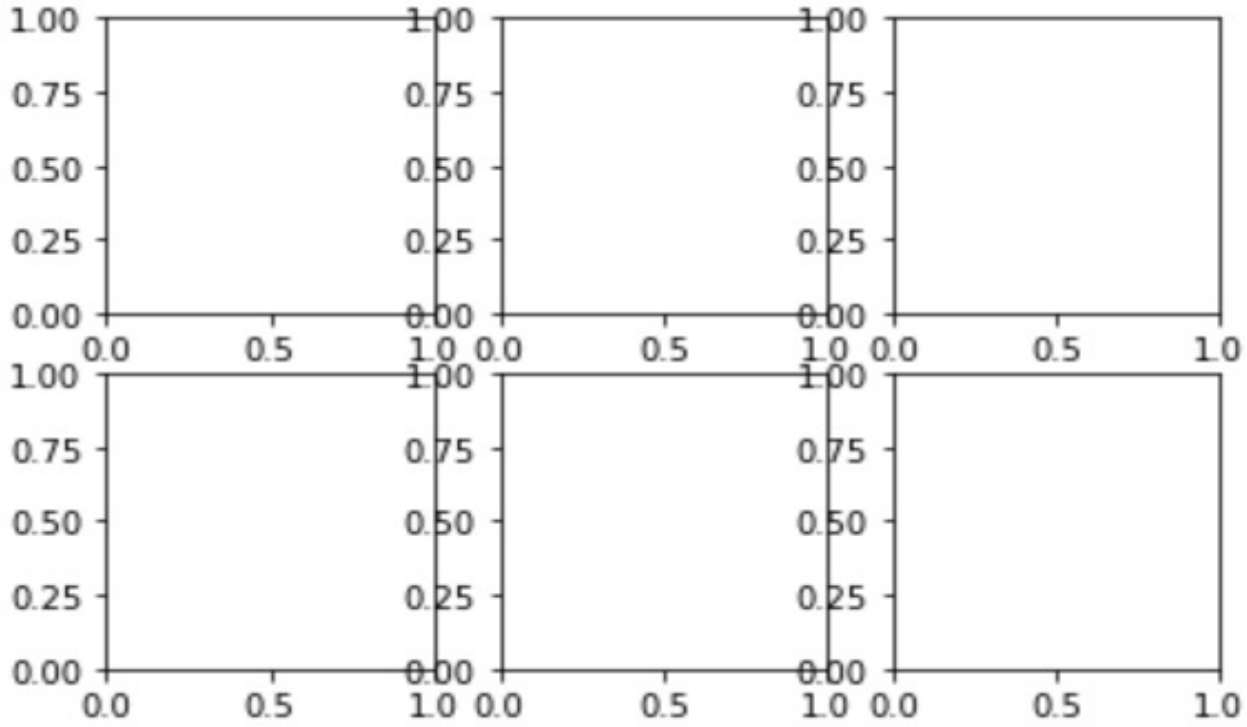
【例3.56】绘制简单的plot图表，结果如图3.3所示。

```
import matplotlib.pyplot as plt
fig=plt.figure()
ax1=fig.add_subplot(2,2,1)
ax2=fig.add_subplot(2,2,2)   #这里修改成(2,2,4)试试
```



【例3.58】六个plot的绘制，结果如图3.3所示。

```
fig,axes=plt.subplots(2,3)
axes
```



3. plot函数

绘制曲线可以使用pyplot中的plot 函数。

plot()的基本格式如下：

matplotlib.pyplot.plot(x,y,format_string,**kwargs)

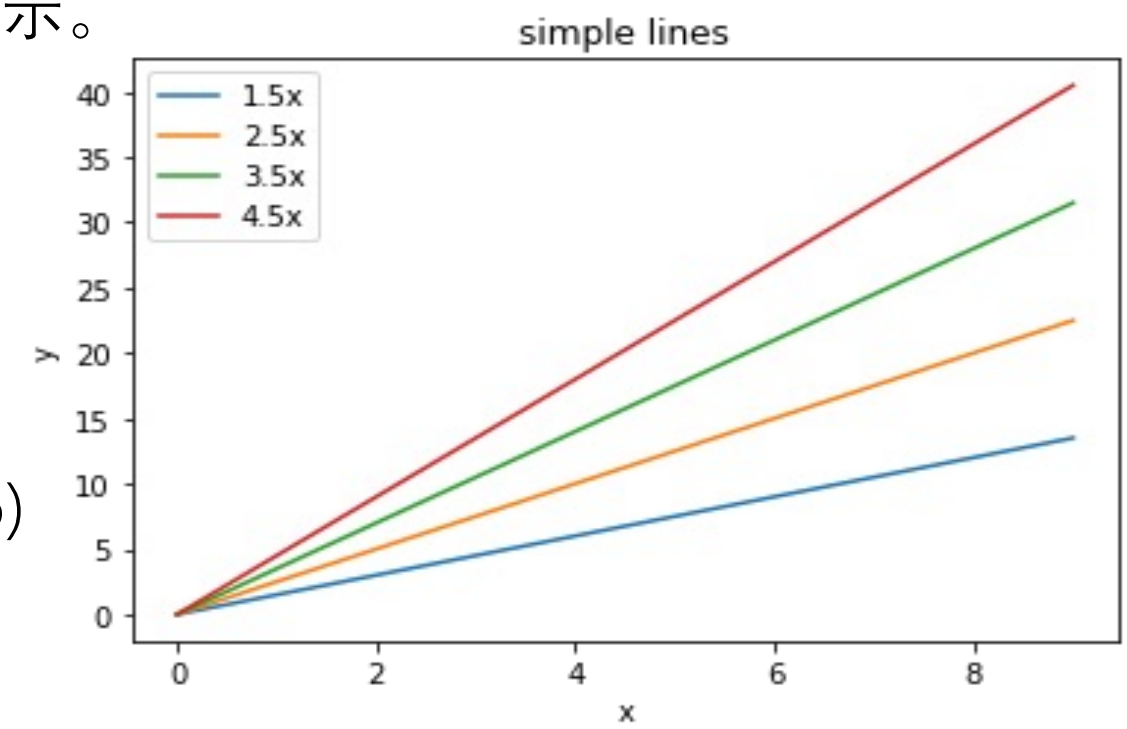
参数：

- 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：第二组或更多组(x,y,format_string)参数。

注：当绘制多条曲线时，各条曲线的x不能省略。

【例3.60】绘制简单直线，结果如图3.7所示。

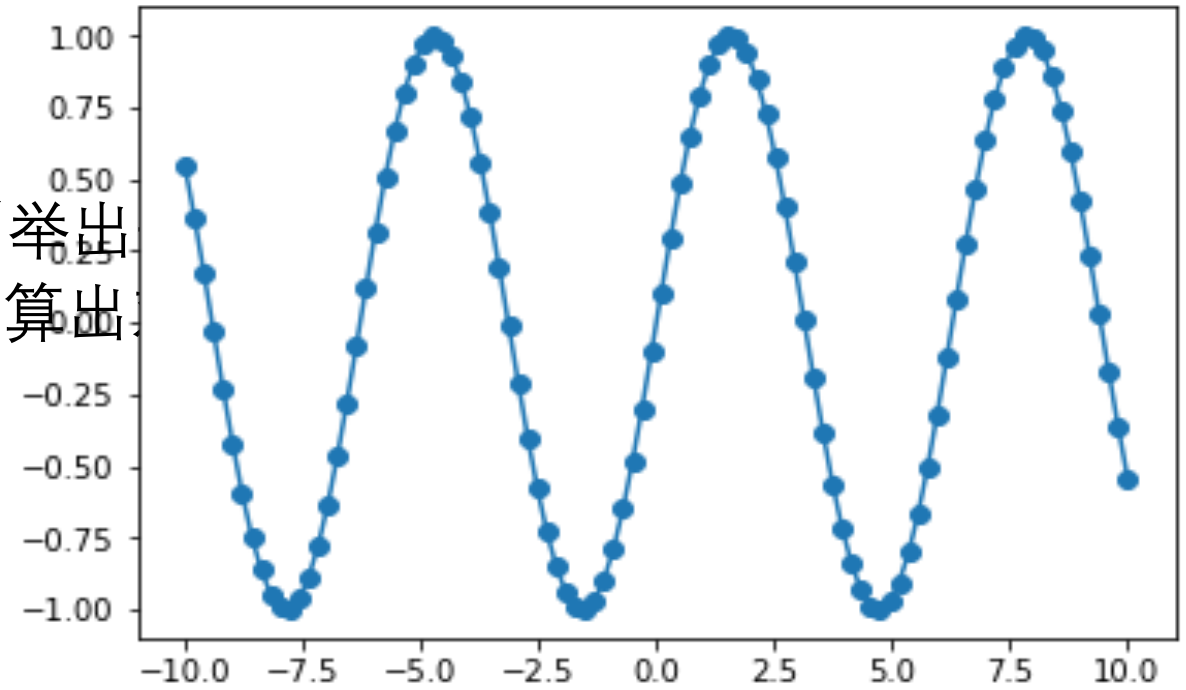
```
import matplotlib.pyplot as plt
import numpy as np
a = np.arange(10)
plt.xlabel('x')
plt.ylabel('y')
plt.plot(a,a*1.5,a,a*2.5,a,a*3.5,a,a*4.5)
plt.legend(['1.5x','2.5x','3.5x','4.5x'])
plt.title('simple lines')
plt.show()
```



对于数学函数来说，绘制图形通常采用多数据点拟合的方式。例如可以罗列出一定数量的x值，再通过函数求出对应的y值，从而构成一列x、y数据对。

【例3.61】绘制sin(x)函数图形。

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-10, 10, 100)  #列举出
y = np.sin(x)                  #计算出
plt.plot(x, y, marker="o")
```

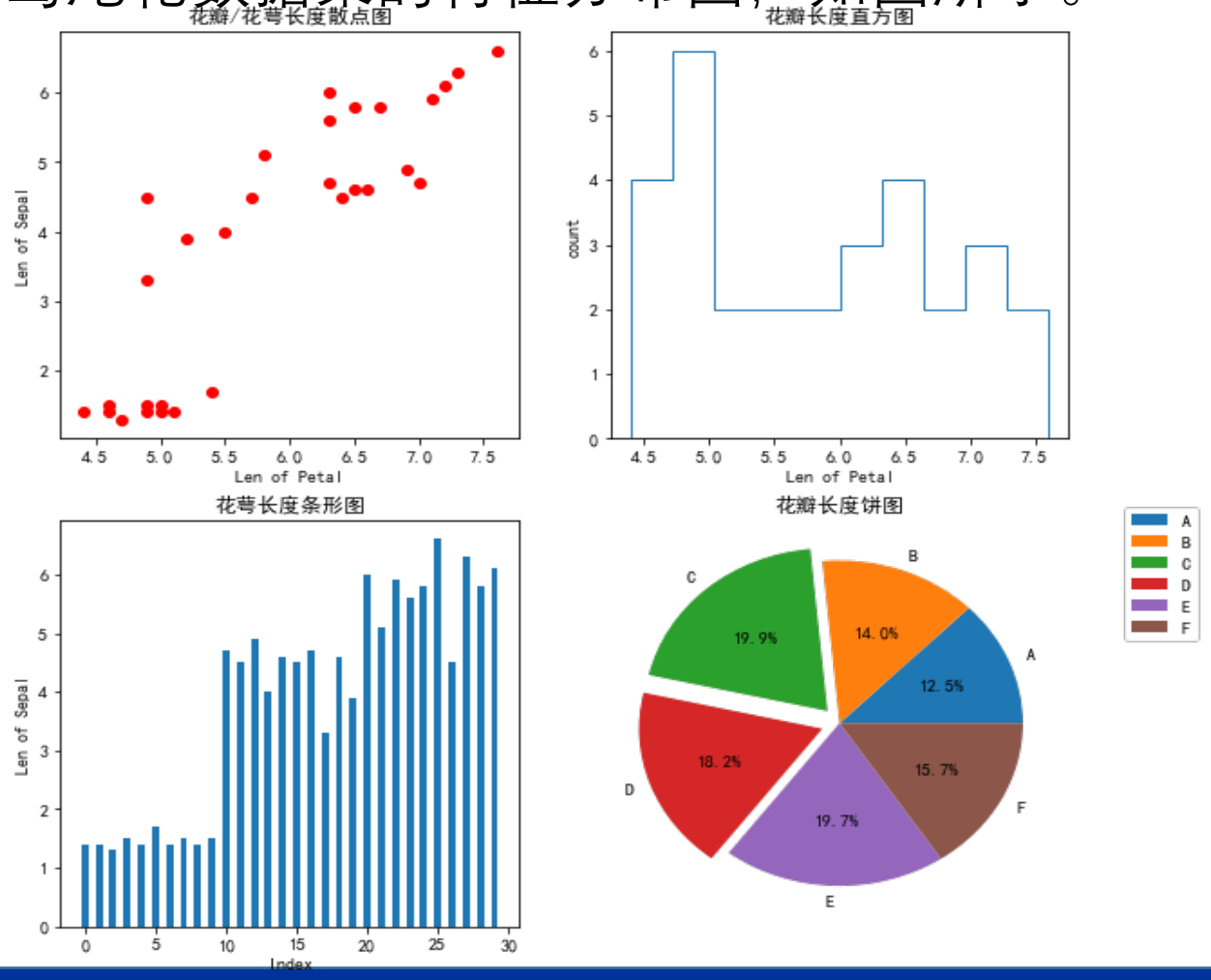


4. 其他类型的图表

- 在实际应用中，需要很多类型的图表。matplotlib.pyplot提供了丰富的绘图函数可供选择，包括：scatter（散点图）、bar（条形图）、pie（饼图）、hist（直方图）以及的plot（坐标图）。

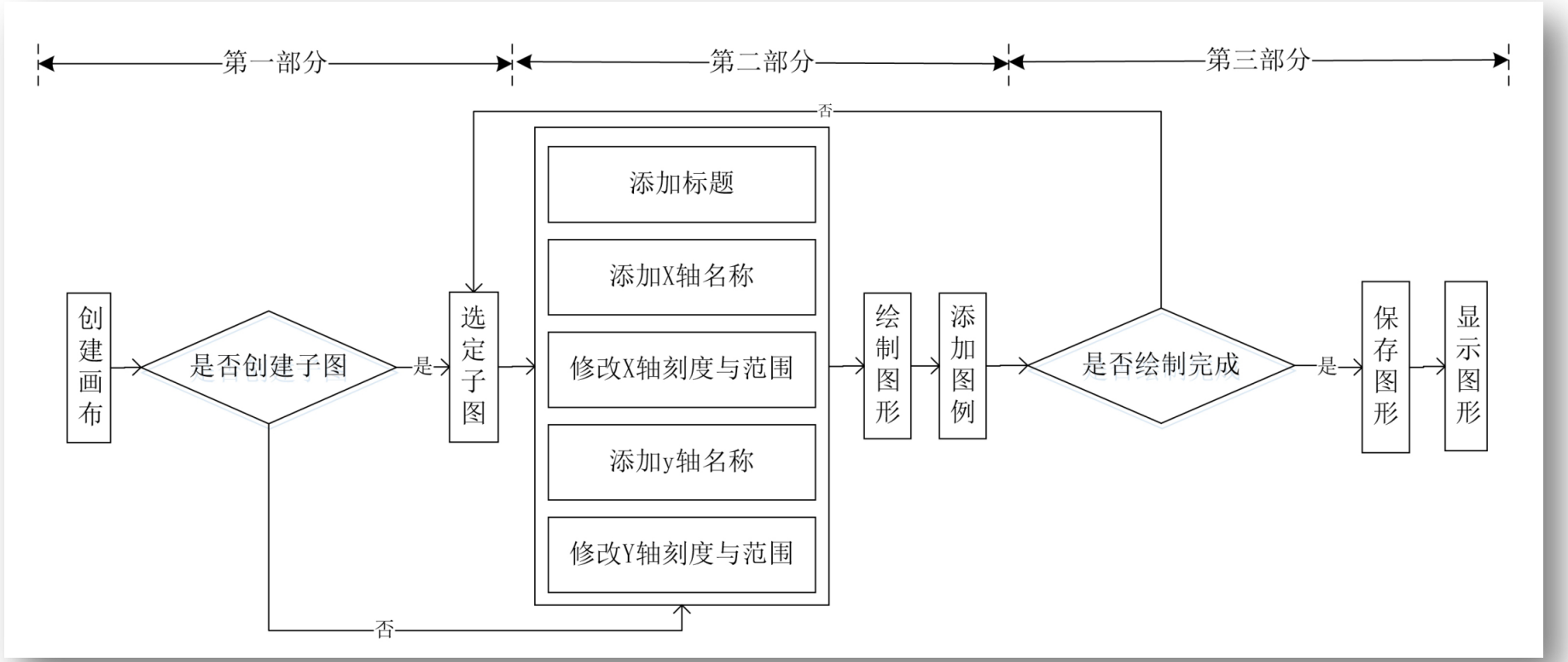
多子图综合练习

【例3.63】绘制鸢尾花数据集的特征分布图，如图所示。



```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
data = pd.read_csv('iris.txt',',',header=None)  #读取鸢尾花数据文件
df=pd.DataFrame(data)                          #转化为dataframe数据类型
df.columns = ['LenPetal','LenSepal']      #花瓣长度，花萼长度两个特征
plt.rcParams['font.sans-serif']=['SimHei']      #显示中文
#============图表1=============
plt.figure(figsize=(10, 10))
plt.subplot(2,2,1)
plt.xlabel("Len of Petal", fontsize=10)       #横轴标签
plt.ylabel("Len of Sepal", fontsize=10)       #纵轴标签
plt.title("花瓣/花萼长度散点图")                 #图表标题
plt.scatter(df['LenPetal'],df['LenSepal'],c='red')  #绘制两个特征数据
#============图表2=============
plt.subplot(2,2,2)
plt.title("花瓣长度直方图")
plt.xlabel("Len of Petal", fontsize=10)       #横轴标签
plt.ylabel("count", fontsize=10)              #纵轴标签
plt.hist(df['LenPetal'],histtype ='step')       #绘制花瓣长度分布直方图
#============图表3=============
x=np.arange(30)
plt.subplot(2,2,3)
plt.xlabel("Index", fontsize=10)                #横轴标签
plt.ylabel("Len of Sepal", fontsize=10)     #横轴标签
plt.title("花萼长度条形图")
plt.bar(x,height=df['LenSepal'], width=0.5)  #绘制花萼数据条形图
#============图表4=============
plt.subplot(2,2,4)
sizes = [2,5,12,70,2,9]
explode = (0,0,0.1,0.1,0,0)
labels = ['A','B','C','D','E','F']
plt.title("花瓣长度饼图")
plt.pie(df['LenPetal'][8:14],explode=explode,autopct='%1.1f%%',labels=labels)  #饼图
plt.legend(loc="upper left",fontsize=10,bbox_to_anchor=(1.1,1.05))
plt.show()
```

基本绘图流程


第一部分
第二部分
第三部分
否
创建画布
是否创建子图
是
选定子图
添加标题
添加X轴名称
修改X轴刻度与范围
添加y轴名称
修改Y轴刻度与范围
绘制图形
添加图例
是否绘制完成
是
保存图形
显示图形
否

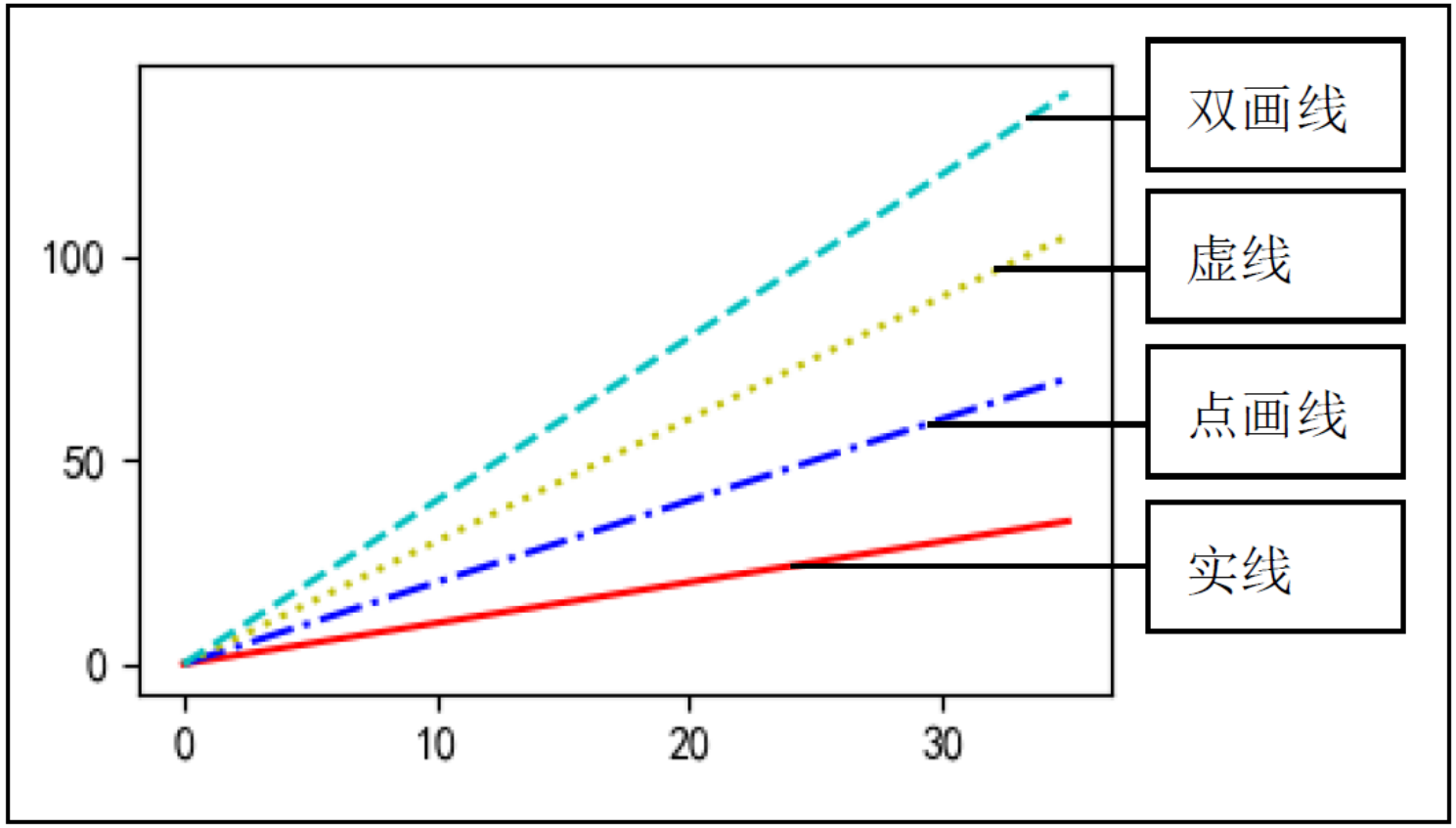
图表的常用设置

- ✓ 基本绘图plot()函数
- ✓ 设置画布
- ✓ 设置坐标轴
- ✓ 添加文本标签
- ✓ 设置标题和图例
- ✓ 添加注释
- ✓ 调整图表与画布边缘间距
- ✓ 其他设置

通用颜色

设 置 值	说　明	设 置 值	说　明
b	蓝色	m	洋红色
g	绿色	y	黄色
r	红色	k	黑色
c	蓝绿色	w	白色
#FFFF00	黄色，十六进制颜色值	0.5	灰度值字符串

线条样式



标记设置

标　记	说　明	标　记	说　明	标　记	说　明
.	点标记	1	下花三角标记	h	竖六边形标记
,	像素标记	2	上花三角标记	H	横六边形标记
o	实心圆标记	3	左花三角标记	+	加号标记
v	倒三角标记	4	右花三角标记	x	叉号标记
^	上三角标记	s	实心正方形标记	D	大菱形标记
>	右三角标记	p	实心五角星标记	d	小菱形标记
<	左三角标记	*	星形标记	|	垂直线标记

图例位置参数设置值

位置（字符串）	描　述	位置（字符串）	描　述
best	自适应	center left	左侧中间位置
upper right	右上方	center right	右侧中间位置
upper left	左上方	upper center	上方中间位置
lower left	左下方	lower center	下方中间位置
lower right	右下方	center	正中央
right	右侧		

相关函数简介

- figure（）：创建一个新的绘图窗口。
- figtext（）：为figure添加文字
- axes（）：为当前figure添加一个坐标轴
- plot（）：绘图函数
- polar（）：绘制极坐标图
- axis（）：获取或设置轴属性的边界方法（坐标的取值范围）
- clf ： 清除当前figure窗口　　cla ： 清除当前axes窗口
- close ： 关闭当前figure窗口
- subplot ： 一个图中包含多个axes
- text（）： 在轴上添加文字
- title（）： 设置当前axes标题
- xlabel/ylabel：设置当前X轴或Y轴的标签

相关函数简介

- hist（）：绘制直方图
- hist2d（）：绘制二维在直方图
- hold　：设置当前图窗状态；off或者on
- imread（）：读取一个图像，从图形文件中提取数组
- legend（）：为当前axes放置标签
- pie（）：绘制饼状图
- scatter（）：做一个X和Y的散点图，其中X和Y是相同长度的序列对象
- stackplot（）：绘制一个堆叠面积图
- acorr（）：绘制X的自相关函数
- annotate（）：用箭头在指定的数据点创建一个注释或一段文本
- bar（）：绘制垂直条形图　　barh（）：绘制横向条形图
- barbs（）：绘制一个倒钩的二维场

掌握pyplot基础语法

1.创建画布与创建子图

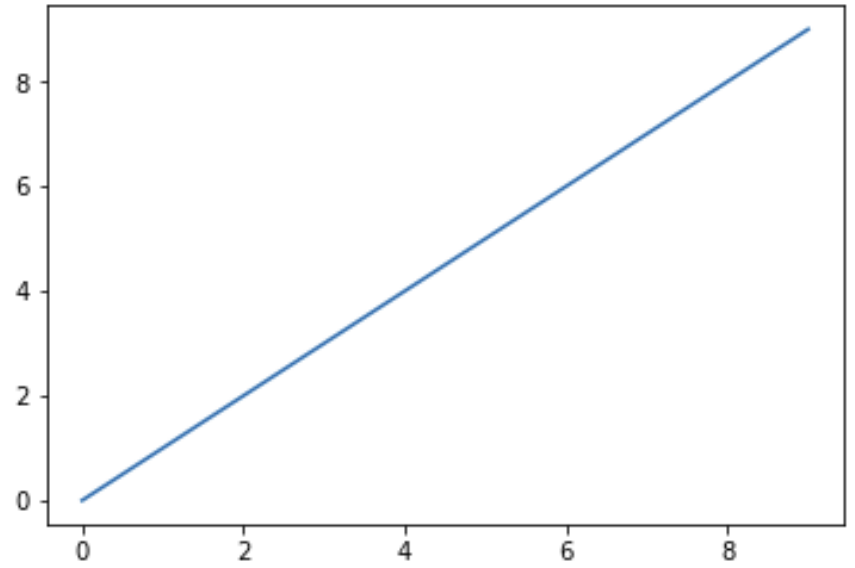
函数名称	函数作用
plt.figure	创建一个空白画布，可以指定画布大小，像素。
figure.add_subplot	创建并选中子图，可以指定子图的行数，列数，与选中图片编号。

第一部分主要作用是构建出一张空白的画布，并可以选择是否将整个画布划分为多个部分，方便在同一幅图上绘制多个图形的情况。最简单的绘图可以省略第一部分，而后直接在默认的画布上进行图形绘制。

Matplotlib绘图基础

```
import matplotlib.pyplot as plt
import numpy as np
data=np.arange(10)
plt.plot(data)
```

```
Out[2]: [<matplotlib.lines.Line2D at 0x1e2c7508b00>]
```



绘制的图位于图片（figure）对象中。

掌握pyplot基础语法

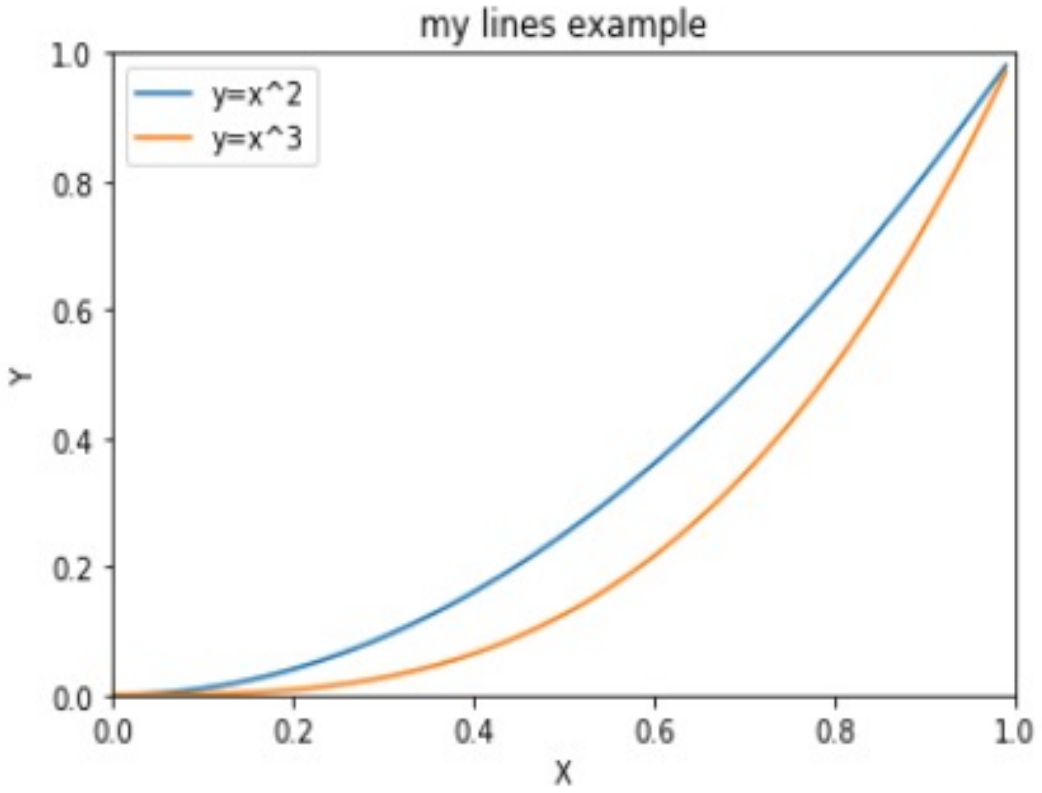
2.添加画布内容

函数名称	函数作用
plt.title	在当前图形中添加标题，可以指定标题的名称、位置、颜色、字体大小等参数。
plt.xlabel	在当前图形中添加x轴名称，可以指定位置、颜色、字体大小等参数。
plt.ylabel	在当前图形中添加y轴名称，可以指定位置、颜色、字体大小等参数。
plt.xlim	指定当前图形x轴的范围，只能确定一个数值区间，而无法使用字符串标识
plt.ylim	指定当前图形y轴的范围，只能确定一个数值区间，而无法使用字符串标识
plt.xticks	指定x轴刻度的数目与取值。
plt.yticks	指定y轴刻度的数目与取值。
plt.legend	指定当前图形的图例，可以指定图例的大小、位置、标签。

第二部分是绘图的主体部分。其中添加标题，坐标轴名称，绘制图形等步骤是并列的，没有先后顺序，可以先绘制图形，也可以先添加各类标签。但是添加图例一定要在绘制图形之后。

6.2掌握pyplot基础语法

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
data=np.arange(0,1,0.01)
plt.title('my lines example')
plt.xlabel('X')
plt.ylabel('Y')
plt.xlim(0,1)
plt.ylim(0,1)
plt.xticks([0,0.2,0.4,0.6,0.8,1])
plt.yticks([0,0.2,0.4,0.6,0.8,1])
plt.plot(data,data**2)
plt.plot(data,data**3)
plt.legend(['y=x^2','y=x^3'])
plt.show()
```



6.2 掌握pyplot基础语法

◆3.绘图的保存与显示

函数名称	函数作用
plt.savafig	保存绘制的图片，可以指定图片的分辨率、边缘的颜色等参数。
plt.show	在本机显示图形。

第三部分主要用于保存和显示图形。

例:

```
fig.savefig(save_path, format='png', transparent=True, dpi=300, pad_inches = 0)
```

6.3设置pyplot的动态rc参数

- pyplot使用rc配置文件来自定义图形的各种默认属性，被称为rc配置或rc参数。
- 在pyplot中几乎所有的默认属性都是可以控制的，例如视图窗口大小以及每英寸点数、线条宽度、颜色和样式、坐标轴、坐标和网格属性、文本、字体等。

两种方式可以设置参数，即全局参数定制和rc设置方法。

查看matplotlib的rc参数：

```
import matplotlib as plt
print(plt.rc_params())
```

6.3设置pyplot的动态rc参数：查看rc参数

1.使用参数字典

```
import matplotlib as plt
print(plt.rc_params())
```

参数众多

常用参数：

Axes：设置坐标轴边界、颜色、坐标刻度值大小和网格的显示；

Figure：设置边界颜色、图形大小和子区；

Font：设置字号、字体和样式；

Grid：设置网格颜色和线型；

Legend：设置图例和其中的文本显示；

Lines：设置线条颜色、宽度、线型等；

Savefig：对保存图像进行单独设置；

Xtick和ytick：X、Y轴的主刻度和次刻度设置颜色、大小、方向和标签大小。

```
_internal.classic_mode: False
agg.path.chunksize: 0
animation.avconv_args: []
animation.avconv_path: avconv
animation.bitrate: -1
animation.codec: h264
animation.convert_args: []
animation.convert_path: convert
animation.embed_limit: 20.0
animation.ffmpeg_args: []
animation.ffmpeg_path: ffmpeg
animation.frame_format: png
animation.html: none
animation.html_args: []
animation.writer: ffmpeg
axes.autolimit_mode: data
axes.axisbelow: line
axes.edgecolor: k
axes.facecolor: w
```

6.3设置pyplot的动态rc参数

2. 线条的常用rc参数名称、解释与取值

rc参数名称	解释	取值
lines.linewidth	线条宽度	取0-10之间的数值，默认为1.5。
lines.linestyle	线条样式	可取“-”“--”“-.”“：”四种。默认为“-”。
lines.marker	线条上点的形状	可取“o” “D” “h” “.” “,” “S”等20种，默认为None。
lines.markersize	点的大小	取0-10之间的数值，默认为1。

6.3设置pyplot的动态rc参数

◆常用线条类型解释

linestyle取值	意义	linestyle取值	意义
-	实线	-.	点线
--	长虚线	:	短虚线

6.3设置pyplot的动态rc参数

marker取值	意义	marker取值	意义
‘o’	圆圈	‘.’	点
‘D’	菱形	‘s’	正方形
‘h’	六边形1	‘*’	星号
‘H’	六边形2	‘d’	小菱形
‘-’	水平线	‘v’	一角朝下的三角形
‘8’	八边形	‘<’	一角朝左的三角形
‘p’	五边形	‘>’	一角朝右的三角形
‘,’	像素	‘^’	一角朝上的三角形
‘+’	加号	‘\’	竖线
‘None’	无	‘x’	X

6.3设置pyplot的动态rc参数

- 需要注意的是，由于默认的Pyplot字体并不支持中文字符的显示，因此需要通过设置font.sans-serif参数改变绘图时的字体，使得图形可以正常显示中文。同时，由于更改字体后，会导致坐标轴中的部分字符无法显示，因此需要同时更改axes.unicode_minus参数。

```
plt.rcParams['font.family'] = ['SimHei'] #用来显示中文标签
plt.rcParams['axes.unicode_minus'] = False  #用来正常显示负号
```

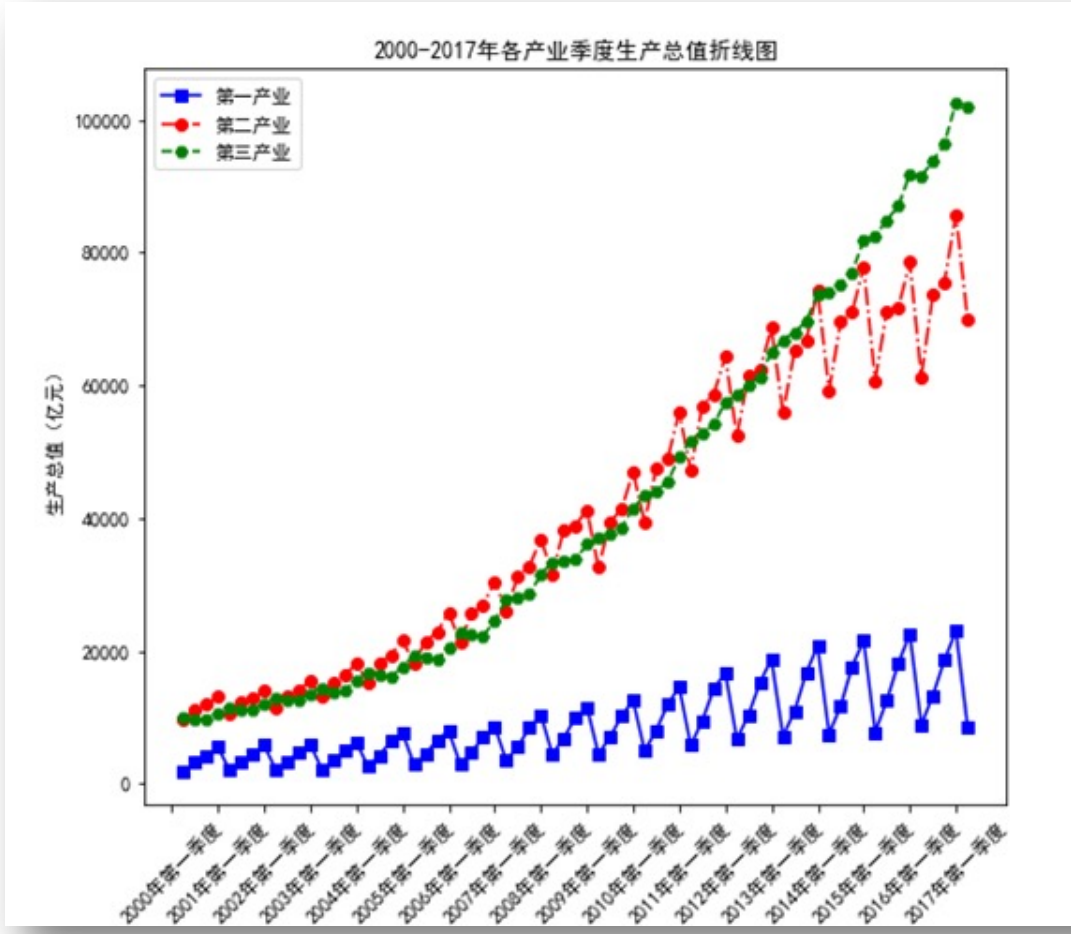
除了设置线条和字体的rc参数外，还有设置文本、箱线图、坐标轴、刻度、图例、标记、图片、图像保存等rc参数。

折线图、散点图

折线图

折线图

- 折线图（line chart）是一种将数据点按照顺序连接起来的图形。可以看作是将散点图，按照x轴坐标顺序连接起来的图形。
- 折线图的主要功能是查看因变量y随着自变量x改变的趋势，最适合用于显示随时间（根据常用比例设置）而变化的连续数据。同时还可以看出数量的差异，增长趋势的变化。



折线图

- 通过plot()函数绘制，可以指定线条的样式，点的标记以及颜色等。

```
plot(*args, scalex=True, scaley=True, data=None, **kwargs)
```

plot() 方法用于绘制点或线等，需指定所有点的坐标，可同时绘制多条线。

① plot([x], y, [fmt], *, data=None, **kwargs)

② plot([x], y, [fmt], [x2], y2, [fmt2], ..., **kwargs)

- kwargs：关键字参数，例如：color （颜色）、marker（标记）、linestyle（线条样式）、linewidth（线条宽度）、markersize（标记大小）、label（标签，用于图例）等。

plot函数

- plot函数在官方文档的语法中只要求填入不定长参数，实际可以填入的主要参数主要如下。

- x：**x轴坐标**，可选，当没有传值时，**采用默认值**，值的个数与y轴坐标个数相同，**从0开始不断增大**；
- y：**y轴坐标**，不可省略，通常是一个数或一维数组；
- data:可选，通常为**可索引的对象**，例如**字典**、**DataFrame**等；
- fmt:定义基本样式，由**[颜色][点的标记][线条样式]**三部分字符组成，也可**通过关键字参数设置**；
  - color：**接收特定string。指定线条的颜色。默认为None。**
  - linestyle：接收特定string。指定线条类型。默认为“-”。
  - marker：接收特定string。表示绘制的点的类型。默认为None。
  - alpha：接收0-1的小数。表示点的透明度。默认为None。

常见的样式含义

部分颜色字符及其含义

颜色字符	说明	颜色字符	说明
‘b’	蓝色	‘m’	洋红色
‘g’	绿色	‘y’	黄色
‘r’	红色	‘k’	黑色
‘c’	青绿色	‘w’	白色

如果通过**color关键字**参数单独设置颜色，可使用**颜色单词全称**（如“green”）以及**十六进制字符串**（如“#aabbcc”）。

部分线条样式字符及其含义

线条样式字符	说明	线条样式字符	说明
‘-’	实线	‘:’	虚线
‘--’	破折线	空格	无线条
‘-.’	点划线		

部分标记字符及其含义

标记字符	说明	标记字符	说明
s	实心方形	v	下三角形
p	实心五角	^	上三角形
o	实心圆	<	左三角形
.x	x 字	>	右三角形
+	+ 字	1	下三角
*	星型	2	上三角
h	六角形（边朝上）	3	左三角
H	六角形（角朝上）	4	右三角
|	竖线 （|）	_	横线（_）
D	菱形	d	瘦菱形

注意：**三种字符顺序可打乱**，也**可省略**部分，没指定线条样式时，将**不显示线条**，没指定标记则**不显示点**。

rcParams中常见的属性及其默认值

属性名	含义	默认值
axes.facecolor	坐标系的背景颜色	white
axes.edgecolor	坐标系的边框颜色	black
axes.linewidth	轴线粗细	0.8
axes.spines.right	右边的轴线	True
axes.unicode_minus	Unicode编码的减号	True
figure.dpi	图的分辨率	100.0
figure.figsize	图的大小，单位英寸	[6.4, 4.8]
figure.subplot.hspace	子块之间的高度	0.2
figure.subplot.wspace	子块之间的宽度	0.2
font.size	字体大小	10
font.family	字体	['sans-serif']
lines.linewidth	线条粗细	1.5
lines.markersize	线中标记大小	6.0
savefig.format	图保存的格式	png

常见中文字体

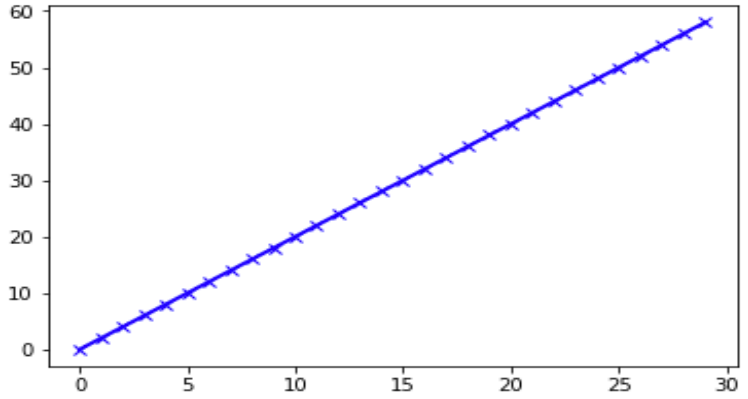
中文字体	说明
‘SimHei’	中文黑体
‘Kaiti’	中文楷体
‘LiSu’	中文隶书
‘FangSong’	中文仿宋
‘YouYuan’	中文幼圆
‘STSong’	华文宋体

采用中文字体时，部分字体下，**负号不能正常显示**，需设置 **axes.unicode_minus** 为 **False** 。

6.4 pyplot中的常用绘图：折线图

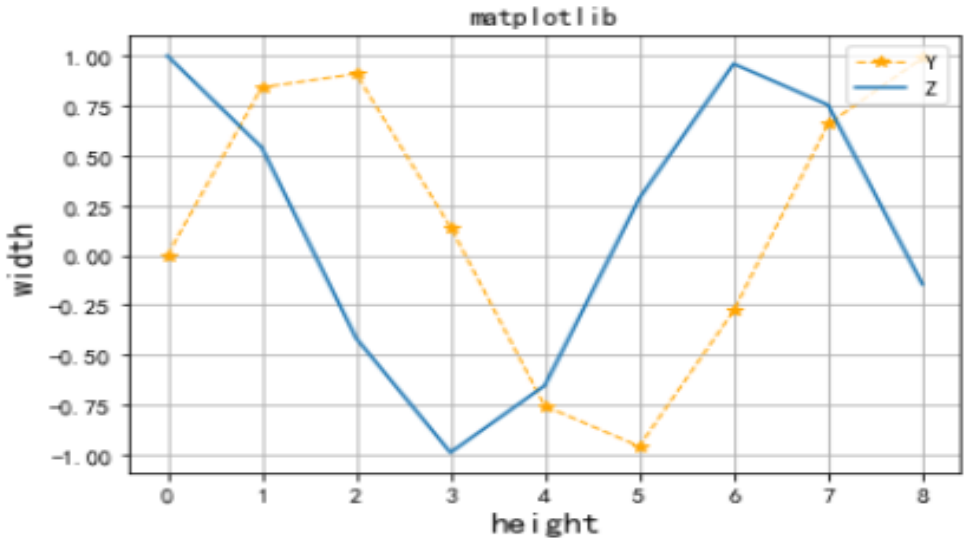
1. 简单折线图

```
import numpy as np
x1 = np.arange(0, 30)
plt.plot(x1,x1*2, 'b')
plt.show()
```

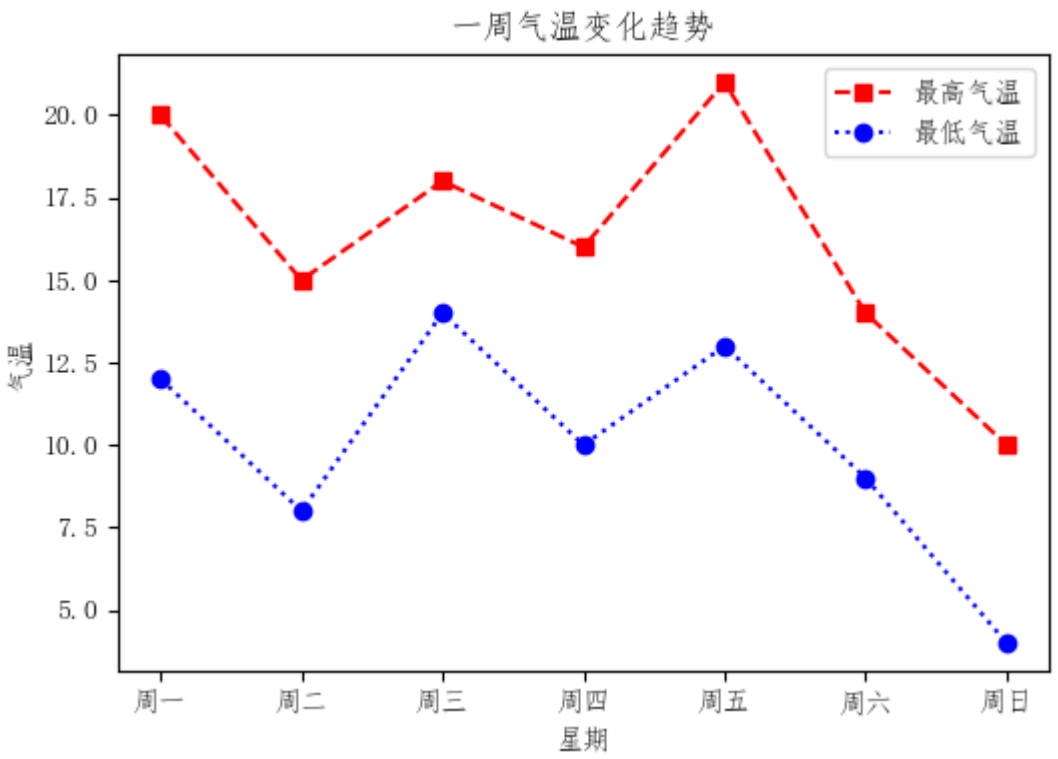


2. 带点的折线图

```
plt.plot(x, y, marker='*',
linewidth=1, linestyle='--',
color='orange')
plt.plot(x, z)
```



折线图 练习1




```
import matplotlib.pyplot as plt
plt.figure(num="气温趋势", figsize=(6, 4))  # 绘图区域大小
plt.rcParams['font.family'] = 'STSong'  # 设置字体
x = ["周一", "周二", "周三", "周四", "周五", "周六", "周日"]
y_1 = [20, 15, 18, 16, 21, 14, 10]  # 一周最高气温
y_2 = [12, 8, 14, 10, 13, 9, 4]  # 一周最低气温
plt.title("一周气温变化趋势")  # 图片标题
plt.xlabel("星期")  # X轴标签
plt.ylabel("气温")  # Y轴标签
plt.plot(x, y_1, "rs--", label="最高气温")  # 绘制最高气温
plt.plot(x, y_2, "bo:", label="最低气温")  # 绘制最低气温
plt.legend()  # 显示图例
plt.savefig("aaa.png")  # 保存图片
plt.show()  # 显示图片
```

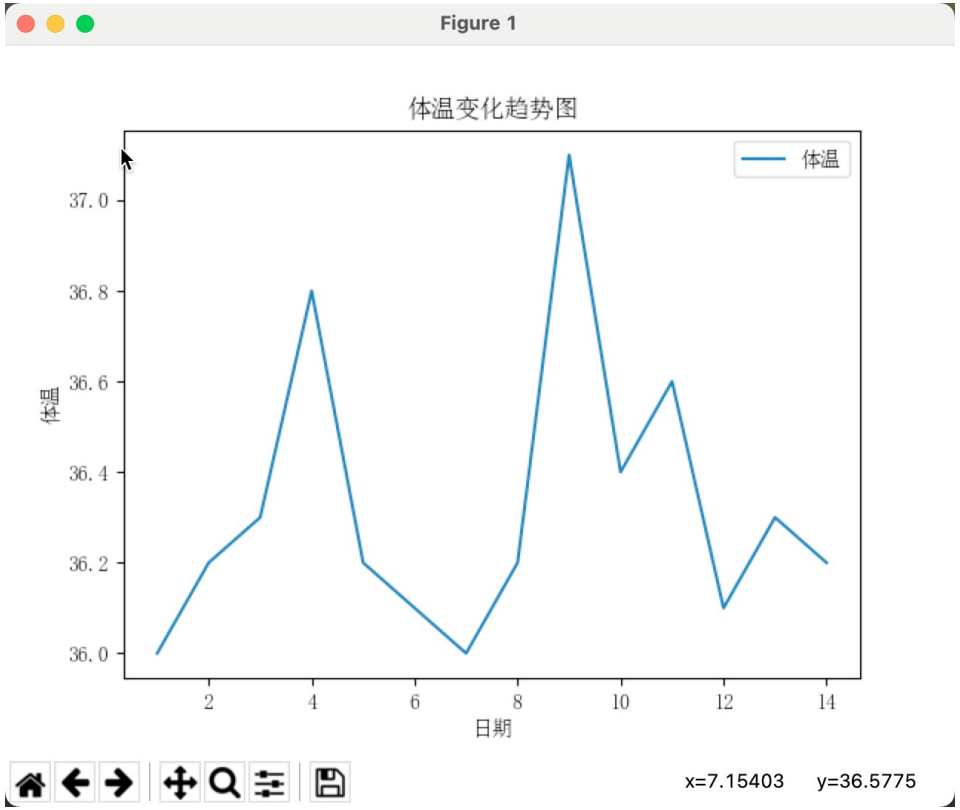
注意：**pyplot 默认不支持中文显示**，两种解决方案：

- 使用 **rcParams['font.family']属性修改字体，**此时，整个图中的字体都会改变；
- 在需要显示中文的地方，增加一个属性：**fontproperties**，此时只修饰部分地方，其他地方的字体不会跟随改变；

折线图 练习2

- 从体温.csv文件读取数据，绘制折线图。

	A	B
1	日期	体温
2	1	36
3	2	36.2
4	3	36.3
5	4	36.8
6	5	36.2
7	6	36.1
8	7	36
9	8	36.2
10	9	37.1
11	10	36.4
12	11	36.6
13	12	36.1
14	13	36.3
15	14	36.2
16		

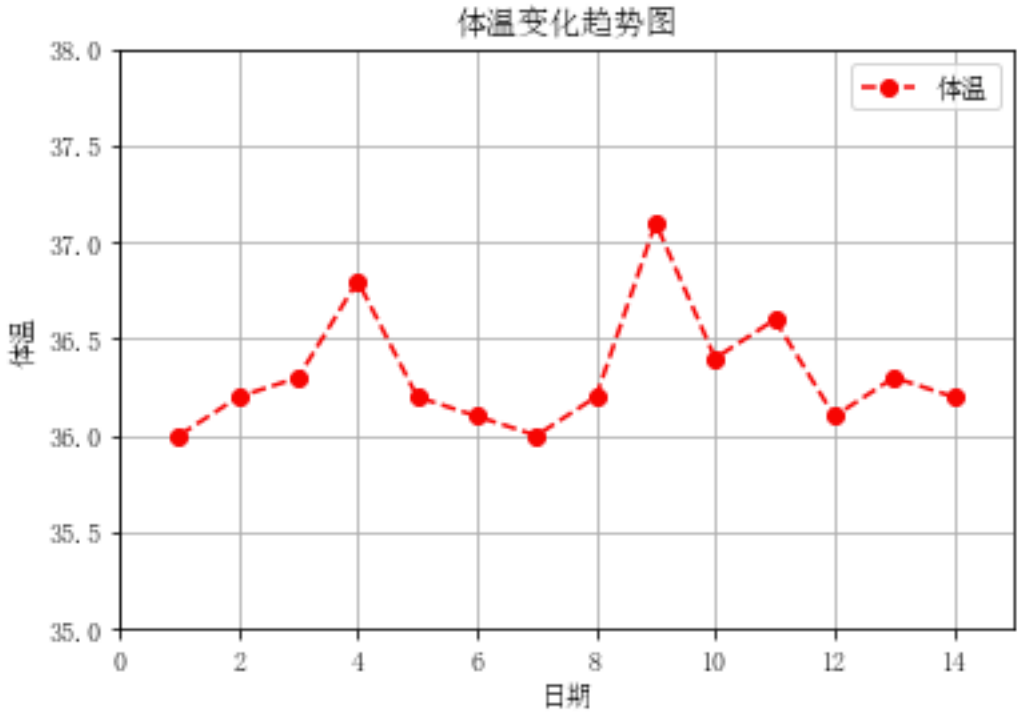


参考代码

```
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_excel('data/体温.xls') #导入Excel文件
#折线图
x =df['日期']                          #x轴数据
y=df['体温']                           #y轴数据
plt.plot(x,y)
plt.xlabel('日期')
plt.ylabel('体温')
plt.title('体温变化趋势图')
plt.legend()
plt.show()
```

思考

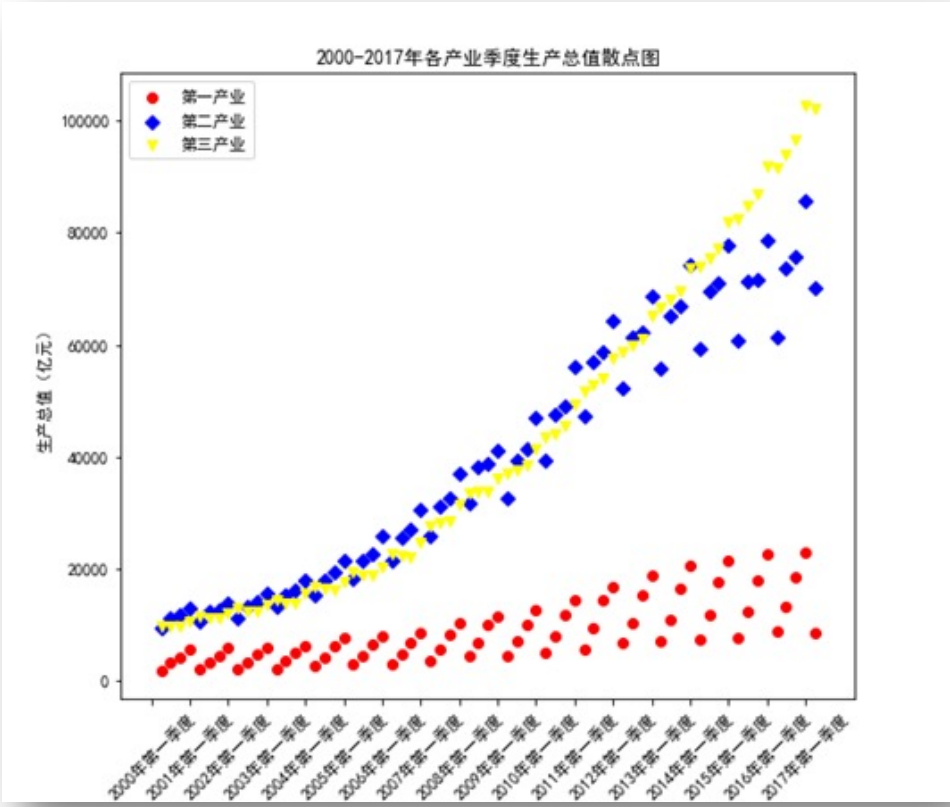
- 如果要画出如下的图，需要设定哪些参数？



散点图

散点图

- 散点图（scatter diagram）又称为散点分布图，是以一个特征为横坐标，另一个特征为纵坐标，利用坐标点（散点）的分布形态反映特征间的统计关系的一种图形。
- 值是由点在图表中的位置表示，类别是由图表中的不同标记表示，通常用于比较跨类别的数据。



散点图

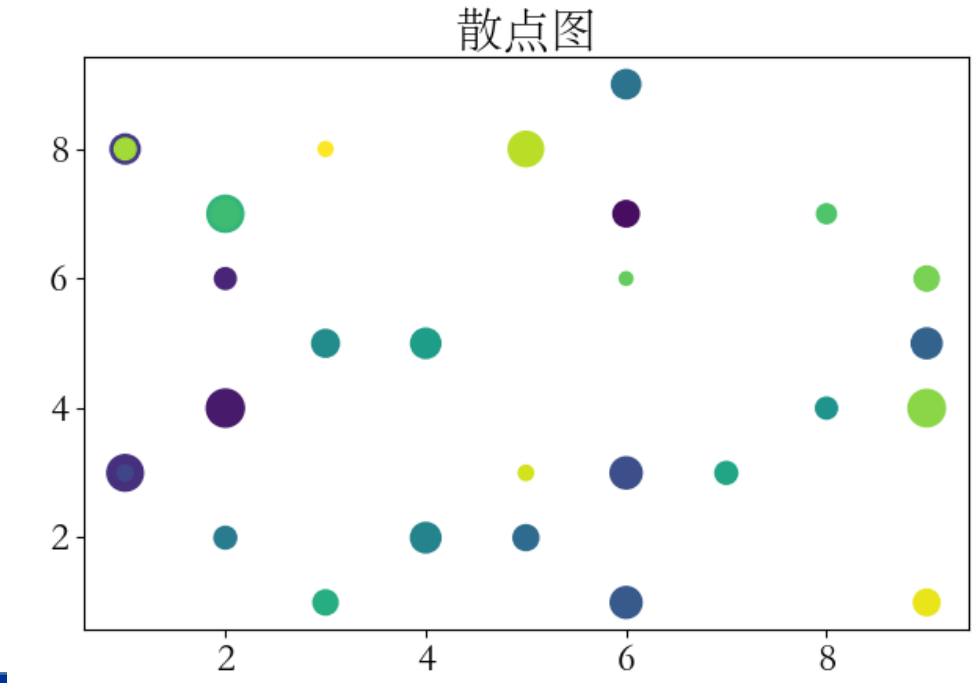
- 用**不同颜色、不同大小的点**表示数据之间的关系。Pyplot 中绘制散点图的函数为 scatter()，主要参数如下：

```
scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None,
vmax=None, alpha=None, **kwargs)
```

  - **x**：散点图中点的**x轴坐标**；
  - **y**：散点图中点的**y轴坐标**；
  - **s**：散点图**点的大小**，默认为20，标量或数组；
  - **c**：散点图**点的颜色**，默认为蓝色，标量或数组；
  - **marker**：指定散点图**点的形状**，默认为圆形；
  - **alpha**：设置**散点的透明度**；
  - **cmap**：**颜色映射**，可选，将颜色映射到已有色系，例如**plt.cm.Blues**；

散点图案例

编程程序，绘制散点图，要求图中的**每个散点随机呈现不同的大小和颜色**。效果如图所示，**随机生成30个点**，x轴坐标和y轴坐标都在**1-10**之间，颜色取值在**0-30**之间。大小在**30-300**之间。

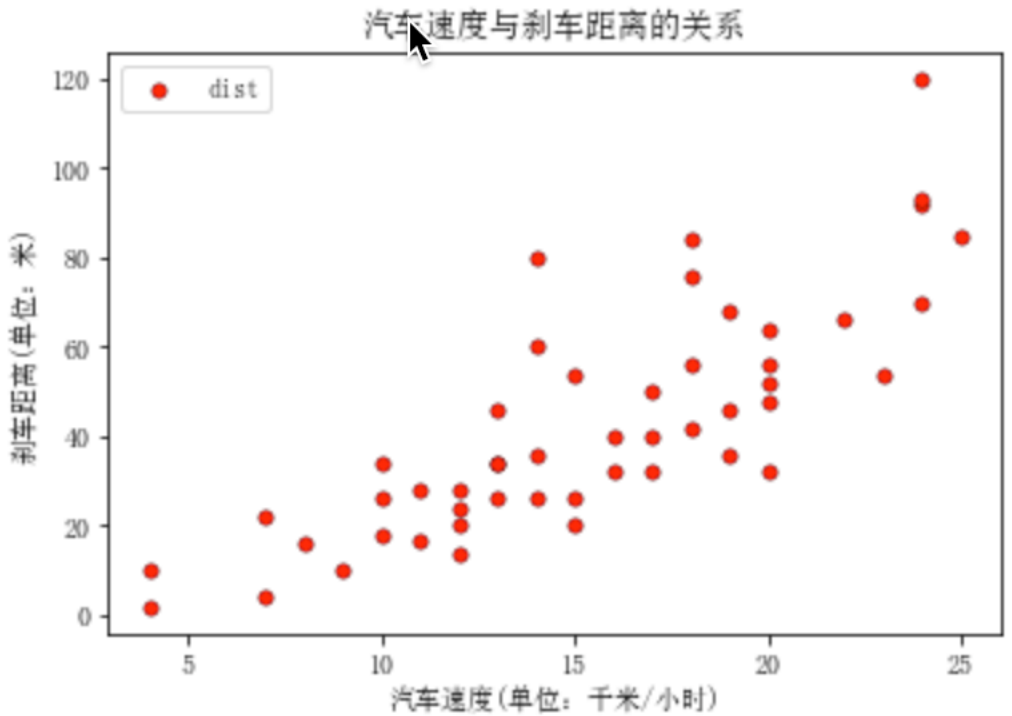



```
import matplotlib.pyplot as plt
import numpy as np
num = 30
plt.rcParams['font.family'] = 'STSong'  # 设置字体
plt.rcParams['font.size'] = 18  # 设置字体大小
x_scatter = np.random.randint(1, 10, num)
y_scatter = np.random.randint(1, 10, num)
colors = range(num)
size = np.random.randint(30, 300, num)
plt.scatter(x_scatter, y_scatter, c=colors, s=size)
plt.title("散点图")
plt.savefig("figure")  # 保存图片
plt.show()  # 显示图片
```

散点图练习1

- 从cars.csv读取数据，绘制散点图

	A	B
1	speed	dist
2	4	2
3	4	10
4	7	4
5	7	22
6	8	16
7	9	10
8	10	18
9	10	26
10	10	34
11	11	17
12	11	28
13	12	14
14	12	20
15	12	24
16	12	28
17	13	26
18	13	34
19	13	34
20	13	46
21	14	26
22	14	36
23	14	60
24	14	80



参考代码

```
# 导入模块
import pandas as pd
import matplotlib.pyplot as plt
```

```
# plt.figure()
plt.rcParams['font.family']='SimSun' #设置matplotlib支持中文显示 ， 注意要设置成你电脑上已安装的字体;
# # 设置绘图风格
# plt.style.use('ggplot')
# 设置中文编码和符号的正常显示
plt.rcParams['font.sans-serif'] = 'Microsoft YaHei'
plt.rcParams['axes.unicode_minus'] = False
```

In [3]:
```
# 读入数据
cars = pd.read_csv('cars.csv')
print(cars.head())
```

In [5]:
```
# 绘图
plt.scatter(cars['speed'], # x轴数据为汽车速度
            cars['dist'], # y轴数据为汽车的刹车距离
            s = 30, # 设置点的大小
            c = 'red', # 设置点的颜色
            marker = 'o', # 设置点的形状为圆圈
            linewidths = 0.3, # 设置散点边界的粗细
            edgecolors = 'black' # 设置散点边界的颜色
            )
# 添加轴标签和标题
plt.title('汽车速度与刹车距离的关系')
plt.xlabel('汽车速度(单位：千米/小时)')
plt.ylabel('刹车距离(单位：米)')
# 去除图边框的顶部刻度和右边刻度
plt.tick_params(top=False, right = False)
plt.legend()
# 显示图形
# plt.show()
```