python绘图及数据可视化备课

python绘图及数据可视化备课
补充 上次实验课的库与爬虫
1 jieba库进行词频分析
一、jieba库基本介绍
二、jieba库使用说明
2 re库进行正则匹配
校验数字的表达式
校验字符的表达式
特殊需求表达式
5.0 问题导入
5.1 matplotlib库基础
拓展：默认值相关
5.2 plt.plot()绘图
5.3 划分子图
拓展三张图如何划分?
5.4: 条形图
5.4.2 多组条形图
拓展图例的位置
5.5 散点图
拓展：三维散点图
5 . 6 饼图
拓展：极轴饼图
拓展: 多层嵌套饼图
5.7 直方图
5.8 箱线图
5.9小提琴图
5.10热力图
5.11云图

补充 上次实验课的库与爬虫

于刊老师担心我准备的内容不够讲两节课的，如果我讲完这一章的内容还没有结束的话我就讲一讲我最近捣鼓的东西

1 jieba库进行词频分析

一、jieba库基本介绍

> jieba是优秀的中文分词第三方库，需要额外安装，安装方式为**pip install jieba**,jieba库提供三种分词模式，最简单只需掌握一个函数
>
> 利用一个中文词库，可以确定汉字之间的关联概率,汉字间概率大的组成词组，形成分词结果

二、jieba库使用说明

```
#精确模式、全模式、搜索引擎模式精确模式：
#把文本精确的切分开，不存在冗余单词
#全模式：把文本中所有可能的词语都扫描出来，有冗余
#搜索引擎模式：在精确模式基础上，对长词再次切分
```

把文本精确的切分开，不存在冗余单词

全模式：把文本中所有可能的词语都扫描出来，有冗余

搜索引擎模式：在精确模式基础上，对长词再次切分

jieba库常用函数

函数	描述
jieba.cut(s)	精确模式，返回一个可迭代的数据类型
jieba.cut(s, cut_all=True)	全模式，输出文本s中所有可能单词
jieba.cut_for_search(s)	搜索引擎模式，适合搜索引擎建立索引的分词结果
jieba.lcut(s)	精确模式，返回一个列表类型，建议使用
jieba.lcut(s, cut_all=True)	全模式，返回一个列表类型，建议使用
jieba.lcut_for_search(s)	搜索引擎模式，返回一个列表类型，建议使用
jieba.add_word(w)	向分词词典中增加新词w

2 re库进行正则匹配

正则表达式（英文名称：regular expression，regex，RE）是用来简洁表达一组字符串特征的表达式。最主要应用在字符串匹配中。

正则表达式的常用操作符(1)

操作符	说明	实例
.	表示任何单个字符	
[ ]	字符集，对单个字符给出取值范围	[abc]表示a、b、c，[a-z]表示a到z单个字符
[^]	非字符集，对单个字符给出排除范围	[^abc]表示非a或b或c的单个字符
*	前一个字符0次或无限次扩展	abc* 表示 ab、abc、abcc、abccc等
+	前一个字符1次或无限次扩展	abc+ 表示 abc、abcc、abccc等
?	前一个字符0次或1次扩展	abc? 表示 ab、abc
|	左右表达式任意一个	abc|def 表示 abc、def

正则表达式的常用操作符(2)

操作符	说明	实例
{m}	扩展前一个字符m次	ab{2}c表示abbc
{m,n}	扩展前一个字符m至n次（含n）	ab{1,2}c表示abc、abbc
^	匹配字符串开头	^abc表示abc且在一个字符串的开头
$	匹配字符串结尾	abc$表示abc且在一个字符串的结尾
( )	分组标记，内部只能使用 | 操作符	(abc)表示abc，(abc|def)表示abc、def
\d	数字，等价于[0-9]	
\w	单词字符，等价于[A-Za-z0-9_]	

Re库主要功能函数

Re库是Python的标准库，主要用于字符串匹配，调用方式：import re

1、re库采用 raw string 类型（原生字符串类型），不用对转义符再次转义。表示为：r'text' 例如：re.search(r'\d{3}-\d{8}',string)

2、re库采用 string 类型表示正则表达式，则需要转义。表示：'\\d{3}-\\d{8}' 例如：re.search('\\d{3}-\\d{8}',string)

函数	说明
re.search()	在一个字符串中搜索匹配正则表达式的第一个位置，返回match对象
re.match()	从一个字符串的开始位置起匹配正则表达式，返回match对象
re.findall()	搜索字符串，以列表类型返回全部能匹配的子串
re.split()	将一个字符串按照正则表达式匹配结果进行分割，返回列表类型
re.finditer()	搜索字符串，返回一个匹配结果的迭代类型，每个迭代元素是match对象
re.sub()	在一个字符串中替换所有匹配正则表达式的子串，返回替换后的字符串

在这里需要注意，match对象是一种特殊的类型，如**re.search()**与**re.match()**虽然也能做到类似的效果，但实际使用是有问题的，在这里使用的**findall()**函数实现。

Match对象的属性：

属性	说明
.string	待匹配的文本
.re	匹配时使用的patter对象（正则表达式）
.pos	正则表达式搜索文本的开始位置
.endpos	正则表达式搜索文本的结束位置

Match对象的方法：

方法	说明
.group(0)	获得匹配后的字符串
.start()	匹配字符串在原始字符串的开始位置
.end()	匹配字符串在原始字符串的结束位置
.span()	返回(.start(), .end())

校验数字的表达式

```
数字：^[0-9]\*$
n位的数字：^\d{n}$
至少n位的数字：^\d{n,}$
m-n位的数字：^\d{m,n}$
零和非零开头的数字：^(0|[1-9][0-9]\*)$
非零开头的最多带两位小数的数字：^([1-9][0-9]\*)+(\.[0-9]{1,2})?$
带1-2位小数的正数或负数：^(\-)?\d+(\.\d{1,2})$
正数、负数、和小数：^(\-|\+)?\d+(\.\d+)?$
有两位小数的正实数：^[0-9]+(\.[0-9]{2})?$
有1~3位小数的正实数：^[0-9]+(\.[0-9]{1,3})?$
非零的正整数：^[1-9]\d\*$ 或 ^([1-9][0-9]\*){1,3}$ 或 ^\+?[1-9][0-9]\*$
非零的负整数：^\-[1-9][]0-9"\*$ 或 ^-[1-9]\d\*$
非负整数：^\d+$ 或 ^[1-9]\d\*|0$
非正整数：^-[1-9]\d\*|0$ 或 ^((-\d+)|(0+))$
非负浮点数：^\d+(\.\d+)?$ 或 ^[1-9]\d\*\.\d\*|0\.\d\*[1-9]\d\*|0?\.0+|0$
非正浮点数：^((-\d+(\.\d+)?)|(0+(\.0+)?))$ 或 ^(-([1-9]\d\*\.\d\*|0\.\d\*[1-
9]\d\*))|0?\.0+|0$
正浮点数：^[1-9]\d\*\.\d\*|0\.\d\*[1-9]\d\*$ 或 ^(([0-9]+\.[0-9]\*[1-9][0-9]\*)|
([0-9]\*[1-9][0-9]\*\.[0-9]+)|([0-9]\*[1-9][0-9]\*))$
- 负浮点数：^-([1-9]\d\*\.\d\*|0\.\d\*[1-9]\d\*)$ 或 ^(-(([0-9]+\.[0-9]\*[1-9][0-
9]\*)|([0-9]\*[1-9][0-9]\*\.[0-9]+)|([0-9]\*[1-9][0-9]\*)))$
- 浮点数：^(-?\d+)(\.\d+)?$ 或 ^-?([1-9]\d\*\.\d\*|0\.\d\*[1-9]\d\*|0?\.0+|0)$
```

校验字符的表达式

```
汉字：^[\u4e00-\u9fa5]{0,}$
英文和数字：^[A-Za-z0-9]+$ 或 ^[A-Za-z0-9]{4,40}$
长度为3-20的所有字符：^.{3,20}$
由26个英文字母组成的字符串：^[A-Za-z]+$
由26个大写英文字母组成的字符串：^[A-Z]+$
由26个小写英文字母组成的字符串：^[a-z]+$
由数字和26个英文字母组成的字符串：^[A-Za-z0-9]+$
由数字、26个英文字母或者下划线组成的字符串：^\w+$ 或 ^\w{3,20}$
中文、英文、数字包括下划线：^[\u4E00-\u9FA5A-Za-z0-9_]+$
中文、英文、数字但不包括下划线等符号：^[\u4E00-\u9FA5A-Za-z0-9]+$ 或 ^[\u4E00-\u9FA5A-
Za-z0-9]{2,20}$
可以输入含有^%&',;=?$\"等字符：**[^%&',;=?$\x22]+
禁止输入含有~的字符：[^~\x22]+
```

特殊需求表达式

Email地址：**^\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*$**
域名：**[a-zA-Z0-9][-a-zA-Z0-9]{0,62}(\.[a-zA-Z0-9][-a-zA-Z0-9]{0,62})+\.?**
InternetURL：**[a-zA-z]+://[^\s]* 或 ^http://([\w-]+\.)+[\w-]+(/[\w-./?%&=]*)?$**
手机号码：**^(13[0-9]|14[5|7]|15[0|1|2|3|4|5|6|7|8|9]|18[0|1|2|3|5|6|7|8|9])\d{8}$**
电话号码("XXX-XXXXXXX"、"XXXX-XXXXXXXX"、"XXX-XXXXXXX"、"XXX-XXXXXXXX"、"XXXXXXX"和"XXXXXXXX)：**^(\(\d{3,4}-)|\d{3.4}-)?\d{7,8}$**
国内电话号码(0511-4405222、021-87888822)：**\d{3}-\d{8}|\d{4}-\d{7}**
电话号码正则表达式（支持手机号码，3-4位区号，7-8位直播号码，1一4位分机号）：**((\d{11})|^((\d{7,8})|(\d{4}|\d{3})-(\d{7,8})|(\d{4}|\d{3})-(\d{7,8})-(\d{4}|\d{3}|\d{2}|\d{1})|(\d{7,8})-(\d{4}|\d{3}|\d{2}|\d{1}))$)**
身份证号(15位、18位数字)，最后一位是校验位，可能为数字或字符X：**(^\d{15}$)|(^\d{18}$)|(^\d{17}(\d|X|x)$)**
帐号是否合法(字母开头，允许5-16字节，允许字母数字下划线)：**^[a-zA-Z][a-zA-Z0-9_]{4,15}$**
密码(以字母开头，长度在6~18之间，只能包含字母、数字和下划线)：**^[a-zA-Z]\w{5,17}$**
强密码(必须包含大小写字母和数字的组合，不能使用特殊字符，长度在 8-10 之间)：**^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])[a-zA-Z0-9]{8,10}$**
强密码(必须包含大小写字母和数字的组合，可以使用特殊字符，长度在8-10之间)：**^(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,10}$**
日期格式：**^\d{4}-\d{1,2}-\d{1,2}**
一年的12个月(01～09和1～12)：**^(0?[1-9]|1[0-2])$**
一个月的31天(01～09和1～31)：**^((0?[1-9])|((1|2)[0-9])|30|31)$**

在这里我们需要注意的是**^[\u4e00-\u9fa5]{0,}$**，这个是汉字的编码范围，在爬虫过程中经常会有一个爬目录的操作^http://([\w-]+.)+[\w-]+(/[\w-./?%&=]*)?$对于**URL**编码的正则匹配同样很重要。

3 python爬虫实战

```
from typing import Pattern
import requests
import re
for i in range(5200804,5719275):
    url=f'https://www.1biquge.com/8/8145/{i}.html'
    a=requests.get(url)
    print(url)
    a.encoding="utf-8"
    file=open("./源代码.txt","w",encoding="utf-8")
    file.write(a.text)
    file.close()
    f = open("源代码.txt", encoding="utf-8")
    file_handle = open('小说文本.txt', mode='a',encoding="utf-8")
    line = f.read()
    newa =
re.compile(u'[\n\s*\r\u3002\uff1b\uff0c\uff1a\u201c\u201d\uff08\uff09\u3001\uff1f\u300a\u300b\u4e00-\u9fa5]')
    b=newa.findall(line)
    n=len(b)
    for i in range(1,n):
        file_handle.write(b[i-1])
    line = f.readline()
    file_handle.close()
    f.close()
```

这个实际上是一个还没有弄完的脚本，因为我爬取的是笔趣阁的小说，毕竟笔趣阁本身就是盗版网站加密也没那么麻烦，还真让我爬出来了一些信息。但是在爬取过程在还是注意到开发者是做了简易的防爬虫操作的，在一定的章节后，他随机的增加了一个四位数字，这导致了以上代码在进行无意义的空跑，但是还是发现这个代码可以应用于同网站少数小说的爬取，猜测是当时网站还没有做防爬虫的操作。

之后在类似的爬取其他内容过程中，发现有些网站爬虫根本无法爬取。花了一些时间研究，找到结论，我们一般写的爬虫会默认告诉服务器自己发送了一个Python爬取请求，而一般网站是不允许被爬虫访问的，估计是因为会涉及到企业根本吧。最后，我们可以通过更改User-Agent字段就可以轻易骗过该网站。

我们可以使用**fake_useragent**进行User-Agent字段的欺骗，这里就不再展示

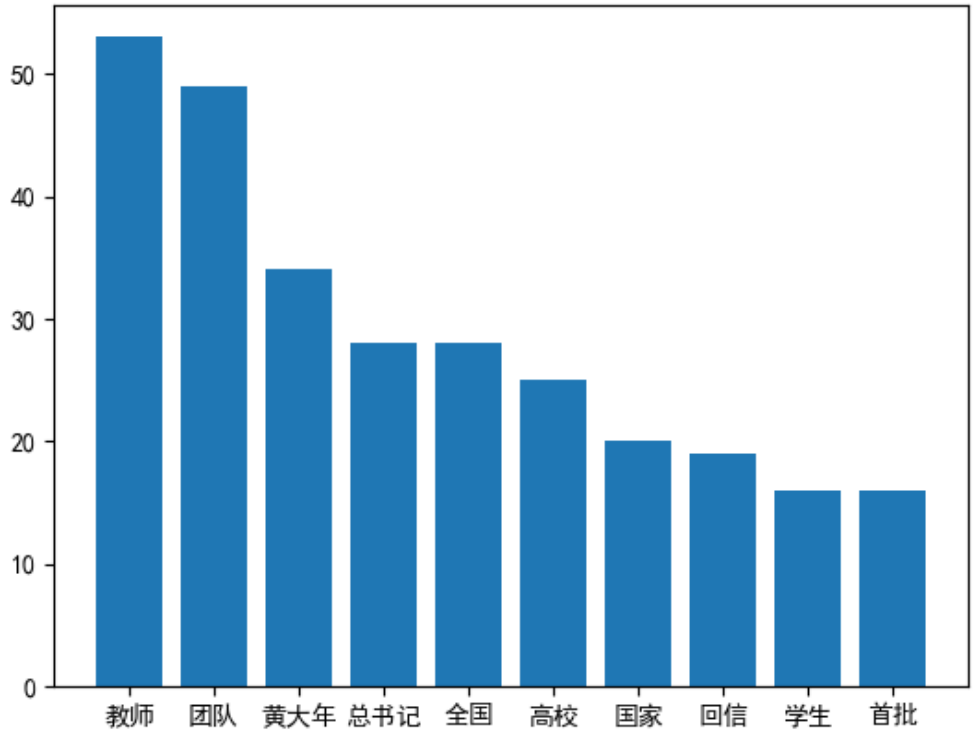
最后提示一句**《刑法》第217条第1项规定，以营利为目的，未经著作权人许可，复制发行其文字作品、音乐、电影、电视、录像作品、计算机软件及其他作品，处以有期徒刑，并处或单 处罚金。** 大家未来对爬虫技术掌握爬虫技术之后，能从网络上爬取一些东西，但绝对不要违法。

5.0 问题导入

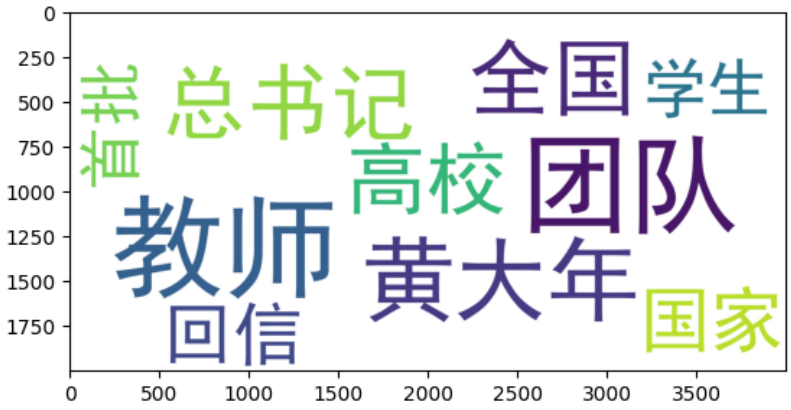
> 使用 wordcloud 和 matplotlib.pyplot 方法对“222.txt”中的分词制作 分词云图，并保存为“分词云图 1.png”。
>
> 使用“222.txt”中出现频率最高的前 10 个词语，制作条形图，X 轴对应 10 个词语，y 轴对应每个词语的频率值，保存成“条形图 2.png”。

上面是上一次实验课，于刊老师的问题，下面是我解决这两道题目的方式，

```
import matplotlib
import pandas as pd
import matplotlib.pyplot as plt
#课本提示我们汉语要设置字体，不然乱码
matplotlib.rcParams['font.family']='SimHei'
a=pd.read_table('333.txt',encoding='utf-8',sep='       ',engine='python')
#这个是之前李师姐讲到的读写操作
df=pd.DataFrame(a)
#print (a)
#print(df['词语'])
#测试用数据可以忽略
x=df['词语']
y=df['次数']
figure=plt.Figure()
plt.bar(x,y)
plt.show()
```




```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# 要先pip install wordcloud
from wordcloud import WordCloud
a=pd.read_table('333.txt',encoding='utf-8',sep='       ',engine='python')
df=pd.DataFrame(a)
#fit_word函数，接受字典类型，其他类型会报错
dic=dict(zip(df['词语'],df['次数']))
wordcloud = WordCloud(font_path='simhei.ttf',background_color="white",width
=4000,height= 2000,margin= 10 ).fit_words(dic)
plt.imshow(wordcloud)
# 显示
plt.show()
```



这里使用了**matplotlib库**和**wordcloud库**两个库实现，

其实做 Python的数据可视化，可以使用的库分别是**Matplotlib**，**Seaborn**，**Bokeh**，**Plotly**，**Pyecharts**等。

我们课本主要将的是**Matplotlib**库，我其实没有在互联网上找到matplotlib库的官方中文文档的，关于这些自学起来实际上是看的浏览器的翻译，如有错误还请海涵……

5.1 matplotlib库基础

拓展：默认值相关

```
fig=plt.figure()
ax=fig.add_axes([0.1,0.1,0.8,0.8])
plt.show()
```

fig.add_axes([])里面的参数分别指的距离左边，下边，坐标轴宽度，坐标轴高度，范围是(0, 1)等价于**plt.axes()**

ax=fig.add_axes([0.1,0.1,0.8,0.8])语句表示的是在画布中，坐标轴距离画布左边0.1倍的位置，距离下边0.1倍的位置，确定了这两个位置后，坐标轴的整体宽度和高度占0.8倍的大小，换句话说，距离右边和上边0.9~~-(0.1+0.8)~~倍。执行如下代码，会生成一张带有坐标轴的白色，两个语句实际上都是**默认值。**

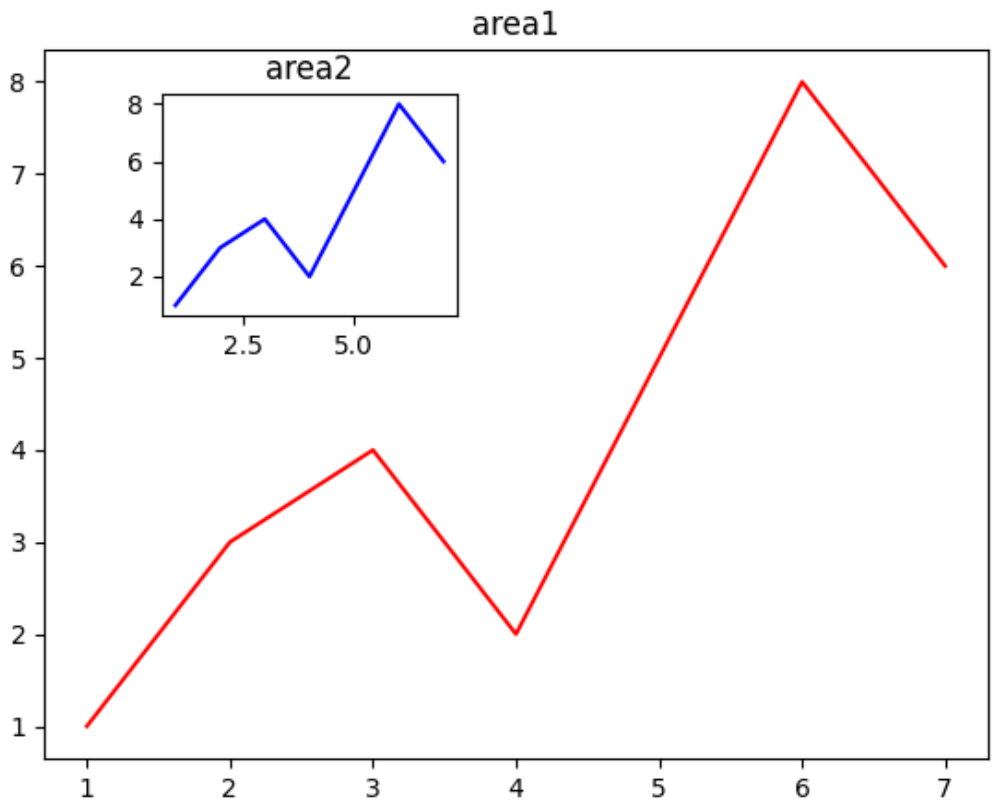
```
pyplot.grid(b=None, which='major', axis='both', **kwargs)
##**kwargs 允许你将不定长度的键值对，作为参数传递给一个函数。 如果你想要在一个函数里处理带名字的参数，你应该使用**kwargs
#以下是一个实例：
plt.grid(True, linestyle = "--",color = "gray", linewidth = "0.5",axis = 'both')

# 显示网格，这个参数默认为forse
# linestyle：线型
# color：颜色
# linewidth：宽度
# axis：x，y，both，显示x/y/两者的格网
pyplot.title(label, fontdict=None, loc=None, pad=None, *, y=None,**kwargs)
```

```
plt.tick_params(bottom='on',top='off',left='on',right='off')
# 显示刻度的那根轴线，凸出来的地方。设置为off时都不显示。默认为全显示
#plt.axis('off')
# 关闭坐标轴
```

如果想要了解这些具体的参数还是推荐去看下那个纯英文的matplotlib库手册

显示创建画布和坐标轴的好处是让我们对绘图过程有了完全的控制权（比如可以指定在什么地方绘图），而且绘图的逻辑更强 。

```
import numpy as np
import matplotlib.pyplot as plt
# 新建figure
fig = plt.figure()
# 定义数据
x = [1, 2, 3, 4, 5, 6, 7]
y = [1, 3, 4, 2, 5, 8, 6]
# 新建区域ax1
# figure的百分比,从figure 10%的位置开始绘制, 宽高是figure的80%
left, bottom, width, height = 0.1, 0.1, 0.8, 0.8
# 获得绘制的句柄
ax1 = fig.add_axes([left, bottom, width, height])
ax1.plot(x, y, 'r')
ax1.set_title('area1')
# 新增区域ax2,嵌套在ax1内
left, bottom, width, height = 0.2, 0.6, 0.25, 0.25
# 获得绘制的句柄
ax2 = fig.add_axes([left, bottom, width, height])
ax2.plot(x, y, 'b')
ax2.set_title('area2')
plt.show()
```



> 补充知识点，在课本表5—2中出现的函数均有add_函数名(参数)和plt.函数名(参数)两种形式，实际使用中记住一种就行。

5.2 plt.plot()绘图

plt.plot(x, y, c="b",ls="-", lw=2, label="文本串")
其中，第二个简单的格式是最常用的格式，参数x表示指定的x轴上的数值，参数y表示指定的y轴上的数值，参数c指定绘制线条的颜色，参数ls指定的折线图的线条风格，参数lw指定绘制的线条宽度，参数label指定标记图内容的标签文本。

plt.plot()函数中可以明确地给出了x,y的值，然后得到相应的(x,y)坐标点，把这些点再连接成线。当**plt.plot()**函数也允许只给出y值的情况，这时系统给自动将x值指定为range(len(y))，然后根据x,y的值进行点连线。

> len() 函数返回对象中项目的数量。

```
import matplotlib.pyplot as plt
#只给出y值，画出相应图像
plt.plot([0, 1, 2])
plt.show()
```

在5.2plt.plot()绘图这一节里，我们以实例5-1为例，

```
import matplotlib.pyplot as plt
fig=plt.figure()
ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])
x=[1,2,3,4,5,6]
y=[3,5,1,7,9,12]
plt.plot(x,y)
plt.show()
#带有默认设置的程序
import matplotlib.pyplot as plt
x=[1,2,3,4,5,6]
y=[3,5,1,7,9,12]
plt.plot(x,y)
plt.show()
#去除默认值的程序
```

再pycharm环境中允许，生成的图片实际上是一模一样的，这是因为在使用了plt.plot()语句，会默认创建一个figure对象和axes坐标系。

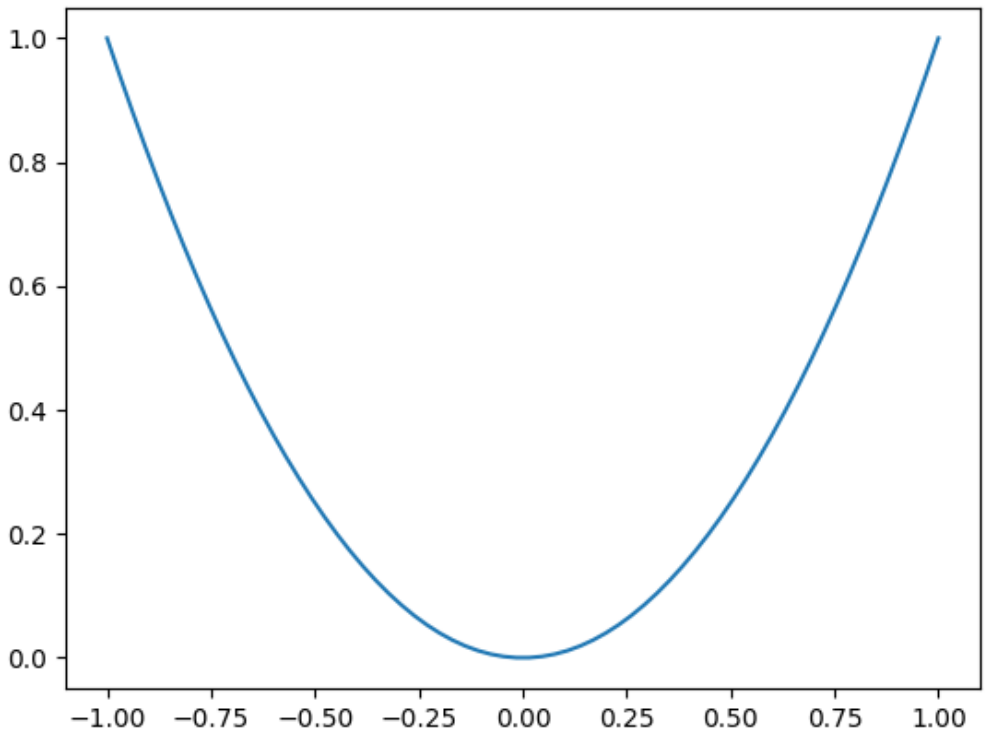
例5-2

```
import matplotlib.pyplot as plt
x = [1,2,3,4,5]
y1 = [1,2,3,4,5]
y2 = [2,4,6,8,10]
y3 = [3,6,9,12,15]
plt.plot(x, y1, color="g",linestyle="-
",marker="o",markerfacecolor="g",markersize=20)
plt.plot(x, y2,
color="r",linestyle=":",marker="D",markerfacecolor="r",markersize=20)
plt.plot(x, y3,
color="b",linestyle="-.",marker="s",markerfacecolor="b",markersize=20)
plt.show()
```

引例：画一张二次曲线

```
import matplotlib.pyplot as plt
import numpy as np
#从-1到1之间等间隔采66个数.也就是说所画出来的图形是66个点连接得来的
#注意：如果点数过小的话会导致画出来二次函数图像不平滑
x = np.linspace(-1, 1,66)
#linspace(start,stop,num,endpoint= True)函数在[start,stop]范围内计算返回num个均匀间隔的点，可以通过endpoint确认是否包含间隔点
# 绘制x^2函数的图像
y = x**2
plt.plot(x, y)
plt.show()
```

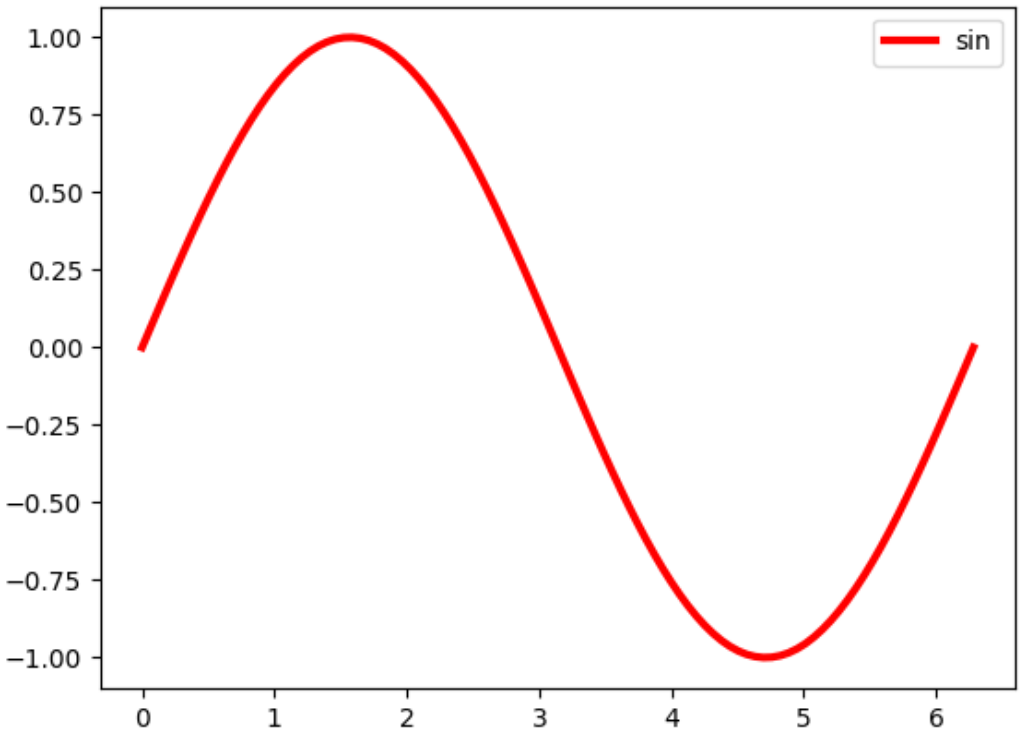
```
numpy.random.randint(low, high=None, size=None, dtype='l')
#返回一个随机整型数，范围从低（包括）到高（不包括），即[low, high)。
#如果没有写参数high的值，则返回[0,low)的值。
#输出随机数的尺寸，比如size = (m * n* k)则输出同规模即m * n* k个随机数。默认是None的，仅仅返回满足要求的单一随机数。
```



我们可以看到当图像间隔足够小的时候，折线图可以模拟出二次曲线的形式，类似的方法可以实现正弦函数的产生

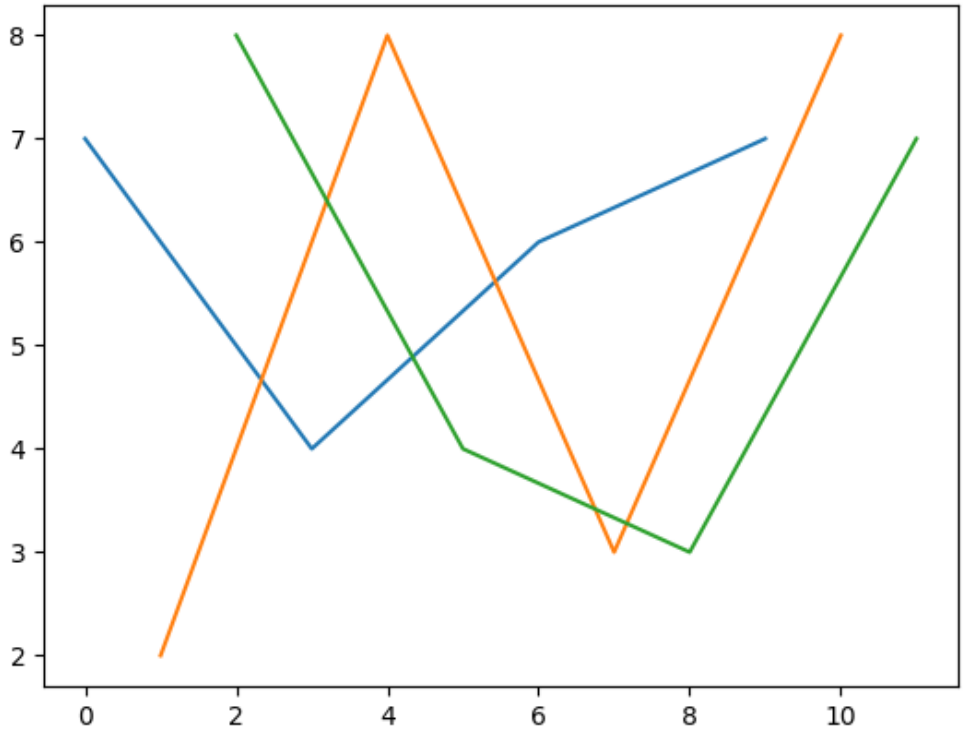
例5-4

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0,2*np.pi, 100)
y = np.sin(x)
plt.plot(x, y,c='r',ls="-", lw=3, label="sin")
plt.legend()
plt.show()
```



例5-5

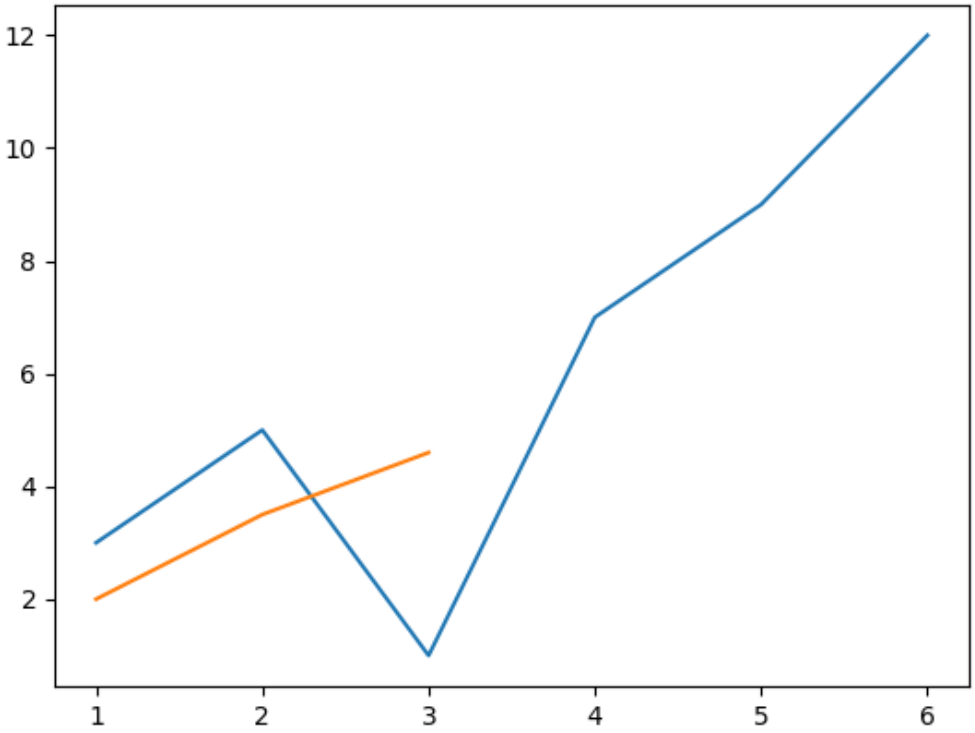
```
import numpy as np
import matplotlib.pyplot as plt
t= np.arange(12)
x= t.reshape((4,3))#将arange(12)生成的12个数展成[4,3]的二维数组
y=np.random.randint(2,9,size=[4,3])
plt.plot(x,y)
```



5.3 划分子图

例5-7:

```
import matplotlib.pyplot as plt
x = [1,2,3,4,5,6]
y = [3,5,1,7,9,12]
plt.plot(x,y)
x = [1,2,3]
y = [2.0,3.5,4.6]
plt.plot(x,y)
plt.show()
```



在这个情况下，两个折线是被默认存储到同一个画布上，然而我们在中间多添加一个plt.show()就会是分别位于一张画布上。在这个实例中也能看出来在，两张图片的X轴范围不同的情况下，放在同一张画布下是不容易显示的.

为此我们要进行划分子图

格式一：**plt.subplot(nrows,ncols,plot_number)**将一个画布分成"行×列"个部分，其中nrows表示要划分的行数，ncols表示要划分的列数，plot_number表示当前子图区。

格式二：

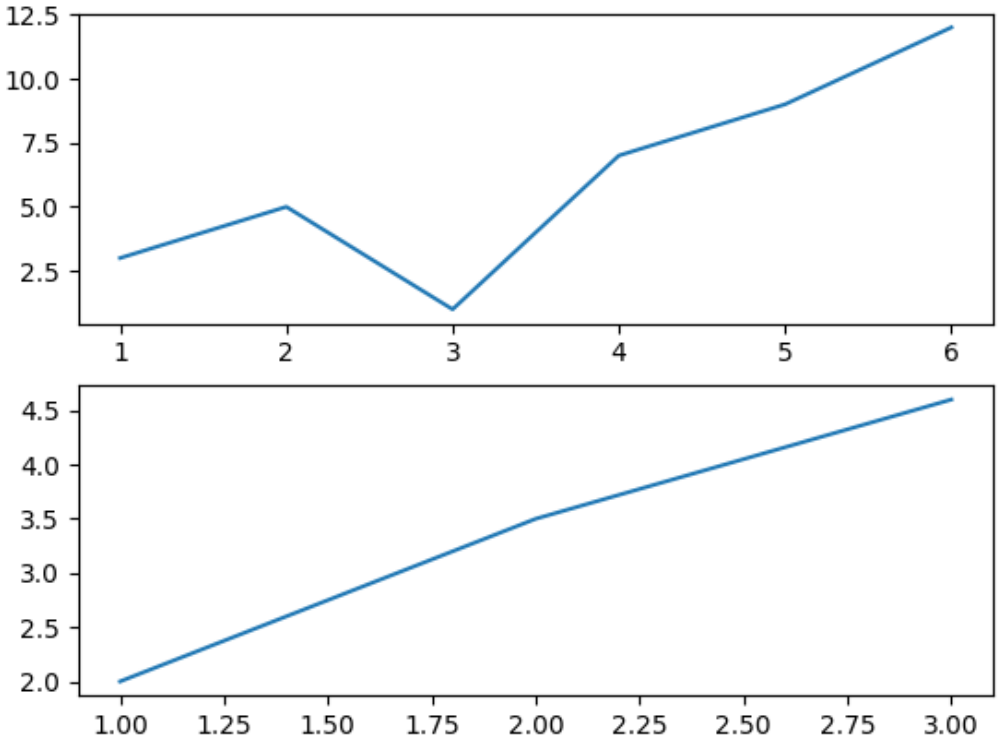
```
figure = plt.figure()
figure.add_subplot(nrows,ncols,plot_number)
```

实例5-8 : 一个figure对象画风两个子图实例

```
import matplotlib.pyplot as plt
x = [1,2,3,4,5,6]
y = [3,5,1,7,9,12]
figure = plt.figure()
axes1 = figure.add_subplot(2,1,1)
plt.plot(x,y)
x = [1,2,3]
y = [2.0,3.5,4.6]
axes2 = figure.add_subplot(2,1,2)
plt.plot(x,y)
plt.show()
```

以格式一重写代码：

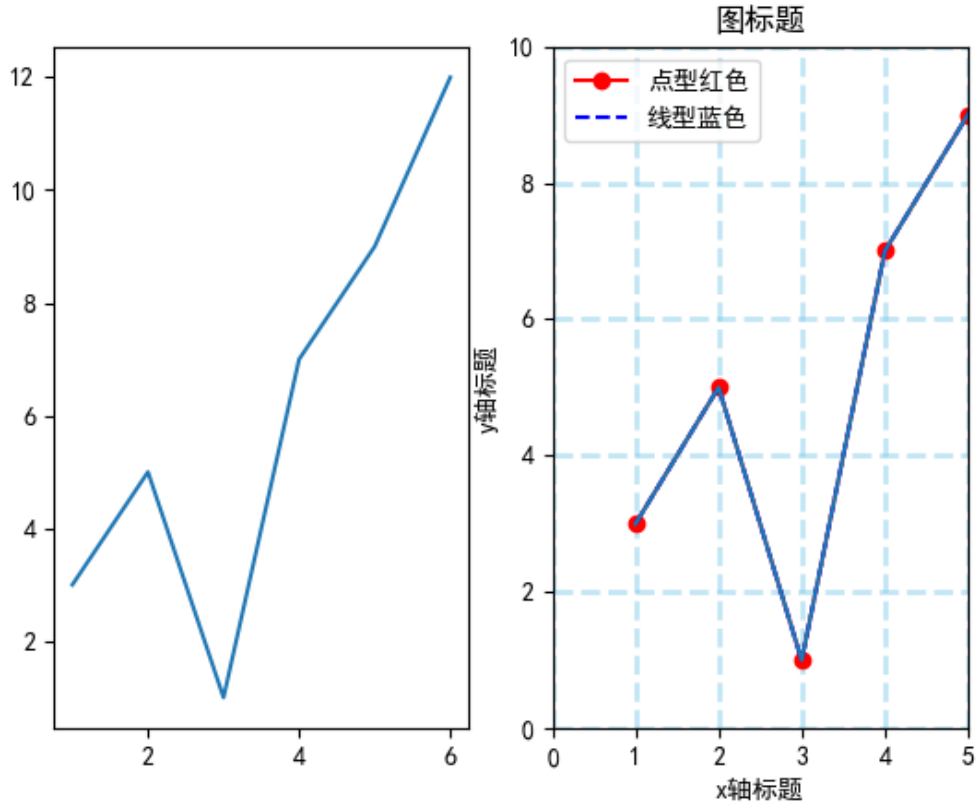
```
import matplotlib.pyplot as plt
x = [1,2,3,4,5,6]
y = [3,5,1,7,9,12]
figure = plt.figure()
plt.subplot(2,1,1)
plt.plot(x,y)
#补充一点课本上出现了两次的plt.show这可能是教材编写老师的失误，因为两编plt.show是得不到目标格式的
x = [1,2,3]
y = [2.0,3.5,4.6]
plt.subplot(2,1,2)
plt.plot(x,y)
plt.show()
```



例5-9：设置绘图的各种参数实例

```
import matplotlib.pyplot as plt
#以下两句代码的作用是显示汉字图标题
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False
#创建figure对象，划分子图
figure = plt.figure()
axes1 = figure.add_subplot(1,2,1)
axes2 = figure.add_subplot(1,2,2)
#指定x,y列表值
x = [1,2,3,4,5,6]
y = [3,5,1,7,9,12]
axes1.plot(x,y)
#设置子图2的基本元素
axes2.set_title('图标题')
axes2.set_xlabel('x轴标题')
```

```
axes2.set_ylabel('y轴标题')
#设置坐标轴范围
axes2.set_xlim(0,5)
axes2.set_ylim(0,10)
#设置绘制的线型和颜色及标注内容
plot1=axes2.plot(x,y,marker='o',color='r',label='点型红色')
plot2=axes2.plot(x,y,linestyle='--',color='b',label='线型蓝色')
axes2.legend(loc='upper left') #在指定位置显示标注
#显示网格绘制子图2
axes2.grid(b=True,which='major',axis='both',alpha=
0.5,color='skyblue',linestyle='--',linewidth=2)
axes2.plot(x,y)
plt.savefig('tu1.jpg',dpi=300) #保存图形
plt.show()
```



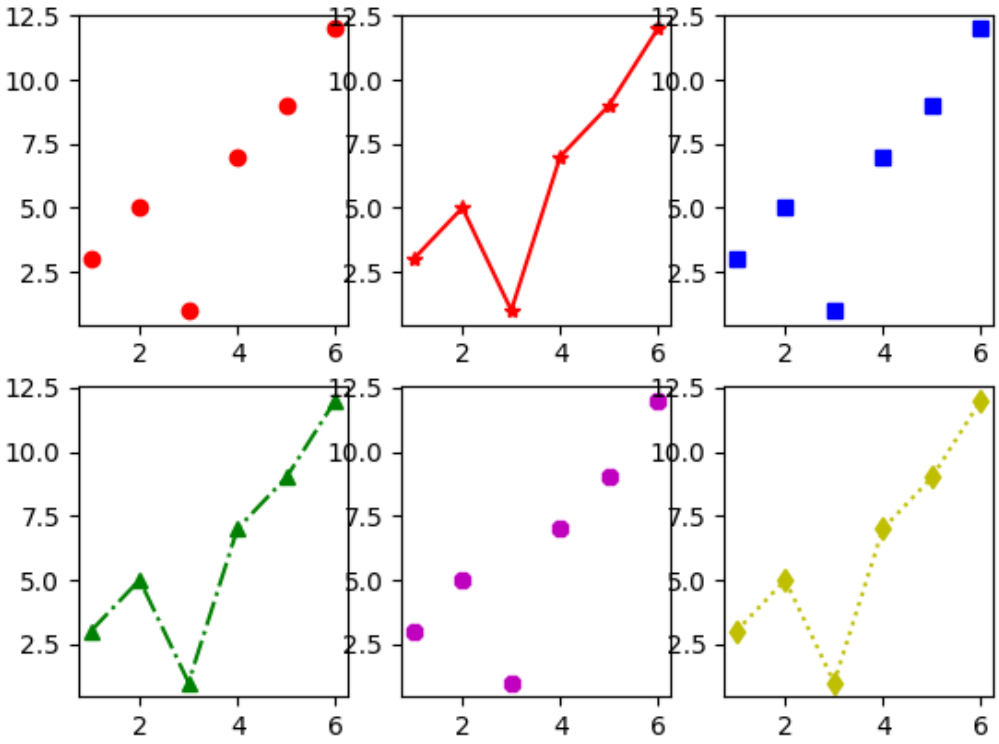
在python绘图中，不能直接显示汉字，但是我们经常希望绘制的图形中的标题及题注等能以汉字的形式显示，可以在代码中倒入库语句后面添加一下两句代码可以显示汉字**plt.rcParams['font.sans-serif']=['SimHei']**与**plt.rcParams['axes.unicode_minus']=False**这两个一个是设置字符显示，一个是设置字体，**SimHei**这个实际上就是我们生活中常用到的**黑体**

例5-10:将一个figure对象划分为六个子图，分别绘制不同线型、不同颜色和标记的线条。

```
import matplotlib.pyplot as plt
x = [1,2,3,4,5,6]
y = [3,5,1,7,9,12]
figure = plt.figure()
axes1 = figure.add_subplot(2,3,1)
axes2 = figure.add_subplot(2,3,2)
axes3 = figure.add_subplot(2,3,3)
axes4 = figure.add_subplot(2,3,4)
axes5 = figure.add_subplot(2,3,5)
axes6 = figure.add_subplot(2,3,6)
```

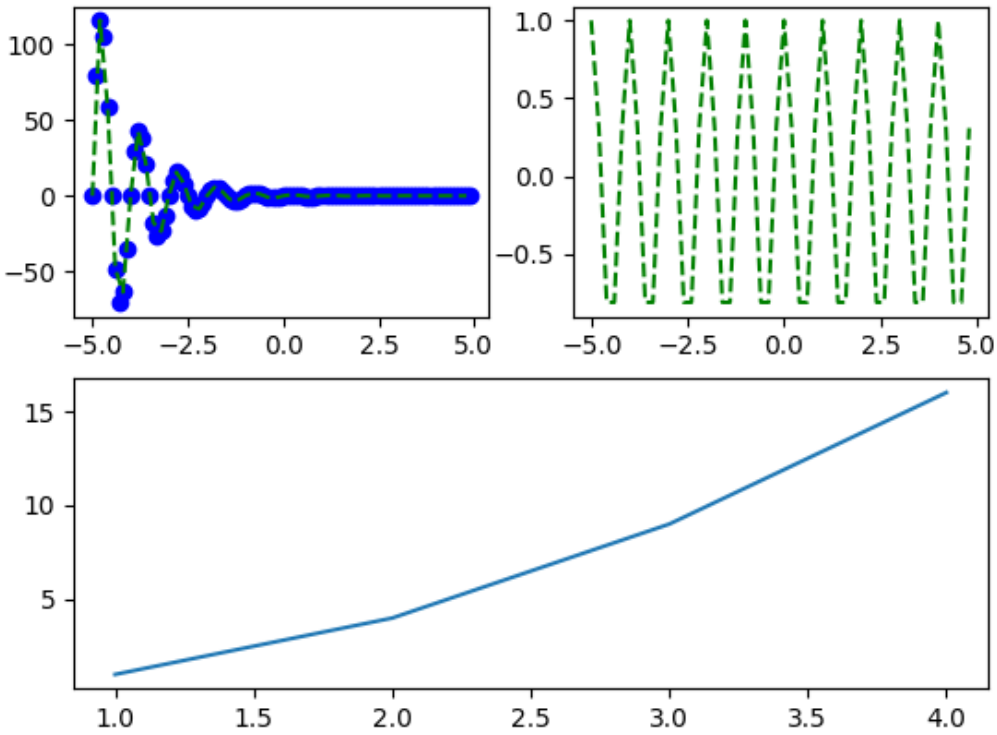
```
axes1.plot(x,y, 'ro') # 红色圆点
axes2.plot(x,y, 'r-*') #红色星花直线
axes3.plot(x,y, 'bs') #蓝色方块
#axes4.plot(x,y, 'g^-.') #绿色三角点划线
axes4.plot(x,y, '^g-.')
axes5.plot(x,y, 'm8')  #洋红八边形
axes6.plot(x,y, 'yd:') #黄色小菱形虚线
plt.show()
```

在绘图时需要指定线条的线型，标记的符号及线条的颜色的参数值写进同一个大括号，而且对三个参数的前后顺序没有要求。



拓展三张图如何划分?

```
import matplotlib.pyplot as plt
import numpy as np
def f(t):
    return np.exp(-t) * np.sin(2 * np.pi * t)
if __name__ == '__main__':
    t1 = np.arange(-5, 5, 0.1)
    t2 = np.arange(-5, 5, 0.2)
    plt.figure()
    plt.subplot(221)
    plt.plot(t1, f(t1), 'bo', t2, f(t2), 'g--')
    plt.subplot(222)
    plt.plot(t2, np.cos(2 * np.pi * t2), 'g--')
    plt.subplot(212)
    plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
    plt.show()
```



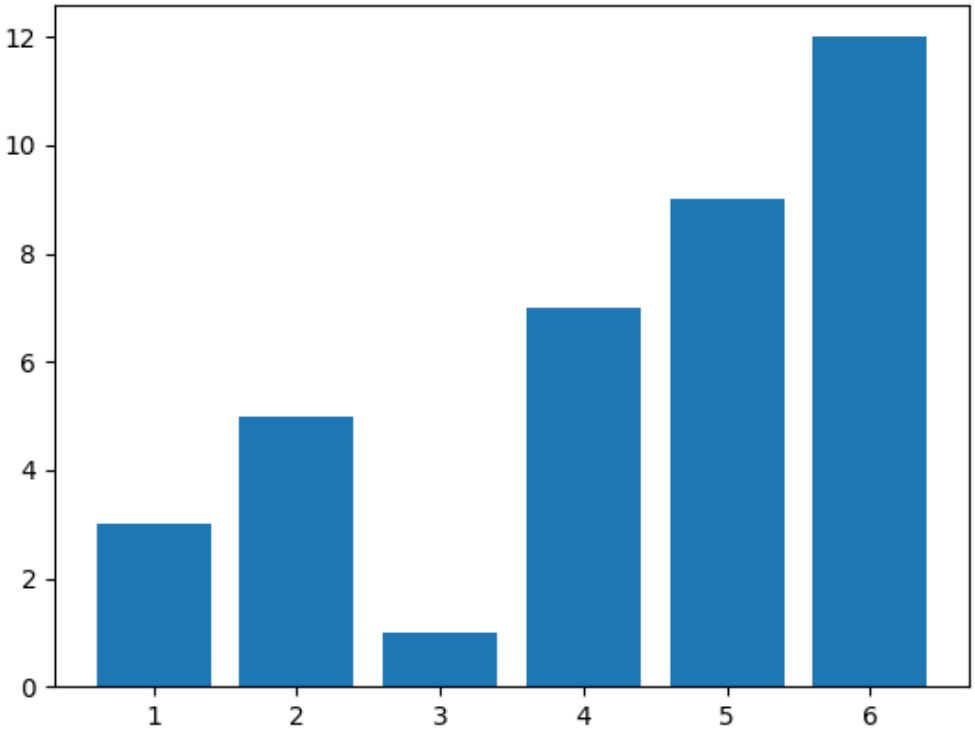
5.4: 条形图

条形图是数据分析中使用较多的数据可视化方式之一。在条形图中可以非常直观地通过位置比较数值大小。因为在条形图中条的高度就是数值，所以一眼就可以看出数值的高底。

bar(x, height, width=0.8, bottom=None, ..., align='center')

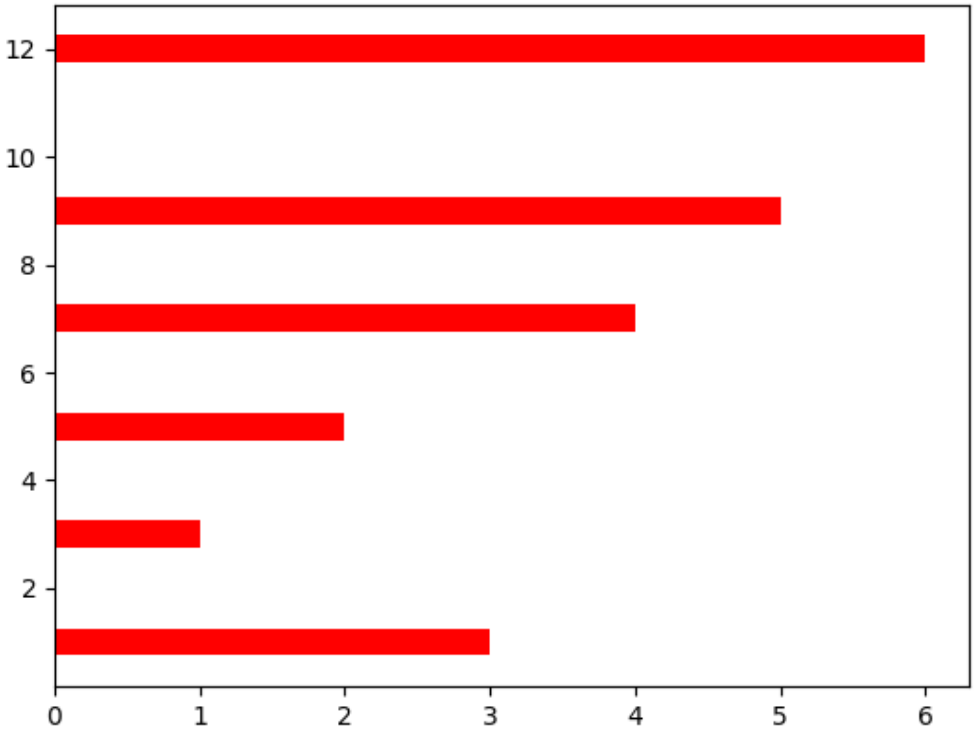
例5-11 :简单条形图

```
import matplotlib.pyplot as plt
x = [1,2,3,4,5,6]
y = [3,5,1,7,9,12]
figure = plt.figure()
plt.bar(x, y) #默认颜色，默认宽度为0.8
plt.show()
```



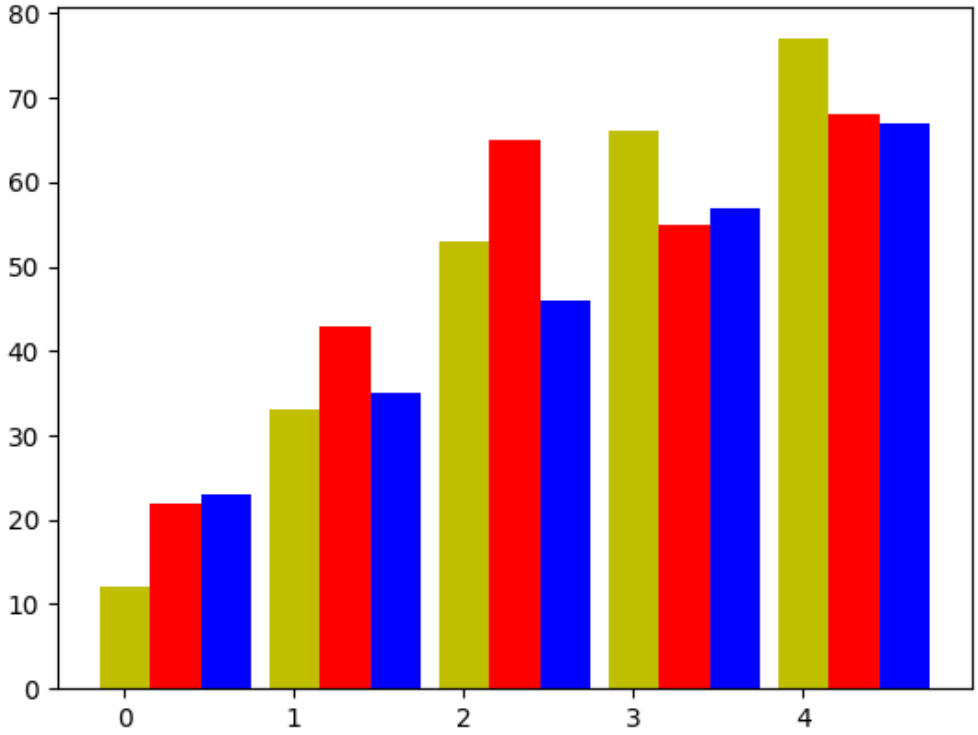
例5-12 : 横向条形图。

```
import matplotlib.pyplot as plt
x = [1,2,3,4,5,6]
y = [3,5,1,7,9,12]
# x指定起始位置从0开始， bottom指定水平条起始位置为左侧， height指定绘制的水平条的宽度，
width 指定绘制的水平条的长度。orientation指定要绘制的是水平条，color指定绘制的条形为红色。
plt.bar(x=0, bottom=y, height=0.5, width=x, orientation="horizontal",color='r')
plt.show()
```



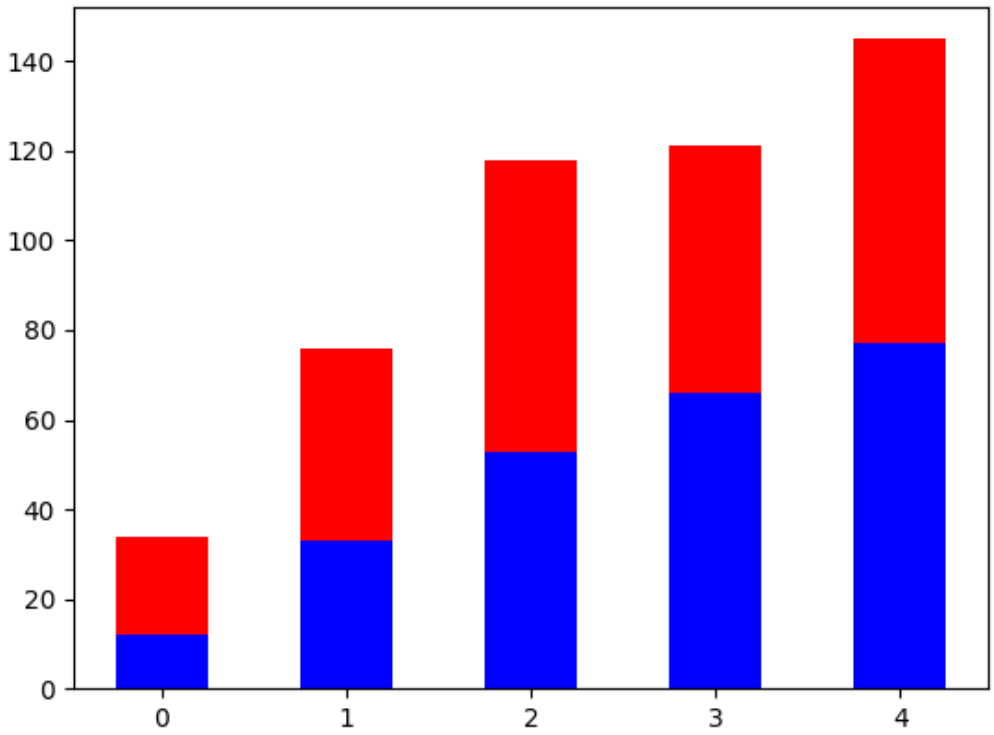
5.4.2 多组条形图

```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(5)
y1= [12, 33, 53, 66,77]
y2 = [22, 43, 65, 55,68]
y3=[23,35,46,57,67]
bar_width = 0.3
plt.bar(x, y1, bar_width,color='y')
plt.bar(x+bar_width,y2, bar_width, align="center",color='r')
plt.bar(x+bar_width+bar_width,y3, bar_width, align="center",color='b')
plt.show()
```

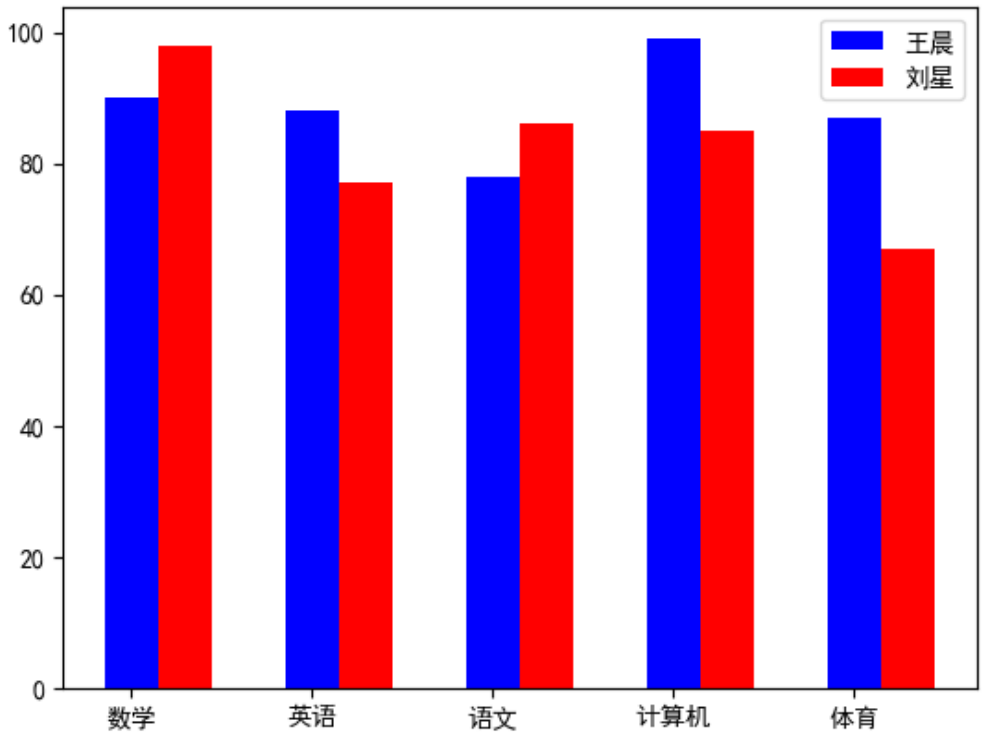


在绘制多组条形图进行对比的时候需要注意一点，bar_width的范围是(0,1)没错，但是多组图表需要保证几组图表的范围之和不超过1，以这个题为例，当bar_width设置为0.4三个表得到总和就超过了1，会导致重叠的发生。

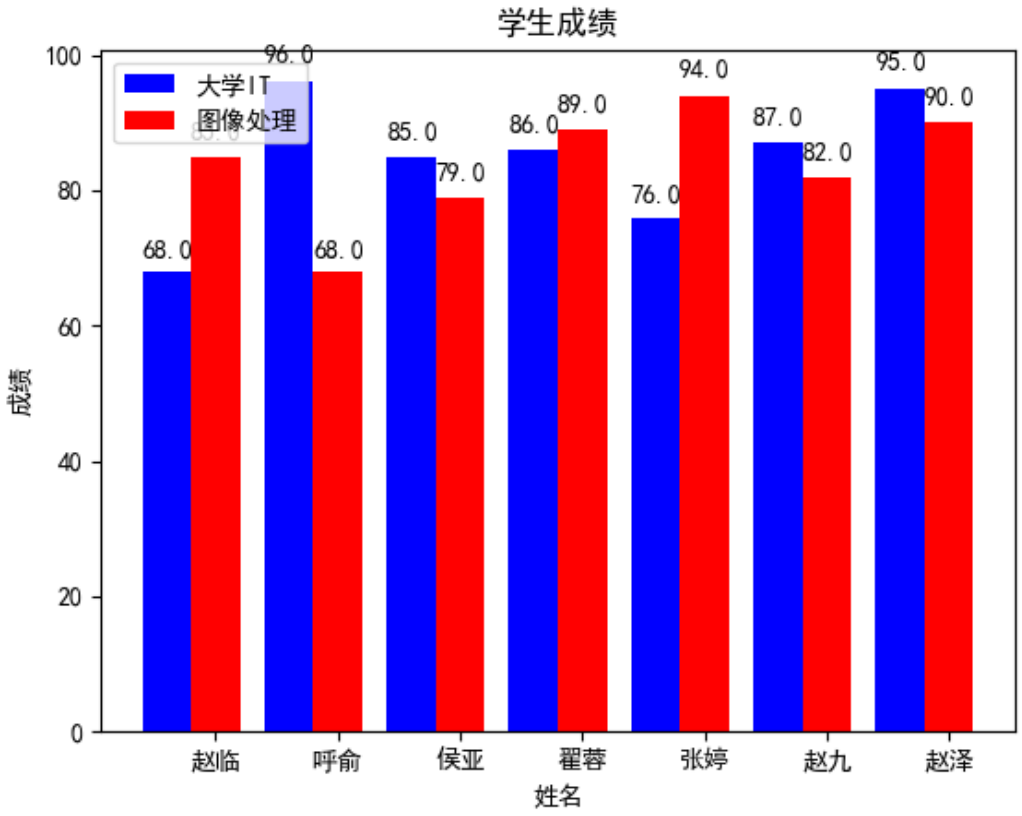
```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(5)
y1= [12, 33, 53, 66,77]
y2 = [22, 43, 65, 55,68]
bar_width = 0.5
plt.bar(x, y1, bar_width,color='b')
plt.bar(x, y2, bar_width, bottom=y1,color='r')
plt.show()
```




```
import numpy as np
import matplotlib.pyplot as plt
#字体设置为黑体
matplotlib.rcParams['font.family'] = 'SimHei'
x = np.arange(5)
y1=[90, 88, 78, 99, 87]
y2=[98, 77, 86, 85, 67]
bar_width = 0.3
str1 = ("数学","英语","语文","计算机","体育")
# 绘图
plt.bar(x, height=y1, width=bar_width, label="王晨", tick_label=str1,color='b')
#tick_label下标的标签，label直接是标签
plt.bar(x+bar_width, height=y2, width=bar_width, label="刘
星",color='r',align="center")
#添加图例
plt.legend()
plt.show()
```




```
import matplotlib.pyplot as plt
# 显示汉字设置
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
# 定义函数来显示柱状上的数值
def autolabel(rects):
    for rect in rects:
        height = rect.get_height()
        plt.text(rect.get_x() + rect.get_width() / 2. - 0.2, 1.03 * height, '%s'
% float(height))
if __name__ == '__main__':
    l1 = [68, 96, 85, 86, 76, 87, 95]
    l2 = [85, 68, 79, 89, 94, 82, 90]
    name = ['赵临', '呼俞', '侯亚', '翟蓉', '张婷', '赵九', '赵泽']
    total_width, n = 0.8, 2
    width = total_width / n
    x = [0, 1, 2, 3, 4, 5, 6]
    a = plt.bar(x, l1, width=width, label='大学IT', fc='b')
    for i in range(len(x)):
        x[i] = x[i] + width
    b = plt.bar(x, l2, width=width, label='图像处理', tick_label=name, fc='r')
    autolabel(a)
    autolabel(b)
    plt.xlabel('姓名')
    plt.ylabel('成绩')
    plt.title('学生成绩')
    plt.legend()
    plt.show()
```



拓展图例的位置

```
在legend的参数中， loc参数设置图例的显示位置的：
 plt.legend([l1, l2], ['first', 'second'], loc = 'upper right')#其中，loc表示位置的；
'best'         : 0, (only implemented for axes legends)(自适应方式)
'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
#实际上bbox_to_anchor()函数的范围是(0,1),也就是将整张图看做1 * 1大小
#对于两个元素的bbox_to_anchor()，也就是(x,y)，这个参数是代表了lengend_box的起点，并且是有后面的loc决定 的。
#例如，设置(0.5,0.5)，loc='center'，那么代表lengend_box的中心点（center）坐标是(0.5,0.5)
```

5.5 散点图

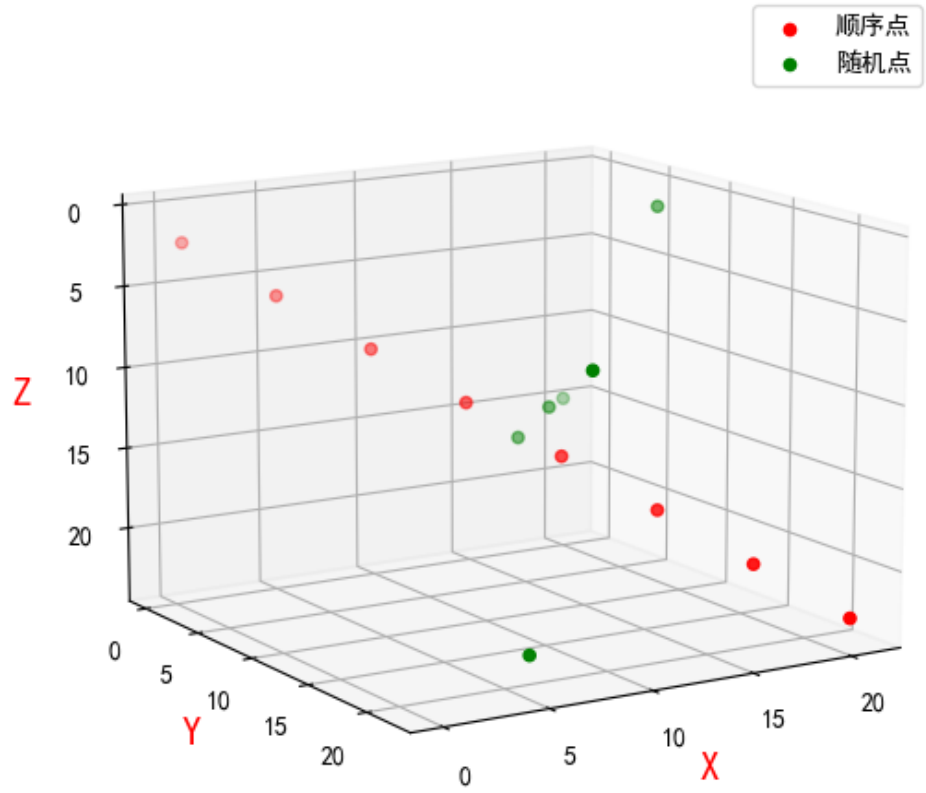
在实际数据分析应用中，散点图是一种常用于观测数据的相关性的数据可视化方式。数据的相关性通常有正相关、负相关及不相关。

```
plt.scatter(x, y, s, c ,marker, alpha)
```

```
import matplotlib.pyplot as plt
x = [1, 2,3,4,5,6]
y = [33, 38,55,59, 66, 77]
plt.scatter(x, y,c='r')
plt.show()
```

拓展：三维散点图

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D  # 空间三维画图
# 数据 1
data1 = np.arange(24).reshape((8, 3))
x1 = data1[:, 0]  # [ 0  3  6  9 12 15 18 21]
y1 = data1[:, 1]  # [ 1  4  7 10 13 16 19 22]
z1 = data1[:, 2]  # [ 2  5  8 11 14 17 20 23]
# 数据 2
data2 = np.random.randint(0, 23, (6, 3))
x2 = data2[:, 0]
y2 = data2[:, 1]
z2 = data2[:, 2]
# 绘制散点图
fig = plt.figure()
ax = Axes3D(fig)
ax.scatter(x1, y1, z1, c='r', label='顺序点')
ax.scatter(x2, y2, z2, c='g', label='随机点')
# 绘制图例
ax.legend(loc='best')
# 添加坐标轴(顺序是Z, Y, X)
ax.set_zlabel('Z', fontdict={'size': 15, 'color': 'red'})
ax.set_ylabel('Y', fontdict={'size': 15, 'color': 'red'})
ax.set_xlabel('X', fontdict={'size': 15, 'color': 'red'})
# 展示
plt.show()
```



拓展三维散点图，就要讲到matplotlib库里的另一个模块，mpl_toolkits.mplot3d，这个本身就来源于matplotlib不需要再行安装。

```
ax.scatter(xs, ys, zs, s=20, c=None, depthshade=True, *args, *kwargs)
```

- xs,ys,zs：输入数据；
- s:scatter点的尺寸
- c:颜色，如c = 'r'就是红色；
- depthshase:透明化，True为透明，默认为True，False为不透明
- *args等为扩展变量，如maker = 'o'，则scatter结果为‘o’的形状

我们可以看到同一个函数的参数列表基本一致。三维散点图与二维散点图的去别在于需要使用Axes3D对象或使用projection ='3d'关键字的任何其他轴一样创建。 创建一个新的matplotlib.figure.Figure并为其添加一个类型为Axes3D的新轴。

5 . 6 饼图

由于我在制作过程中发现图片使用过多真的太大了，所以后面的代码除了我重点想为大家拓展的，其他会直接进行演示

饼图是一种将各项数据的大小与各项数据的总和的比例显示在一张饼中的数据可视化方式。

```
plt.pie(x, explode=None, labels=None, colors=None, autopct=None,
pctdistance=0.6, shadow=False, labeldistance=1.1, startangle=None, radius=None,
counterclock=True, wedgeprops=None, textprops=None, center=(0, 0), frame=False)
```

```
import matplotlib.pyplot as plt
#添加以下两句代码，显示汉字标题
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False
# 数据
labels = ["数学", "语文", "英语", "物理","化学"]
cj = [91, 88, 89, 89,92]
# 画图
plt.pie(x=cj, labels=labels)
plt.show()
```

```
import matplotlib.pyplot as plt
#添加以下两句代码，显示汉字标题
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False
# 数据
labels = ["数学", "语文", "英语", "物理","化学"]
cj = [91, 88, 89, 89,92]
tuchu= [0, 0.1, 0, 0,0]
# 画图
plt.pie(x=cj, labels=labels,explode=tuchu, shadow=True)
plt.show()
```

这个实例添加了两个参数，explode=tuchu, shadow=True,`explode=tuchu`控制突出显示效果，~~其实这个突出效果是用便宜实现的，可以自己设置偏移量的多少~~，`shadow=True`控制添加阴影效果。

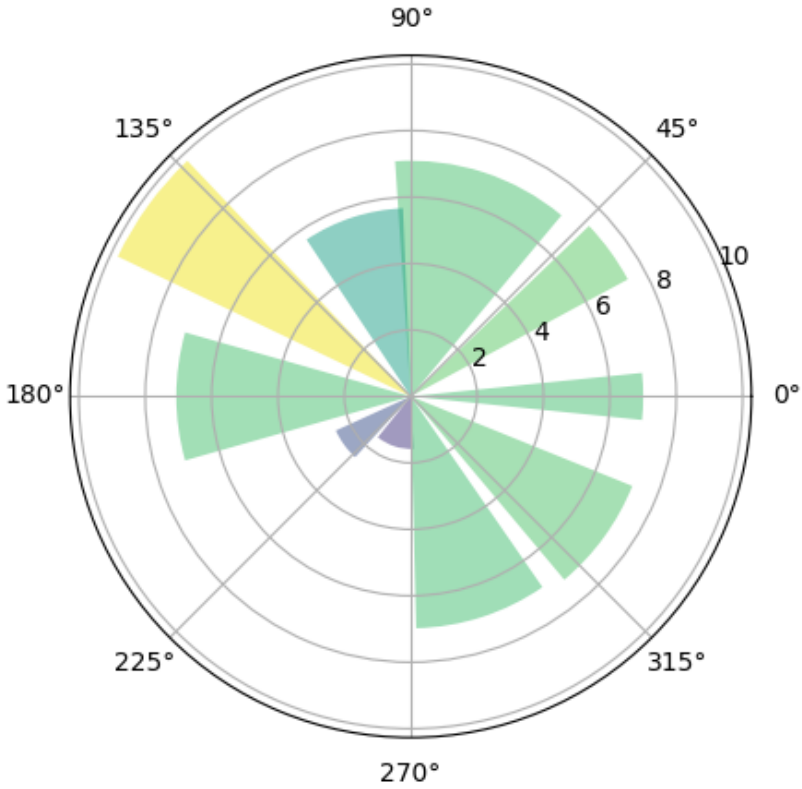
```
import matplotlib.pyplot as plt
#添加以下两句代码，显示汉字标题
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False
# 数据
labels = ["数学", "语文", "英语", "物理","化学"]
cj = [91, 88, 89, 89,92]
tuchu = [0, 0.1, 0, 0,0]
# 画图
plt.pie(x=cj, labels=labels,explode=tuchu, shadow=True,autopct="%0.1f%%")
plt.legend()
plt.show()
```

本实例通过autopct="%0.1f%%显示数据标签，其中%0.1f指定显示小数点后一位的浮点数，后面的两个%%用于显示一个%号，通过plt.legend()控制了显示图例。

拓展：极轴饼图

```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(19680801)
N = 10
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)
# left表示从哪开始，
# radii表示从中心点向边缘绘制的长度（半径）
```

```
# width表示末端的弧长
# 自定义颜色和不透明度
for r, bar in zip(radii, bars):
    bar.set_facecolor(plt.cm.viridis(r / 10.))
    bar.set_alpha(0.5)
plt.show()
```



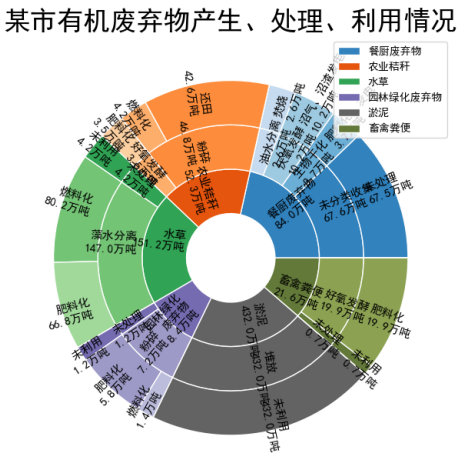
拓展: 多层嵌套饼图

```
from matplotlib import pyplot as plt
import numpy as np
size = 0.25
base = 50
plt.rcParams['font.family'] = 'SimHei'
fig, ax = plt.subplots(figsize=(10, 10))
vals_middle = np.array([
    [47.5, 11.7, 15.2, 9.6],
    [0, 44.8, 7.5, 0],
    [9.2, 68.5, 0, 0],
    [1.2, 7.2, 0, 0],
    [80, 0, 0, 0],
    [1.7, 18.9, 0, 0]
])
vals_outer = np.array([
    [47.5, 11.7, 15.2, 9.6],
    [0, 36.6, 8.2, 7.5],
    [9.2, 38.1, 30.4, 0],
    [1.2, 5.8, 1.4, 0],
    [80, 0, 0, 0],
    [1.7, 18.9, 0, 0]
])
```

```
vals_inner = vals_middle.sum(axis=1)
# 最内圈使用的数值为内圈各类数据加上base
vals_first = vals_inner + base
'''
第二圈使用的数值，因为最内圈每个类别都加上了base，所以为了确保第二圈的数值和内圈相匹配，第二圈的各类别要按照各自所占的比例分配各类的总数值.
'''
vals_second = np.zeros((6, 4))
for i in range(6):
    s_a = vals_first[i]
    s_b = vals_middle[i].sum()
    # 如果第二圈某类总数值为0，则分配base.
    if s_b == 0.0:
        vals_second[i][1] = base
        continue
    for j in range(4):
        vals_second[i][j] = (vals_middle[i][j] / s_b) * s_a
# 第三圈使用的数值，和上方同理
vals_third = np.zeros((6, 4))
for i in range(6):
    s_a = vals_first[i]
    s_b = vals_outer[i].sum()
    if s_b == 0.0:
        vals_third[i][1] = base
        continue
    for j in range(4):
        vals_third[i][j] = (vals_outer[i][j] / s_b) * s_a
# 获取colormap tab20c和tab20b的颜色
cmap_c = plt.get_cmap("tab20c")
cmap_b = plt.get_cmap("tab20b")
# 使用tab20c的全部颜色和tab20b中的 5至8 颜色
cmap_1 = cmap_c(np.arange(20))
cmap_2 = cmap_b(np.array([4, 5, 6, 7]))
# 内圈的颜色是每4个颜色中色彩最深的那个. vstack是将两类颜色叠加在一起
inner_colors = np.vstack((cmap_1[::4], cmap_2[0]))
# 外圈的颜色是全部24种颜色
outer_colors = np.vstack((cmap_1, cmap_2))
labels_first = ["餐厨废弃物\n{}万吨".format(vals_inner[0]),
                "农业秸秆\n{}万吨".format(vals_inner[1]),
                "水草\n151.2万吨",
                "园林绿化\n废弃物\n{}万吨".format(vals_inner[3]),
                "淤泥\n432.0万吨",
                "畜禽粪便\n21.6万吨"
                ]
labels_seocnd = [
    "未分类收集\n67.6万吨",
    "生物干化\n3.7万吨",
    "厌氧发酵\n10.2万吨",
    "油水分离\n2.6万吨",
    "",
    "粉碎\n46.8万吨",
    "好氧发酵\n3.5万吨",
    "",
    "未处理\n4.2万吨",
    "藻水分离\n147.0万吨",
    "",
    "",
    "未处理\n1.2万吨",
```

```
    "粉碎\n7.2万吨",
    "",
    "",
    "堆放\n432.0万吨",
    "",
    "",
    "",
    "未处理\n0.7万吨",
    "好氧发酵\n19.9万吨",
    "",
    "",
]
labels_third = [
    "未处理\n67.5万吨",
    "肥料化、发电\n3.7万吨",
    "沼气、沼渣发电\n10.2万吨",
    "焚烧\n2.6万吨",

    "",
    "还田\n42.6万吨",
    "燃料化\n4.2万吨",
    "肥料化\n3.5万吨",

    "未利用\n4.2万吨",
    "燃料化\n80.2万吨",
    "肥料化\n66.8万吨",
    "",

    "未利用\n1.2万吨",
    "肥料化\n5.8万吨",
    "燃料化\n1.4万吨",
    "",

    "未利用\n432.0万吨",
    "",
    "",
    "",

    "未利用\n0.7万吨",
    "肥料化\n19.9万吨",
    "",
    "",
]
handles, labels = ax.pie(vals_first, radius=1 - size - size,
                         labels=labels_first,
                         labeldistance=0.5, rotatelabels=True, textprops=
{'fontsize': 11},
                         colors=inner_colors, wedgeprops=dict(width=size,
edgecolor='w'))
ax.pie(vals_second.flatten(), radius=1 - size, colors=outer_colors,
       labels=labels_seocnd,
       labeldistance=0.7, rotatelabels=True, textprops={'fontsize': 11},
       wedgeprops=dict(width=size, edgecolor='w'))
ax.pie(vals_third.flatten(), radius=1, colors=outer_colors,
       labels=labels_third,
       labeldistance=0.8, rotatelabels=True, textprops={'fontsize': 11},
       wedgeprops=dict(width=size, edgecolor='w'))
```

```
plt.title('某市有机废弃物产生、处理、利用情况', fontsize=25)
plt.legend(handles=handles, labels=[
    "餐厨废弃物",
    "农业秸秆",
    "水草",
    "园林绿化废弃物",
    "淤泥",
    "畜禽粪便"],
           loc=1
           )
plt.show()
```



这个例子来源于CSDN不是我自己写的，权拿过来做一个参考。

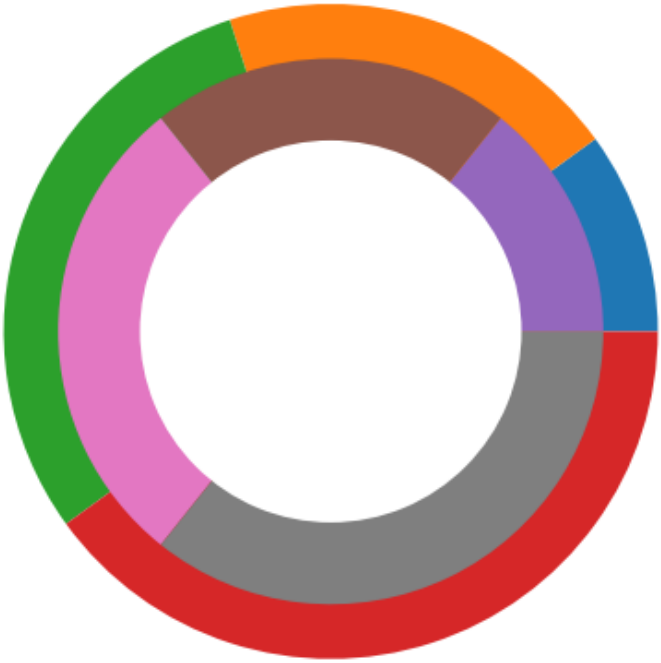
下面写个简单的

目前matplotlib的官方文档中尚无双层饼图绘制的专门介绍~~我那个渣渣的英语即使有也多半找不到~~，但是pie（）函数中有个饼图半径的设置**radius**，我们可以通过手动规定饼图半径，pie()由于是通过一层一层的堆叠，后面一层覆盖掉前面的层次，我们可以这样来实现两张图片的堆叠。

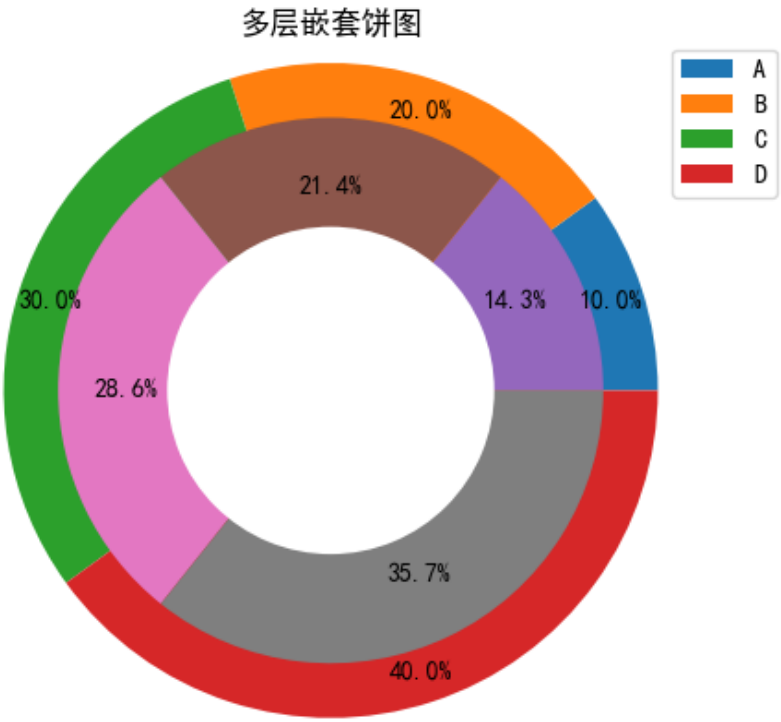
```
import matplotlib.pyplot as plt
vals1 = [1, 2, 3, 4]
vals2 = [2, 3, 4, 5]
fig, ax = plt.subplots()
labels = 'A', 'B', 'C', 'D'
ax.pie(vals1, radius=1.2)
ax.pie(vals2, radius=1)
ax.set(aspect="equal")
plt.show()
```

进一步实现环图，这也是这个道理，使用一张纯白的饼图去覆盖上两张图片就能实现多层嵌套饼图

```
import matplotlib.pyplot as plt
vals1 = [1, 2, 3, 4]
vals2 = [2, 3, 4, 5]
vals3 = [1]
fig, ax = plt.subplots()
labels = 'A', 'B', 'C', 'D'
ax.pie(vals1, radius=1.2)
ax.pie(vals2, radius=1)
ax.pie(vals3,colors='w',radius=0.7)
ax.set(aspect="equal")
plt.show()
```



```
import matplotlib
import matplotlib.pyplot as plt
matplotlib.rcParams['font.family']='SimHei'
vals1 = [1, 2, 3, 4]
vals2 = [2, 3, 4, 5]
vals3=[1]
fig, ax = plt.subplots()
labels = 'A', 'B', 'C', 'D'
ax.pie(vals1, radius=1.2,autopct='%1.1f%%',pctdistance=0.9)
ax.pie(vals2, radius=1,autopct='%1.1f%%',pctdistance=0.75)
ax.pie(vals3, radius=0.6,colors='w')
ax.set(aspect="equal", title='多层嵌套饼图')
#plt.legend()
plt.legend(labels,bbox_to_anchor=(1, 1), loc='best', borderaxespad=0.)
plt.show()
```

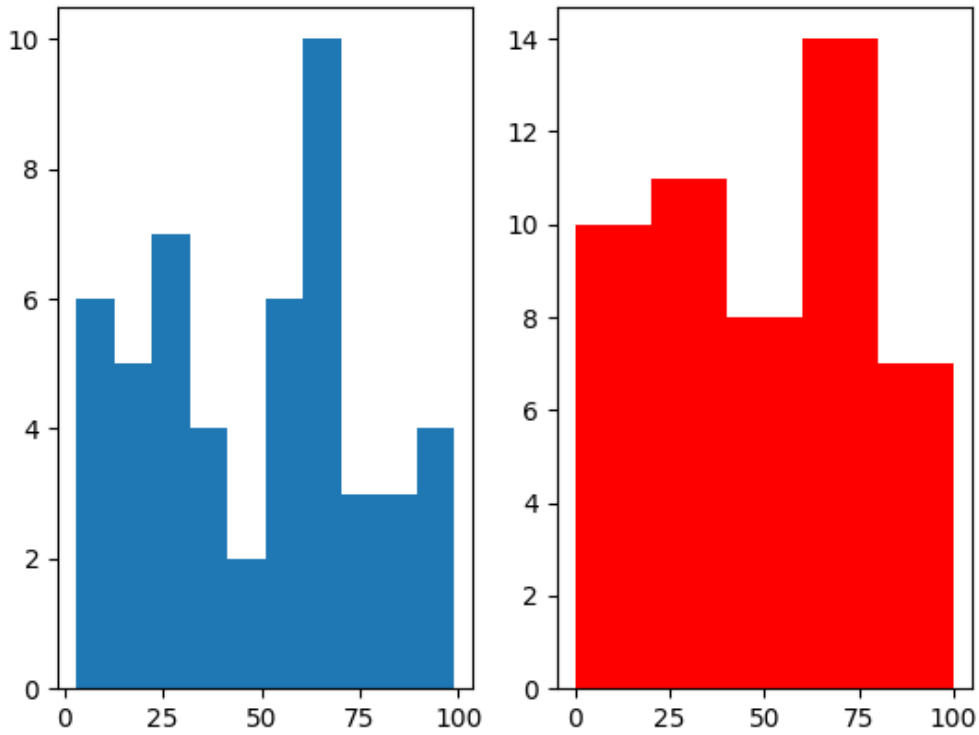


5.7 直方图

在数据分析应用中，直方图是一种对数据分布情况进行表达的数据可视化形式，其横坐标通常表示统计样本，纵坐标表示某个样本对应的某个属性的度量值。

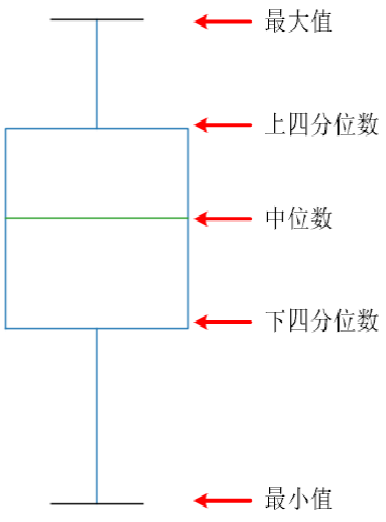
```
plt.hist(x, bins=None, density=None,…)
```

```
import numpy as np
import matplotlib.pyplot as plt
figure = plt.figure()
axes1 = figure.add_subplot(1,2,1)
axes2 = figure.add_subplot(1,2,2)
x = np.random.randint(0,100,size=50)
#在0到100的范围内，随机生成五十个随机数，
axes1.hist(x)
axes2.hist(x,bins=5,range=[0,100],histtype='barstacked',color='r')
#这个实际上频率分布直方图，y坐标是x在一定范围内出现的次数
plt.show()
```



5.8 箱线图

箱线图又称为盒式图、盒状图或箱形图，是一种用于显示一组数据分散情况资料的统计图。箱线图一般不受异常值的影响，是一种相对稳定的数据离散分布情况的描述方式。箱线图中显示的是一组数据的最大值、最小值、中位数、上下四分位数及异常值。



将整组数据等分成4份，就是四分位数，四分位数有3个
第1个四分位数就是通常所说的四分位数，称为下四分位数，通常用Q1表示，等于整组数据中所有数值由小到大排列后第25%的数字，第2个四分位数是中位数，用Q2表示，等于整组数据中所有数值由小到大排列后第50%的数字，第3个四分位数是上四分位数，用Q3表示，等于整组数据中所有数值由小到大排列后第75%的数字

箱线图的大小是由数据升序排列后，中间的50%个数据决定的。因此，前25%数据和后25%数据都无法影响箱线图，它们可以变得任意远，且不会扰动四分位值。若数据集呈标准正态分布，则中位数位于Q1和Q3中间，箱线图的中间线恰好位于上底和下底的正中央。若中位数偏向于下底，则数据集倾向于左偏态；若中位数偏向于上底，则数据集倾向于右偏态。箱线图的宽度在一定程度上反映了数据的波动程度。因为箱线图包含中间50%的数据，若它越扁，则说明数据较为集中，若它越宽，则说明数据越分散。

可以通过观察中位线位于箱体的哪个部分判断数据分布是否均匀，如果中位线位于箱体的中心位置附近则数据部比较均匀，反之则不均匀。

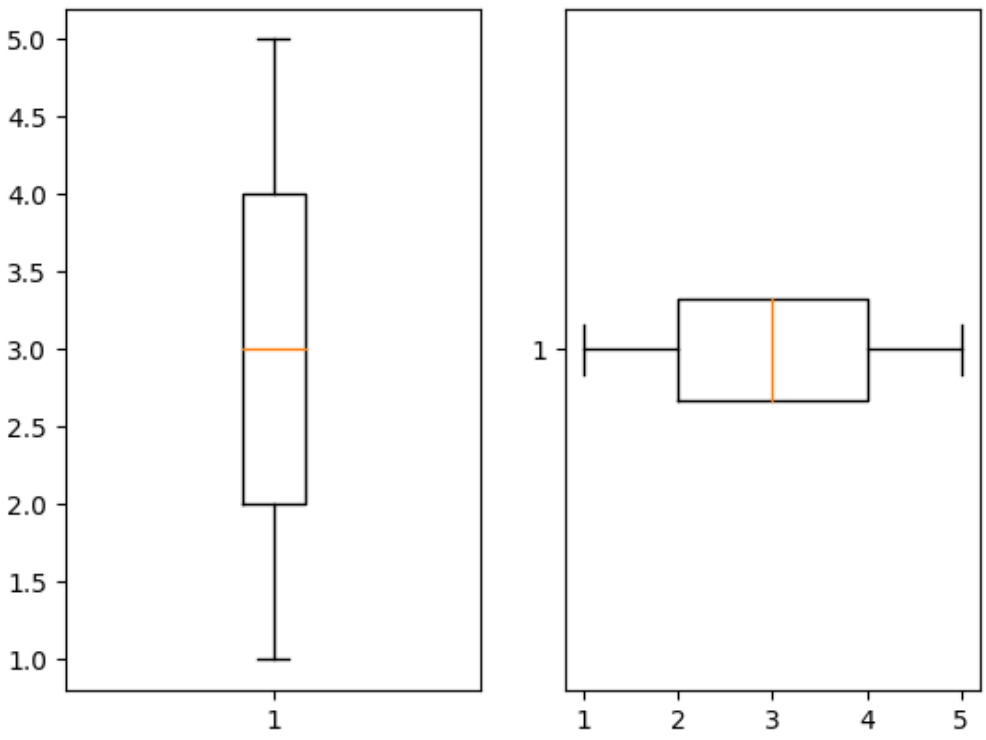
```
plt.boxplot(x, notch=None, sym=None, vert=None, whis=None, positions=None,
widths=None, patch_artist=None, meanline=None, showmeans=None, showcaps=None,
showbox=None, showfliers=None, boxprops=None, labels=None, flierprops=None,
medianprops=None, meanprops=None, capprops=None, whiskerprops=None)
```

例5-23

```
import matplotlib.pyplot as plt
x = [1,2,3,4,5]
plt.boxplot(x)
plt.show()
```

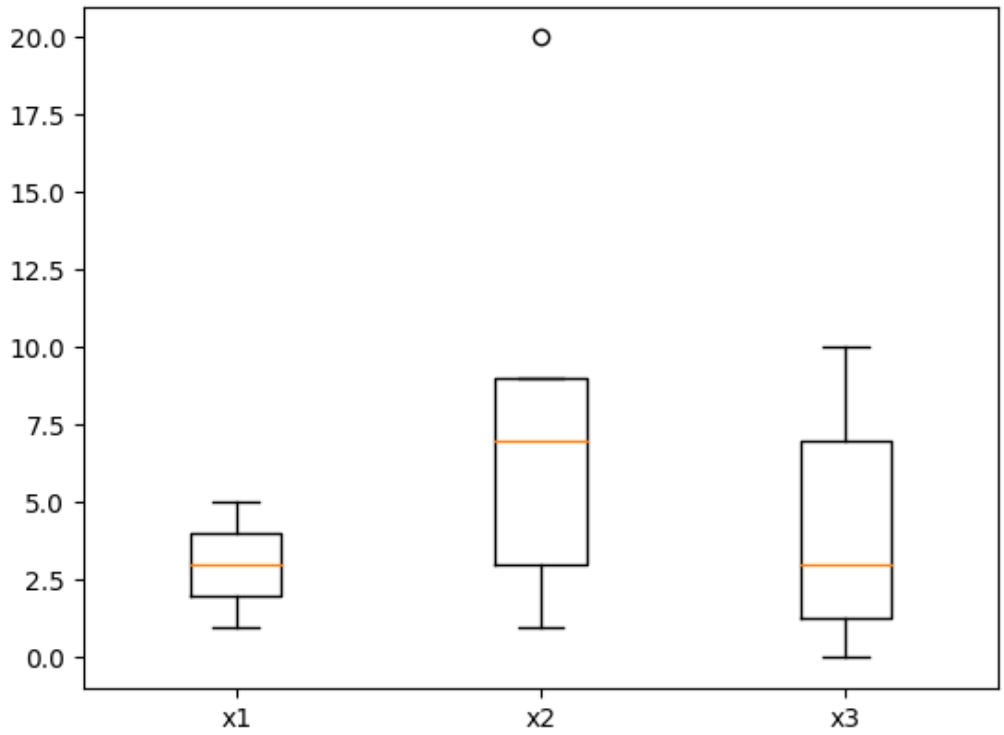
横版5-23：

```
import matplotlib.pyplot as plt
x = [1,2,3,4,5]
plt.boxplot(x,vert = False)
plt.show()
```



例5-24：

```
import matplotlib.pyplot as plt
x1=[1,2,3,4,5]
x2=[1,3,7,9,20]
x3= [0,1,2,4,8,10]
plt.boxplot((x1,x2,x3),labels=('x1','x2','x3'))
plt.show()
```

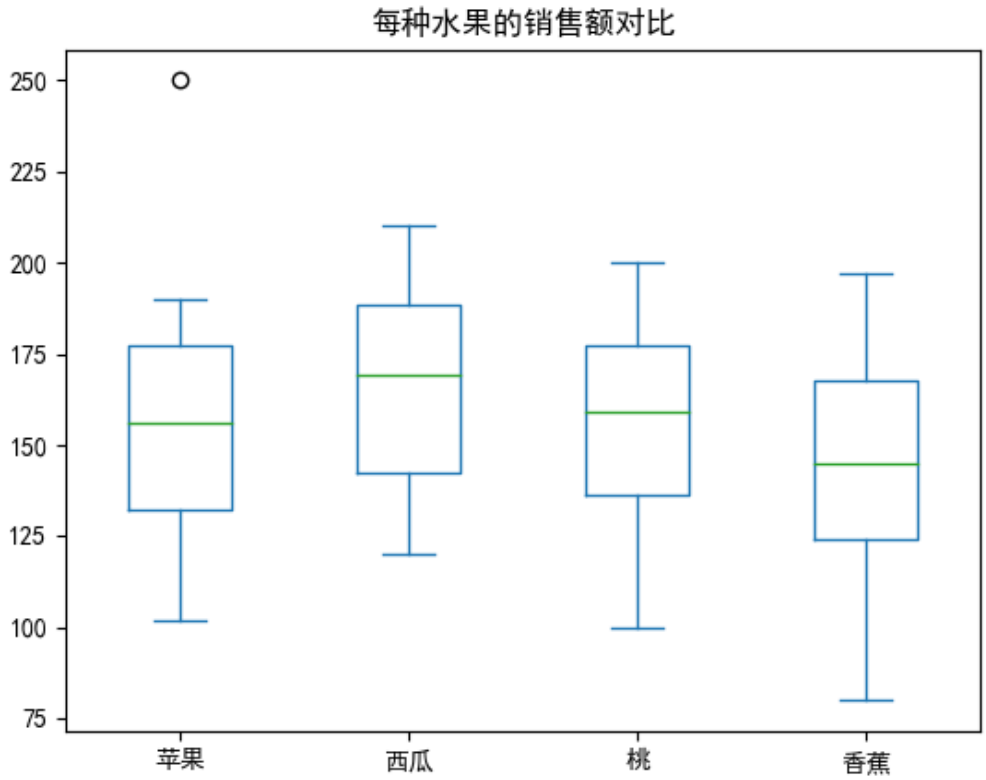


从运行结果可以看出，x2数据有一个异常数据点，绘制在箱线图外部。

在进行数据预处理时，常常需要进行数据的异常值分析，来检测数据是否有错误和不合理的数据。异常值会影响整租数据，应该从本组数据中丢弃异常值。

例5-25：

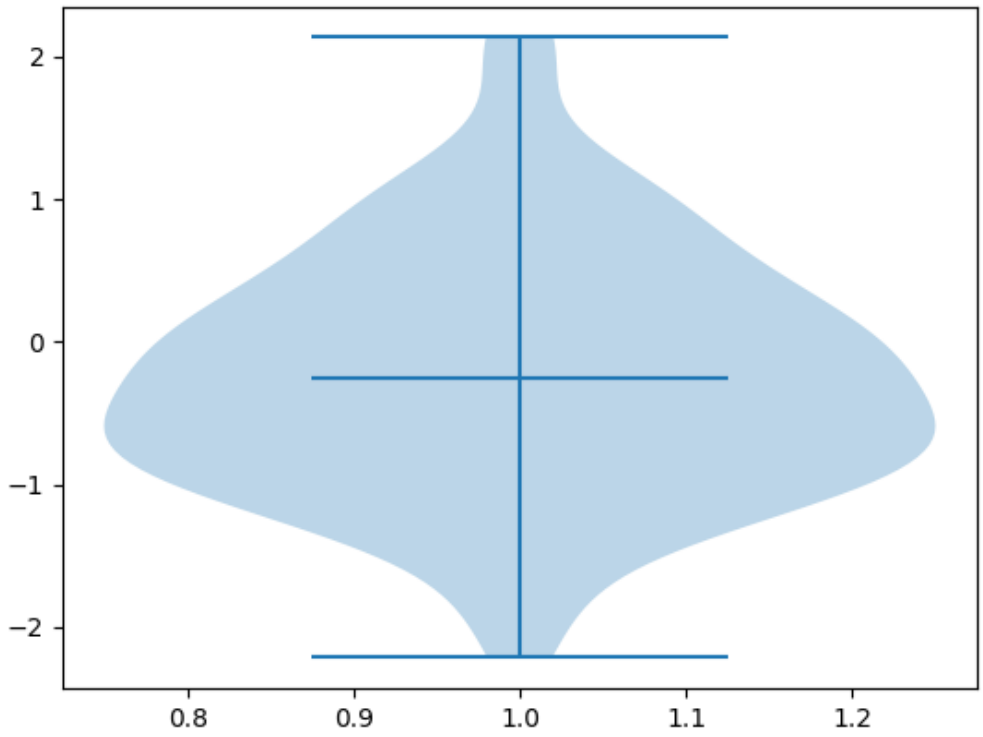
```
import pandas as pd
import matplotlib.pyplot as plt
plt.rcParams['font.family'] = 'SimHei'
data = {
'苹果': [102, 120, 130, 140, 150, 162, 170, 180, 190, 250],
'西瓜': [120, 133, 140, 150, 169, 170, 183, 190, 200, 210],
'桃': [100, 120, 135, 140, 158, 160, 170, 180, 195, 200],
"香蕉": [80, 100, 120, 137, 140, 150, 160, 170, 180, 197]
}
df = pd.DataFrame(data)
df.plot.box(title="每种水果的销售额对比")
plt.show()
```



5.9小提琴图

小提琴图是用来展示多组数据的分布状态及概率密度的一种数据可视化形式。小提琴图结合了箱线图和密度图的特征，主要用来显示数据的分布形状，特别适用于大数据统计分析。

```
import matplotlib.pyplot as plt
import numpy as np
data=np.random.normal(0, 1, 100)
plt.violinplot(data,showmeans=False,showmedians=True)
plt.show()
```



5.10热力图

Seaborn是基于matplotlib库的Python可视化库， 它提供了一个高级界面来绘制有吸引力的统计图形。Seaborn其实是在matplotlib的基础上进行了更高级的API封装，从而使作图更加容易，不需要经过大量的调整就能使图变得更精致。
绘图热力图是大数据分析应用的一种常用可视化形式，用来表示大量数据的关联关系。
在实际应用中，通常使用seaborn来绘制热力图。

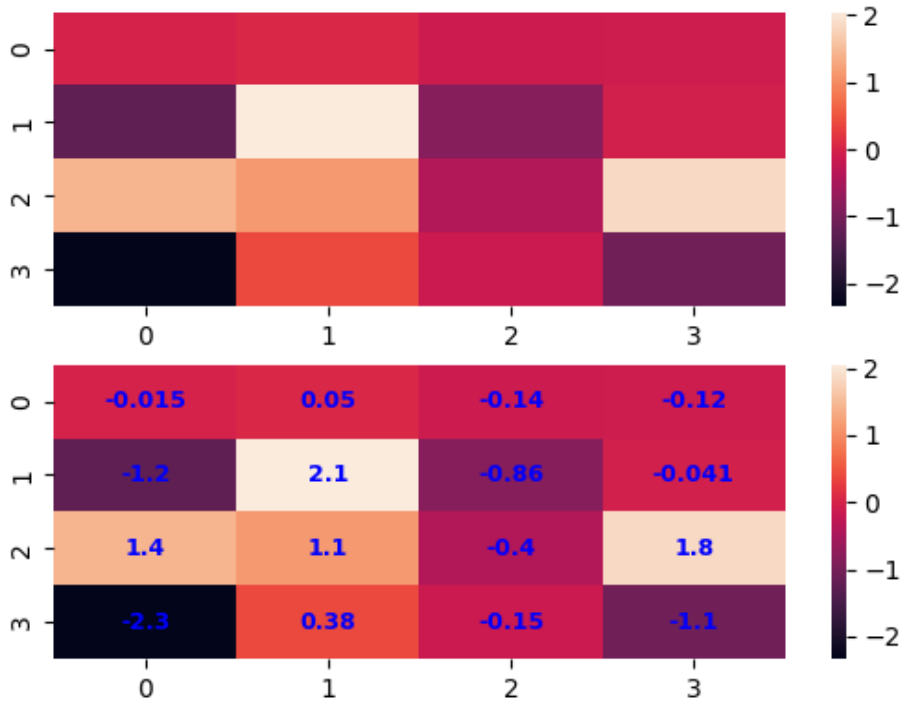
```
seaborn.heatmap(data, vmin=None, vmax=None,cmap=None, center=None, robust=False,
annot=None, fmt='.2g', annot_kws=None,linewidths=0, linecolor='white',
cbar=True, cbar_kws=None, cbar_ax=None,square=False, xticklabels='auto',
yticklabels='auto', mask=None,…)
```

热力图绘制属于Python绘图的中高级应用，在绘制前要先添加如下语句。

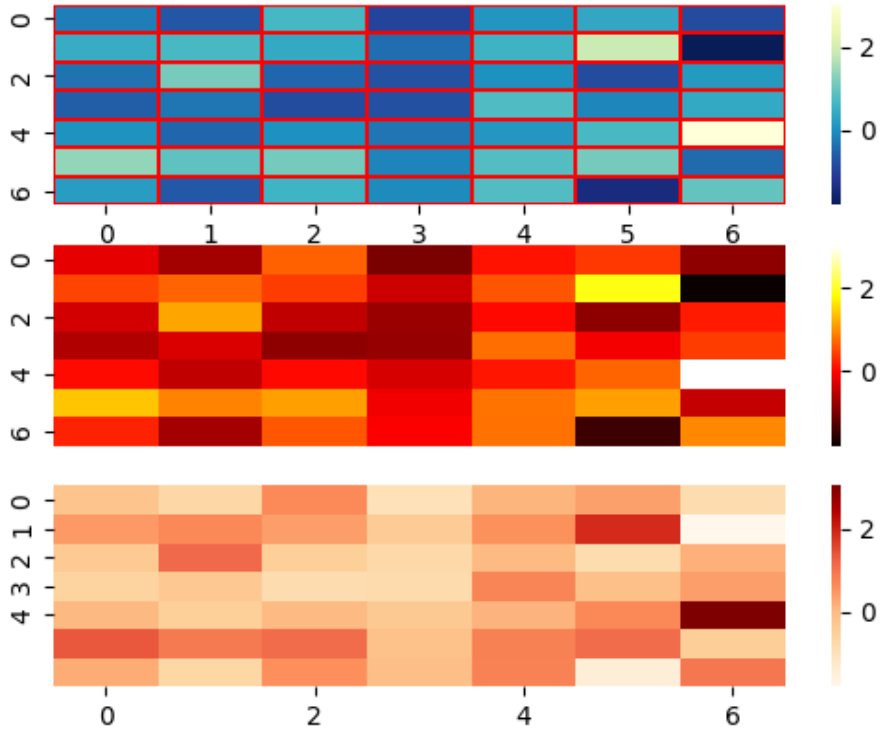
```
import seaborn as sns
```

例5-27：

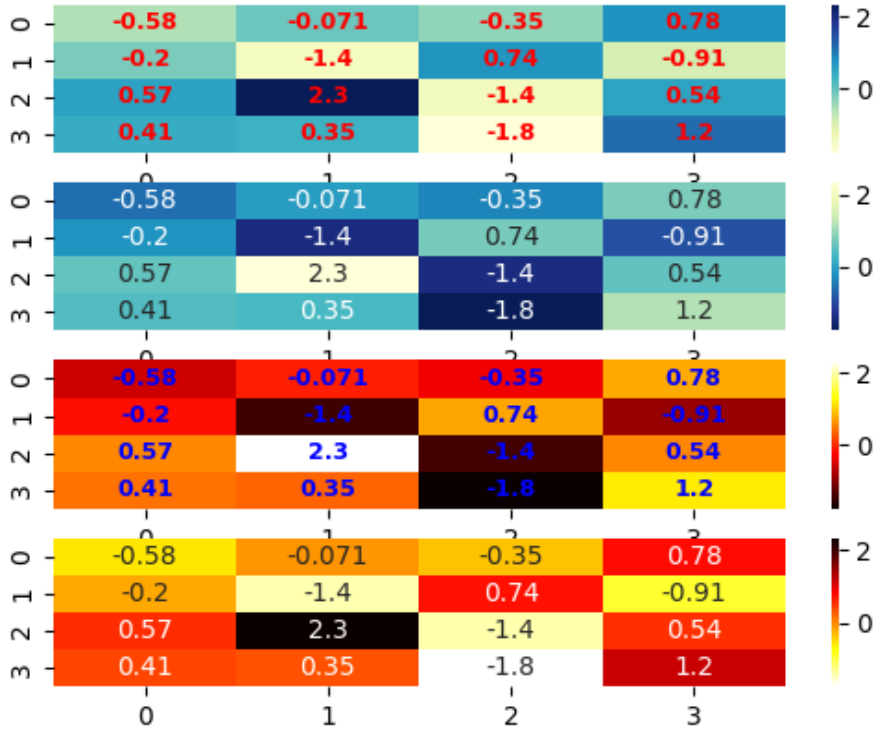
```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
x = np.random.randn(4, 4)
f, ax = plt.subplots(nrows=2)
sns.heatmap(x, ax=ax[0])
sns.heatmap(x, annot=True, ax=ax[1], annot_kws={'size':9,'weight':'bold',
'color':'blue'})
```




```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
x = np.random.randn(7, 7)
f, ax = plt.subplots(nrows=3)
sns.heatmap(x, cmap="YlGnBu_r", linewidths = 0.05, linecolor= 'red', ax = ax[0])
p1 = sns.heatmap(x, ax=ax[1], cmap="hot", center=None, xticklabels=False)
p2 = sns.heatmap(x, ax=ax[2], cmap="OrRd", center=None, xticklabels=2,
yticklabels=list(range(5)))
```



```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
x = np.random.randn(4, 4)
f, ax = plt.subplots(nrows=4)
sns.heatmap(x, cmap="YlGnBu",annot=True, ax=ax[0], annot_kws=
{'size':9,'weight':'bold', 'color':'red'})
sns.heatmap(x, cmap="YlGnBu_r",annot=True, ax=ax[1])
sns.heatmap(x, cmap="hot",annot=True, ax=ax[2], annot_kws=
{'size':9,'weight':'bold', 'color':'blue'})
sns.heatmap(x, cmap="hot_r",annot=True, ax=ax[3])
```



5.11云图

词云图是目前文本大数据分析中常用的一种数据可视化方式，通过词云图可以直观的看到相应数据文本中的高频词。目前制作词云图使用最多的是Python库中的wordcloud库，wordcloud库中有3个主要的函数，分别是**wordcloud.Woedcloud()**,**wordcloud.ImageColorGenerator()**及**wordcloud.random_color_func()**

由于于老师没有给我，这些数据，我自己从网上下载了个txt格式的红楼梦，做了一下。

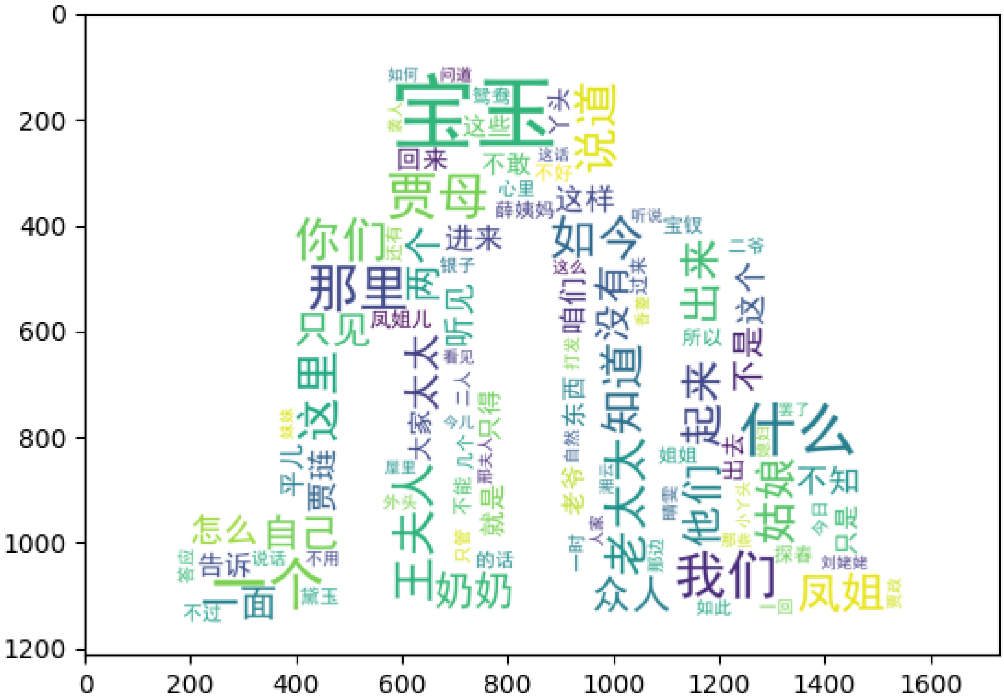
matplotlib.pyplot模块用于绘图展示，**jieba**库用于分词，**numpy**模块和**PIL.image**模块用于制作背景图，**re**用于正则匹配，

注意**PIL.image**模块的真名是pillow,不要试图使用PIL之类的找到添加这个库。

```
pip install pillow
```

```
import re
import jieba
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# 要先pip install wordcloud
from wordcloud import WordCloud
from PIL import Image
f = open("红楼梦.txt", encoding="utf-8")
file_handle = open('红楼梦去字符.txt', mode='a',encoding="utf-8")
line = f.read()
a = re.compile(u'[\u4e00-\u9fa5]+')
b=a.findall(line)
n=len(b)
for i in range(1,n):
```

```
    file_handle.write(b[i-1])
line = f.readline()
file_handle.close()
f.close()
f = open("红楼梦去字符.txt", "r", encoding='utf-8').read()
words = jieba.lcut(f)  # 使用精确模式对文本进行分词
counts = {}  # 通过键值对的形式存储词语及其出现的次数
for word in words:
    if len(word) == 1:  # 单个汉字不计算在内
        continue
    else:
        counts[word] = counts.get(word, 0) + 1  # 遍历所有词语，每出现一次其对应的值加
1
lists = list(counts.items())  # 将键值对转换成列表
lists.sort(key=lambda x: x[1], reverse=True)  # 根据词语出现的次数进行从大到小排序
#file=open("./333.txt","w",encoding="utf")
#for i in range(len(lists)):
#    word,count=lists[i]
#    print("{0:<5}{1:>5}".format(word, count))
#    file.write(word+f'{count}')
#这一部分我是想把所有的词语都导出了写到333.txt里面，之后可能会有用
#file.close()
file=open("./红楼梦词频统计.txt","w",encoding="utf")
file.write(f'词语        次数\n')
#在这里先写入表头
for i in range(100):
    word, count = lists[i]
    file.write(word + f'        {count}\n')
file.close()
for i in range(100):
    word, count = lists[i]
    #for循环输出频率前100的
    print("{0:<5}{1:>5}".format(word, count))
a=pd.read_table('红楼梦词频统计.txt',encoding='utf-8',sep='
',engine='python')
df=pd.DataFrame(a)
#fit_word函数，接受字典类型，其他类型会报错
dic=dict(zip(df['词语'],df['次数']))
#通过PIL库设置背景图片
bg=np.array(Image.open("城市.png"))
#当使用PlL.Image.open()打开图片后，如果要使用函数，需要先将image形式转换成array数组,这里是
由于两个数组存储顺序不一致，
wordcloud =
WordCloud(font_path='simhei.ttf',mask=bg,background_color="white").fit_words(dic
)
plt.imshow(wordcloud)
# 显示
plt.show()
```


0
200
400
600
800
1000
1200
1400
1600
宝玉
贾母
说道
你们
那里
如今
这里
王夫人
太太
老太太
知道
没有
我们
他们
什么
姑娘
凤姐
奶奶
一个
一面
自己
起来
出来
不是
这个
不知
贾琏
听见
两个
进来
这样
薛姨妈
怎么
告诉
平儿
大家
只得
就是
咱们
东西
老爷
二人
只是
宝钗
心里
回来
不敢
这些
丫头
凤姐儿
银子
所以
黛玉
不过
如何
问道
鸳鸯
探春
刘姥姥