```
# 引入Matplotlib引入
from matplotlib import pyplot as plt
import numpy as np
```

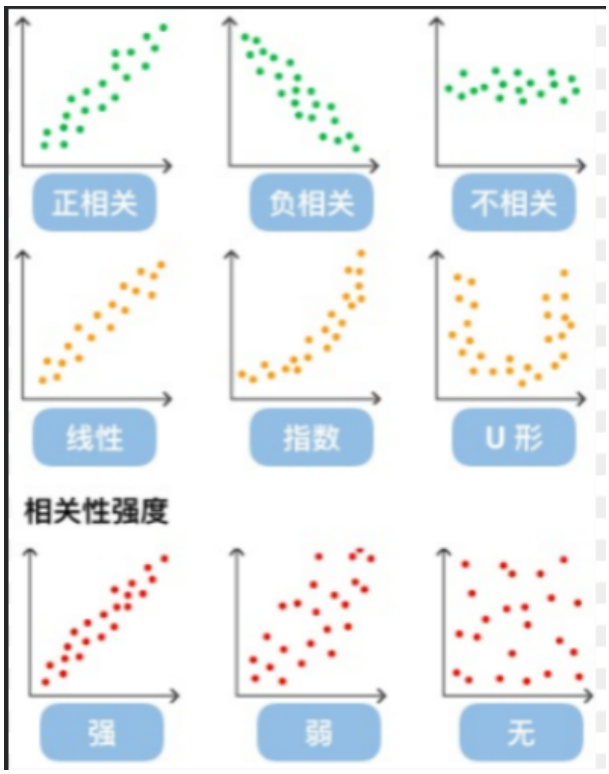
```
# 设置中文字体
plt.rcParams['font.sans-serif'] = ['SimHei']
# 中文负号
plt.rcParams['axes.unicode_minus'] = False

# 设置分别率 为100
plt.rcParams['figure.dpi'] = 100

# 设置大小
plt.rcParams['figure.figsize'] = (5,3)
```

散点图 scatter()

散点图也叫 X-Y 图，它将所有的数据以点的形式展现在直角坐标系上，以显示变量之间的相互影响程度，点的位置由变量的数值决定。

通过观察散点图上数据点的分布情况，我们可以推断出变量间的相关性。如果变量之间不存在相互
关系，那么在散点图上就会表现为随机分布的离散的点，如果存在某种相关性，那么大部分的数据
点就会相对密集并以某种趋势呈现。

数据的相关关系主要分为：正相关（两个变量值同时增长）、负相关（一个变量值增加另一个变量值下降）、
不相关、线性相关、指数相关等，表现在散点图上的大致分布如下图所示。那些离点集群较远的点我们称为离群点或者异常点。



`matplotlib.pyplot.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None, vmax=None, alpha=None, linewidths=None, *, edgecolors=None, plotnonfinite=False, data=None, **kwargs)

- x, y → 散点的坐标
- s → 散点的面积
- c → 散点的颜色（默认值为蓝色，'b'，其余颜色同plt.plot( )）
- marker → 散点样式（默认值为实心圆，'o'，其余样式同plt.plot( )）
- alpha → 散点透明度（[0, 1]之间的数，0表示完全透明，1则表示完全不透明）
- linewidths →散点的边缘线宽
- edgecolors → 散点的边缘颜色
- cmap → Colormap，默认 None，标量或者是一个 colormap 的名字，只有 c 是一个浮点数数组的时才使用

以下实例 scatter() 函数接收长度相同的数组参数，一个用于 x 轴的值，另一个用于 y 轴上的值：

```
# x轴数据
x = np.array([1, 2, 3, 4, 5, 6, 7, 8])
# y轴数据
y = np.array([1, 4, 9, 16, 7, 11, 23, 18])

plt.scatter(x, y)
#plt.plot(x,y)
```

设置图标大小：

```
# x轴数据
x = np.array([1, 2, 3, 4, 5, 6, 7, 8])
# y轴数据
y = np.array([1, 4, 9, 16, 7, 11, 23, 18])
# 生成一个[0,1)之间的随机浮点数或N维浮点数组。
print((20 * np.random.rand(8))** 2)

s = (20 * np.random.rand(8))** 2

plt.scatter(x, y, s,alpha=0.4)
plt.show()
```

自定义点的颜色和透明度：

```
# 颜色方式:
- 1.颜色英文
- 2.字母 r b g
- 3.十六进制 #123ab1
```

```
# x轴数据

x = np.random.rand(50)    # rand()生成一个[0,1)之间的随机浮点数或N维浮点数组。
# y轴数据
y = np.random.rand(50)

#
# 生成一个浮点数或N维浮点数组，取数范围：正态分布的随机样本数。
s = (10 * np.random.randn(50))** 2
# 颜色可以使用一组数字序列
# 如只需要3中颜色
#colors = np.resize(np.array([1,2,3]),100)
# 颜色随机
```

```
colors = np.random.rand(50)

plt.scatter(x, y, s,c=colors)
```

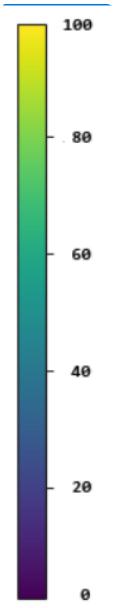
可以选择不同的颜色条--配合cmap参数

颜色条 Colormap

Matplotlib 模块提供了很多可用的颜色条。

颜色条就像一个颜色列表，其中每种颜色都有一个范围从 0 到 100 的值。

下面是一个颜色条的例子： viridis




```
# x轴数据

x = np.random.rand(100)

y = np.random.rand(100)
# 颜色随机
colors = np.arange(1,101)

plt.scatter(x, y, c=colors, cmap='Blues')
```

```
# x轴数据

x = np.arange(1,101)

y = np.arange(1,101)
colors = np.arange(1,101)

plt.scatter(x, y, c=colors, cmap='Set1')
# plt.savefig('./test2.jpg')
# plt.show()
```

```
# x轴数据

x = np.arange(1,11)

y = np.arange(1,11)
# 也可以是小数
colors = np.linspace(0, 1, 10)
#colors = [0.5]*10
plt.scatter(x, y, c=colors, cmap='Set1')

```

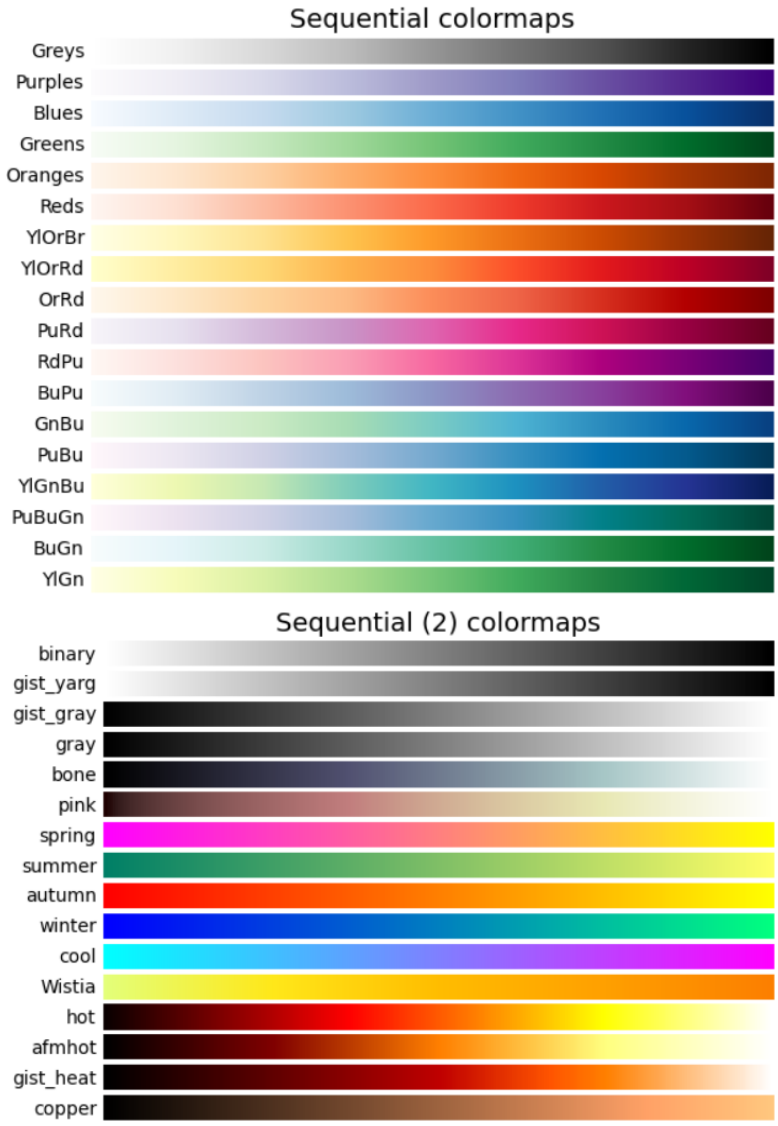
cmap的分类

cmap主要分为以下四大类：

1. Sequential colormaps：连续化色图
   - 特点：在两种色调之间近似平滑变化，通常是从低饱和度（例如白色）到高饱和度（例如明亮的蓝色）。
   - 应用：适用于大多数科学数据，可直观地看出数据从低到高的变化。
   - 1）以中间值颜色命名（eg：viridis 松石绿）:['viridis', 'plasma', 'inferno', 'magma', 'cividis']

   - 2）以色系名称命名，由低饱和度到高饱和度过渡（eg：YlOrRd = yellow-orange-red，其它同理）：

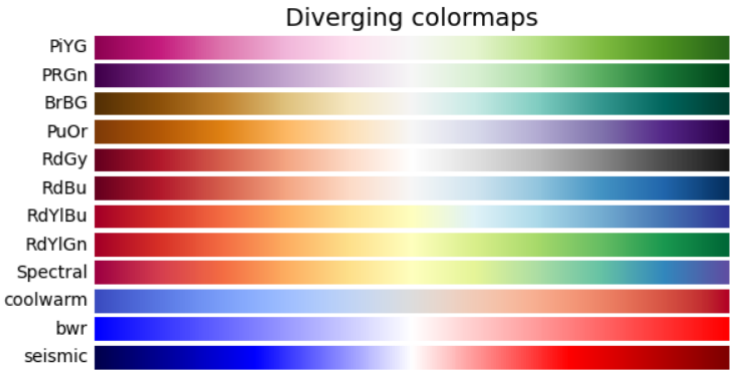
['Greys', 'Purples', 'Blues', 'Greens', 'Oranges', 'Reds','YlOrBr', 'YlOrRd', 'OrRd', 'PuRd', 'RdPu', 'BuPu','GnBu', 'PuBu', 'YlGnBu', 'PuBuGn', 'BuGn', 'YlGn','binary', 'gist_yarg', 'gist_gray', 'gray', 'bone', 'pink','spring', 'summer', 'autumn', 'winter', 'cool', 'Wistia','hot', 'afmhot', 'gist_heat', 'copper']



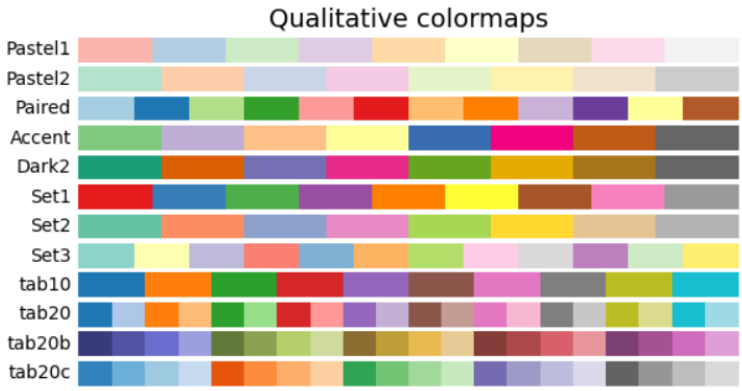
1. Diverging colormaps：两端发散的色图 .

- 特点：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。
- 应用：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）。

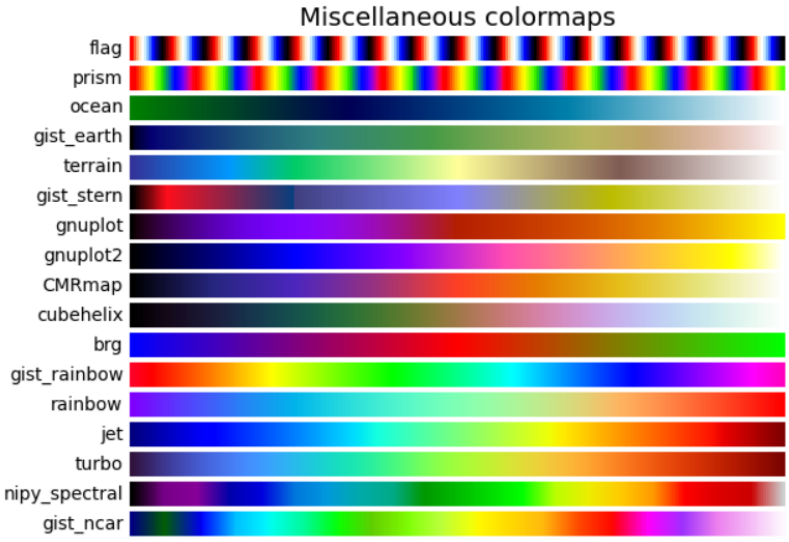
['PiYG', 'PRGn', 'BrBG', 'PuOr', 'RdGy', 'RdBu','RdYlBu', 'RdYlGn', 'Spectral', 'coolwarm', 'bwr', 'seismic']



1. Qualitative colormaps：离散化色图
   - 特点：离散的颜色组合
   - 应用：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))



1. Miscellaneous colormaps：其它色图



保存图片 pyplot.savefig()

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

- fname: （字符串或者仿路径或仿文件）如果格式已经设置，这将决定输出的格式并将文件按fname来保存。如果格式没有设置，在fname有扩展名的情况下推断按此保存，没有扩展名将按照默认格式存储为“png”格式，并将适当的扩展名添加在fname后面。
- dpi: 分辨率，每英寸的点数
- facecolor（颜色或“auto”，默认值是“auto”）：图形表面颜色。如果是“auto”，使用当前图形的表面颜色。
- edgecolor（颜色或“auto”，默认值：“auto”）：图形边缘颜色。如果是“auto”，使用当前图形的边缘颜色。
- format（字符串）：文件格式，比如“png”，“jpg”，“pdf”，“svg”等，未设置的行为将被记录在fname中。
- transparent: 用于将图片背景设置为透明。图形也会是透明，除非通过关键字参数指定了表面颜色和/或边缘

```
'''
1. 第一个参数就是保存的路径
2, 如果路径中包含未创建的文件夹,会报错,需要手动或者使用os模块创建
3. 必须在调用plt.show() 之前保存,否则将保存的是空白图形

'''
import os

# x轴数据
x_axis =[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
# y轴数据
y_axis =[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
# 绘制图形
plt.hist(x_axis, y_axis)
# x和y轴标签
plt.xlabel("X")
plt.ylabel("Y")
# 使用os模块判断目录是否存在
if not os.path.exists("my"):
    # 使用os模块创建文件夹
    os.mkdir("my")
plt.savefig("my/my_show.png")
```

```
plt.show()

```

需要注意 1.在使用保存savefig是必须放到show()函数之前

```
# x轴数据
x_axis =[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
# y轴数据
y_axis =[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
# 绘制图形
plt.plot(x_axis, y_axis)
# x和y轴标签
plt.xlabel("X")
plt.ylabel("Y")

plt.show()
plt.savefig("new_my.jpg")
```

需要注意 2.如果保存到指定文件夹中,一定确保文件夹存在

```
# x轴数据
x_axis =[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
# y轴数据
y_axis =[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
# 绘制图形
plt.plot(x_axis, y_axis)
# x和y轴标签
plt.xlabel("X")
plt.ylabel("Y")

plt.savefig("img/my.png")

plt.show()


```

```
import os
# x轴数据
x_axis =[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
# y轴数据
y_axis =[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

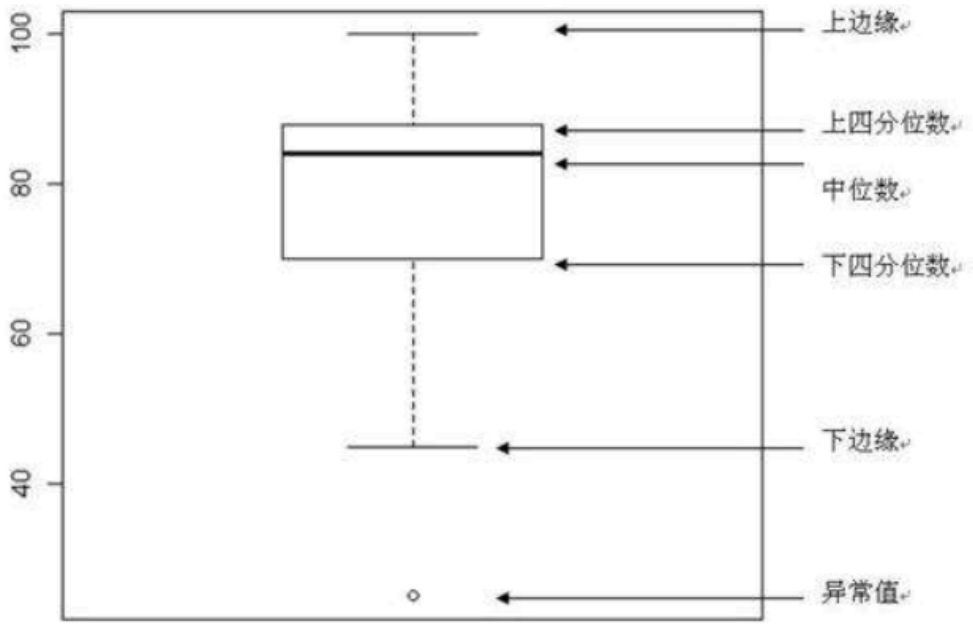
```
# 绘制图形
plt.plot(x_axis, y_axis)
# x和y轴标签
plt.xlabel("X")
plt.ylabel("Y")
# 使用os模块判断文件夹是否存在,不存在进行创建
if not os.path.exists("img"):
    os.mkdir("img")
plt.savefig("img/my.png")

plt.show()
```

箱线图绘制boxplot()

箱线图(Boxplot) 是一种用作显示一组数据分散情况资料的统计图表

箱线图基本介绍

箱线图，又称箱形图（boxplot）或盒式图，不同于一般的折线图、柱状图或饼图等图表，
其包含一些统计学的均值、分位数、极值等统计量，该图信息量较大，不仅能够分析不同类别数据平均水平差异，
还能揭示数据间离散程度、异常值、分布差异等。具体含义可通过如下图表进行说明：



在箱型图中，我们从上四分位数到下四分位数绘制一个盒子，然后用一条垂直触须（形象地称为“盒须”）穿过盒子的中间。上垂线延伸至上边缘（最大值），下垂线延伸至下边缘（最小值）

箱型图应用场景

- 箱型图由于能显示一组数据分散情况，常用于品质管理
- 箱型图有利于数据的清洗，能快速知道数据分别情况
- 箱型图有助于分析一直数据的偏向如分析公司员工收入水平

函数的使用pyplot.boxplot()

`matplotlib.pyplot.boxplot(x, notch=None, sym=None, vert=None, whis=None, positions=None, widths=None, patch_artist=None, bootstrap=None, usermedians=None, conf_intervals=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, manage_ticks=True, autorange=False, zorder=None, *, data=None)

- x：输入数据。类型为数组或向量序列。必备参数。
- notch：控制箱体中央是否有V型凹槽。当取值为True时，箱体中央有V型凹槽，凹槽表示中位数的置信区间；取值为False时，箱体为矩形。类型为布尔值，默认值为False。可选参数。
- vert：箱体的方向，当取值为True时，绘制垂直箱体，当取值为False时，绘制水平箱体。类型为布尔值，默认值为True。可选参数。
- positions：指定箱体的位置。刻度和极值会自动匹配箱体位置。类型为类数组结构。可选参数。默认值为range(1, N+1) ，N为箱线图的个数。
- widths：箱体的宽度。类型为浮点数或类数组结构。默认值为0.5或0.15*极值间的距离。
- labels：每个数据集的标签，默认值为'None'。类型为序列。可选参数。
- autorange：类型为布尔值，默认值为False。可选参数。当取值为True且数据分布满足上四分位数（75%）和下四分位数（25%）相等

- showmeans：是否显示算术平均值。类型为布尔值，默认值为False。可选参数。
- meanline：均值显示为线还是点。类型为布尔值，默认值为False。可选参数。当取值为True，且showmeans、shownotches参数均为True，时显示为线
- capprops：箱须横杠的样式。类型为字典，默认值为None。可选参数。
- boxprops：箱体的样式。类型为字典，默认值为None。可选参数。
- whiskerprops：箱须的样式。类型为字典，默认值为None。可选参数。
- flierprops：离群点的样式。类型为字典，默认值为None。可选参数。
- medianprops：中位数的样式。类型为字典，默认值为None。可选参数。
- meanprops：算术平均值的样式。类型为字典，默认值为None。可选参数。

```
# logspace
x = np.array([1,20,30,50,60])
print(np.mean(x))
plt.boxplot(x,showmeans=True,meanline=True)
#print(x)
# showmeans -- 是否显示算术平均值
# meanline：均值显示为线还是点。类型为布尔值，默认值为False。
# 可选参数。当取值为True，且showmeans、shownotches参数均为True，时显示为线
# labels：每个数据集的标签
#plt.boxplot(x,showmeans=True, meanline=True, labels=["A"])
# 只设置显示平均值 ,只是一个点
#plt.boxplot(x,showmeans=True)
# 如果想想显示的平均值为线 ,必须再设置meanline为True
#plt.boxplot(x,showmeans=True, meanline=True)
plt.grid()
plt.show()
```

```python
# 创建5行5列的数据
x = np.random.randint(10,100,size=(5,5))

box = {"linestyle":'--',"linewidth":1,"color":'blue'}

mean = {"marker":'o','markerfacecolor':'pink','markersize':2}

# boxprops-- 箱体的样式。类型为字典

# meanprops-- 算术平均值的样式。类型为字典

plt.boxplot(x,meanline=True,showmeans=True,labels=
["A","B","C","D","E"], boxprops=box,meanprops=mean)
plt.show()
```

```python
x = np.array([1,2,3,4,5,3,4,5,7,1,8,7]).reshape(3,4)
print(x)
plt.boxplot(x,showmeans=True,meanline=True,labels=list("ABCD"))
plt.grid()
plt.show()
```

```
# 读取文件..语文成绩  数学  英语  历史


线形图  plt.plot
柱状图  plt.bar   barh
散点图  scatter   饼状图 pie
直方图  hist
箱线图  boxplot
--- 设置


```

词云图

wordcloud 是什么？

词云图，也叫文字云，是对文本中出现频率较高的“关键词”予以视觉化的展现，词云图过滤掉大量的低频低质的文本信息，使得浏览者只要一眼扫过文本就可领略文本的主旨。

WordCloud

WordCloud 是一款python环境下的词云图工具包，同时支持python2和python3，能通过代码的形式把关键词数据转换成直观且有趣的图文模式。

pip默认安装方式

```
pip install wordcloud
```

如果是使用conda的方式安装，则使用以下命令安装

```
conda install -c conda-forge wordcloud
```

```
pip install wordcloud
```

```
from wordcloud import WordCloud

with open("data/a_new_hope.txt", encoding="utf-8") as file:
    # 数据文件
```

```
    txt = file.read()
    # 如果数据文件中包含的有中文的话，font_path必须指定字体，否则中文会乱码
    # collocations：是否包括两个词的搭配，默认为True，如果为true的时候会有
    # 重复的数据，这里我不需要重复数据，所以设置为False
    # width 幕布的宽度，height 幕布的高度
    # max_words 要显示的词的最大个数
    # generate 读取文本文件
    wordcloud = WordCloud(font_path="C:/Windows/Fonts/simfang.ttf",
                                                collocations=False,
                          background_color="black",
                          width=800,
                          height=600,
                          max_words=50).generate(txt)
    # 生成图片
    image = wordcloud.to_image()
    # 展示图片
    image.show()
    # 写入文件
    wordcloud.to_file("tag.jpg")
```

```
from wordcloud import WordCloud

# 数据文件
#txt = "The awesome yellow planet of Tatooine emerges from a total eclipse, her two moons glowing against"
txt ="谜语 人家 里，他 把 布鲁斯·韦恩 的 照片 和 蝙蝠侠"
# 如果数据文件中包含的有中文的话，font_path必须指定字体，否则中文会乱码
# collocations：是否包括两个词的搭配，默认为True，如果为true的时候会有
# 重复的数据，这里我不需要重复数据，所以设置为False
# width 幕布的宽度，height 幕布的高度
# max_words 要显示的词的最大个数
# generate 读取文本文件
wordcloud = WordCloud(font_path="C:/Windows/Fonts/simfang.ttf",
                                        collocations=False,
                background_color="black",
                width=800,
                height=600,
                max_words=50).generate(txt)
# 生成图片
image = wordcloud.to_image()
# 展示图片
image.show()
```

```
# 写入文件
wordcloud.to_file("tag.jpg")
```

```
from wordcloud import WordCloud
import jieba
# 数据文件

txt ="皇后区的友好邻居变成了宇宙的，现实世界资本的侵蚀之下，只有哥谭还维持着它的地域性"
txt_list = jieba.cut(txt)
```

```
txt_str = " ".join(txt_list)

```

```

wordcloud = WordCloud(font_path="C:/Windows/Fonts/simfang.ttf",
                                      collocations=False,
                background_color="black",
                width=800,
                height=600,
                max_words=50).generate(txt_str)
# 生成图片
image = wordcloud.to_image()
# 展示图片
image.show()
# 写入文件
wordcloud.to_file("tag.jpg")
```

✻ 详细参数讲解请查看

https://www.yuque.com/darren-irbls/python/pr2zc5

属性	数据类型|默认值	解析
font_path	string	字体路径 windows: C:/Windows/Fonts/ Linux: /usr/share/fonts
width	int (default=400)	输出的画布宽度，默认为400像素
height	int (default=200)	输出的画布高度，默认为200像素
prefer_horizontal	float (default=0.90)	词语水平方向排版出现的频率,默认0.9 所以词语垂直方向排版出现频率为0.1
mask	nd-array or None (default=None)	如果参数为空，则使用二维遮罩绘制词云 如果mask非空，设置的宽高值将被忽略 遮罩形状被 mask 取代
scale	float (default=1)	按照比例进行放大画布，如设置为1.5， 则长和宽都是原来画布的1.5倍
min_font_size	int (default=4)	显示的最小的字体大小
font_step	int (default=1)	字体步长，如果步长大于1，会加快运算 但是可能导致结果出现较大的误差
max_words	number (default=200)	要显示的词的最大个数
stopwords	set of strings or None	设置需要屏蔽的词，如果为空，则使用内置的STOPWORDS
background_color	color value default=” black”	背景颜色

属性	数据类型|默认值	解析
max_font_size	int or None default=None	显示的最大的字体大小
mode	string (default=”RGB”)	当参数为“RGBA”并且background_color不为空时，背景为透明
relative_scaling	float (default=.5)	词频和字体大小的关联性
color_func	callable, default=None	生成新颜色的函数，如果为空，则使用 self.color_func
regexp	string or None (optional)	使用正则表达式分隔输入的文本
collocations	bool, default=True	是否包括两个词的搭配
colormap	string or matplotlib colormap default=”viridis”	给每个单词随机分配颜色，若指定color_func，则忽略该方法
random_state	int or None	为每个单词返回一个PIL颜色

中文使用词云图--- 需要使用jieba分词模块

- 还需要学习jieba分词
  -结巴”中文分词：做最好的 Python 中文分词组件

特点

-支持三种分词模式：

- 精确模式，试图将句子最精确地切开，适合文本分析；

- 全模式，把句子中所有的可以成词的词语都扫描出来, 速度非常快，但是不能解决歧义；
- 搜索引擎模式，在精确模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词。
- 支持繁体分词
- 支持自定义词典

安装说明

代码对 Python 2/3 均兼容

- 全自动安装：easy_install jieba 或者 pip install jieba / pip3 install jieba
- 半自动安装：先下载 http://pypi.python.org/pypi/jieba/ ，解压后运行 python setup.py install
- 手动安装：将 jieba 目录放置于当前目录或者 site-packages 目录
- 通过 import jieba 来引用

```
pip install jieba
```

- jieba.cut 方法接受三个输入参数: 需要分词的字符串；cut_all 参数用来控制是否采用全模式；HMM 参数用来控制是否使用 HMM 模型
- jieba.cut_for_search 方法接受两个参数：需要分词的字符串；是否使用 HMM 模型。该方法适合用于搜索引擎构建倒排索引的分词，粒度比较细
  待分词的字符串可以是 unicode 或 UTF-8 字符串、GBK 字符串。注意：不建议直接输入 GBK 字符串，可能无法预料地错误解码成 UTF-8
- jieba.cut 以及 jieba.cut_for_search 返回的结构都是一个可迭代的 generator，可以使用 for 循环来获得分词后得到的每一个词语(unicode)，或者用
  =jieba.lcut 以及 jieba.lcut_for_search 直接返回 list

```
import jieba

seg_list = jieba.cut("我来到北京清华大学")  # 默认是精确模式
print(" ".join(seg_list))

seg_list = jieba.cut("我来到北京清华大学", cut_all=True)
print("Full Mode: " + " ".join(seg_list))  # 全模式

```

```python
seg_list = jieba.cut("我来到北京清华大学", cut_all=False)
print("Default Mode: " + "  ".join(seg_list))  # 精确模式

seg_list = jieba.cut("小明硕士毕业于中国科学院计算所，后在日本京都大学深造")  # 默认是精确模式
print(" ".join(seg_list))

seg_list = jieba.cut_for_search("小明硕士毕业于中国科学院计算所，后在日本京都大学深造")  # 搜索引擎模式
print(" ".join(seg_list))
```

jieba.analyse的使用：提取关键字

- 第一个参数：待提取关键词的文本
- 第二个参数 topK：返回关键词的数量，重要性从高到低排序
- 第三个参数 withWeight：是否同时返回每个关键词的权重
- 第四个参数allowPOS=()：词性过滤，为空表示不过滤，若提供则仅返回符合词性要求的关键词 ,查看:jieba词性表.txt

```python
import jieba
# 必须引入的使用就引入jieba.analyse,才能使用.  不能导入jieba ,然后使用jieba.analyse
import jieba.analyse
text = " 因此电影开头谜语人从用地毯铲杀市长开始就在为验证蝙蝠侠的身份布局了"

seg_list = jieba.cut(text, cut_all=True)
print("Full Mode: " + " ".join(seg_list))  # 全模式

seg_list = jieba.cut(text, cut_all=False)
print("Default Mode: " + " ".join(seg_list))  # 精确模式

# 提取关键字
seg_list = jieba.analyse.extract_tags(text)
print("analysea extract: " + " ".join(seg_list))  # 分析提取

# 提取5个
seg_list = jieba.analyse.extract_tags(text, topK=5)
print("analysea extract topK: " + "  ".join(seg_list))  # 分析提取

# 返回权重
```

```
seg_list = jieba.analyse.extract_tags(text, topK=5, withWeight=True)
print("analysea extract withWeight: " + str((seg_list)))  # 分析提取

# 词性过滤 返回 n名词
seg_list = jieba.analyse.extract_tags(text, allowPOS=("n"))
print("analysea extract allowPOS: " + str((seg_list)))  # 分析提取

```

```
#import jieba
# 必须引入的使用就引入jieba.analyse,才能使用.  不能导入jieba ,然后使用jieba.analyse
import jieba.analyse
text = "皇后区的友好邻居变成了宇宙的，现实世界资本的侵蚀之下，只有哥谭还维持着它的地域性"

#seg_list = jieba.analyse.extract_tags(text, allowPOS=("v"))
seg_list = jieba.analyse.extract_tags(text)
#print("analysea extract allowPOS: " + str((seg_list)))  # 分析提取
# 将列表拼接成字符串
txt_str = " ".join(seg_list)
wordcloud = WordCloud(font_path="C:/Windows/Fonts/simfang.ttf",
                                      collocations=False,
                background_color="black",
                width=800,
                height=600,
                max_words=50).generate(txt_str)
# 生成图片
image = wordcloud.to_image()
# 展示图片
image.show()
# 写入文件
wordcloud.to_file("tag.jpg")
```

```

```

词云图案例:

data/新蝙蝠侠评论.txt --->保存采集的1000条评论

绘制词云图

```
import jieba.analyse
from wordcloud import WordCloud
from matplotlib import pyplot as plt

#绘制画布
fig = plt.figure(dpi=200)
# 获取文本text
text = open("新蝙蝠侠评论.txt",encoding="utf-8").read()

text_cut = jieba.analyse.extract_tags(text,topK=100,allowPOS=("a"))

new_text=' '.join(text_cut)

# 生成词云
wordcloud = WordCloud(font_path="C:/Windows/Fonts/simfang.ttf",
                      collocations=False,
                      background_color="black",
                      width=1200,
                      height=600,
                      max_words=100).generate(new_text)

#生成图片
image = wordcloud.to_image()
# 展示图片
image.show()
# 写入文件
wordcloud.to_file("新蝙蝠侠评论.jpg")
```

```

```

```

```