第 2 讲　数据文件整理

1. 文件基本操作

1.1 文件打开

Python 通过内置函数 open()打开文件，并返回一个文件对象，语法格式如下：

```
f = open(file[,mode])
```

其中，open 函数调用成功会返回一个文件对象，f 为引用文件的对象的变量，mode 为文件的读写模式。比如打开一个名为“test.txt”的文件，代码示例如下：

```
f = open('test.txt','r')
```

访问模式	含　义
r	以读方式打开
w	以写方式打开，此时文件内容会被清空。如果文件不存在则会创建新文件
a	以追加的模式打开，从文件末尾开始，如果文件不存在则会创建新文件
r+	以读写模式打开
w+	以读写模式打开
a+	以追加的读写模式打开
rb	以二进制读模式打开
wb	以二进制写模式打开
ab	以二进制追加模式打开
rb+	以二进制读写模式打开
wb+	以二进制读写模式打开
ab+	以二进制读写模式打开

2.2 文件关闭

如果不再使用该文件，需要及时地关闭文件，释放文件对象，否则一旦程序崩溃，很可能导致文件数据丢失。在 Python 中，可以通过调用文件对象的 close 方法来关闭文件，语法格式如下：

```
文件对象.close()
```

2.3 读文件

从文件中读取数据时，可以通过调用文件对象的 read()方法、readline()方法或 readlines()方法实现，具体如下：

（1）使用 read()方法

read()方法从文件当前位置读取指定数量的字符，以字符串形式返回。语法格式如下：

```
str = f.read([size])
```

（2）使用 readline()方法

readline()方法可以从指定文件中读取一行数据。语法格式如下：

```
str = f.readline()
```

（3）使用 readlines()方法

readline()方法可以把文件内容以行为单位一次性读取。readlines()方法会返回一个列表，列表中的元素为文件中的每一行数据。语法格式如下：

```
list = f.readlines()
```

2.4 数据写入

要将数据写入文件，python 提供了 write()方法和 writelines()方法，来实现文件写入功能。

（1）write()方法

参数说明：

f：是读取内容的文件对象。

list：读取的内容返回到字符串列表 list 中。

调用文件对象 write()方法，可以在文件当前位置写入字符串，并返回写入字符个数。语法格式如下：

```
f.write(str)
```

（2）writelines()方法

使用 writelines()方法可以向文件中写入字符串序列。语法格式如下：

```
f.writelines(seq)
```

边学边练：
图像的爬取和保存
（1）将当前工作目录设为“D:\\python”
（2）在当前工作目录下创建新的目录 data
（3）编写程序，保存在当前工作路径下，并实现将第 1 讲中爬取下来的流浪狗图片，保存到 data 目录下。

2.文件目录操作

1.创建目录

os 模块中有两个函数可以实现目录的创建，分别是 mkdir()函数和 makedirs()函数。os.mkdir()函数用于创建单个目录，语法格式如下：

```
os.mkdir(path)
```

2.获取和更改当前目录

通过调用 os.getcwd()函数可以获取当前工作目录。语法格式如下：

```
os.getcwd()
```

getcwd()函数不带参数，返回表示当前工作目录的字符串。

如果需要更改当前工作目录，可以使用 os.chdir()函数。语法格式如下：

```
os.chdir(path)
```

其中，path 参数表示要切换的目标工作目录，如果目标工作目录不存在，则报错。

3.获取目录内容

使用 os 模块的 listdir()函数获得指定目录中文件和目录内容。语法如下：

```
os.listdir(path)
```

参数 path 指定要获得内容目录的路径。函数返回结果为内容列表。

4.检查文件是否存在

对文件进行操作前，往往要先检查文件是否存在。os 模块中的 os.path.exists(path)可以检测文件或文件夹是否存在，path 为要检查的文件路径，可以是绝对路径或相对路径。返回结果为 True/False。示例如下：

```
import os     #导入 os 模块
print(os.path.exists('d:/works/test.txt'))
```

5.文件重命名

os 模块中的 rename()方法可以完成文件的重命名。语法格式如下：

```
os.rename(原文件, 目标文件)
```

原文件和目标文件可以使用绝对路径和相对路径，但必须位于相同的目录中

6.删除文件

使用 os 模块的 remove()函数可以移动文件，语法格式如下：

```
os.remove(文件路径)
```

7.文件和目录合成、文件大小

目录合成：把目录和文件合成一个路径，各路径不包含“/”，自动加上。

```
os.path.join(path1,path2...)：文件和目录名合成
```

文件大小：返回文件大小，不存在返回错误

```
os.path.getsize(path)：返回文件大小
```

8.文件的复制和移动

shutil.copy(src,dst)：复制文件内容以及权限，如果目标文件已存在，则抛出异常

shutil.copyfile(src,dst)：复制文件不复制文件属性，如果目标文件已存在，则直接覆盖。

shutil.move(src,dst)：一定能够文件或文件夹

9.随机选择部分文件

import random

random.sample(sequence, k)：用于截取列表的指定长度的随机数，但是不会改变列表本身的排序

参数：

sequence：一个序列，可以是任何序列：列表，集合，范围等。

k：返回列表的大小

边学边练：

图像文件的整理

将工作目录设为“D:\\python”，在当前工作路径下的工作目录里创建 train 和 test 文件

夹。data 目录下按照 8：2 的比例随机抽取（random.sample()）图片放在子目录下，在 train 目录下的文件按照序号命名为 train_001.jpg，train_002.jpg，train_003.jpg 以此类推 ，test 目录下的文件命名为 test_001.jpg，test_002.jpg，test_003.jpg 以此类推。

3.Matplotlib 操作

Matplotlib 是 Python 的一个扩展库，是一个 Python 2D 绘图库

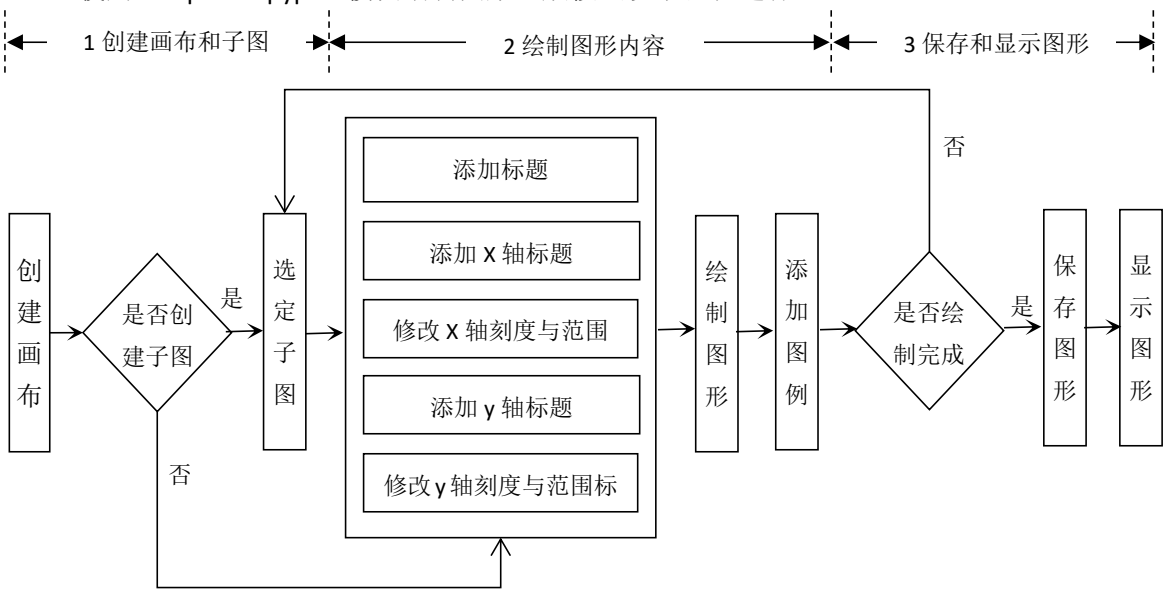
Matplotlib 使用前需进行安装。

```
pip install matplotlib
```

Matplotlib 中 pyplot 模块是一个命令风格函数的集合，运用 matplotlib.pyplot 创建图形时，可以实现图形中创建绘图区域中绘制线条、使用标签装饰绘图等功能。

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

使用 matplotlib.pyplot 模块绘制图形一般按照以下流程进行:



1.创建画布

matplotlib.pyplot 使用 figure()函数创建空白画布，画布中可绘制图表内容、设定图表标签，其格式如下：

```
matplotlib.pyplot.figure(num=None, figsize=None, dpi=None, facecolor=None,
edgecolor=None, frameon=True, FigureClass=<class 'matplotlib.figure.Figure'>, clear=False)
```

部分参数的含义如下：

（1）num：可选参数，表示图形的编号或者名称，数字表示编号，字符串表示名称。如果不提供该参数，则会创建一个新窗口，窗口的编号会自增；如果提供该参数，该窗口的编号即为 num 值。

（2）figsize：可选参数，是一个元组（宽度，高度），用于设置画布的尺寸，以英寸为单位。

2.创建多个子图

很多时候需要在一块画布上绘制多个图形。Figure 对象允许被划分为多个绘图区域，每个区域都拥有属于自己的坐标系统，称为子图。在画布上创建多个绘图区域，使用 subplot()函数实现，subplot()函数的基本格式如下：

```
matplotlib.pyplot.subplot(nrows, ncols, index)
```

其中的参数含义如下：

（1）nrows，ncols：表示一张图被分为 nrows×ncols 个区域。

（2）index：表示子图所处的位置，起始位置索引为 1，即 1<=index<=nrows×ncols。

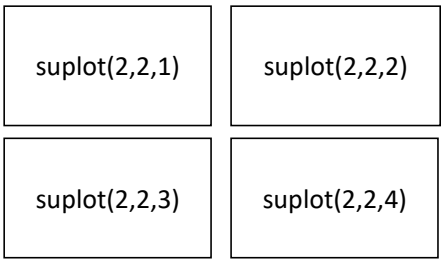


图 9-12 画布分成 2×2 的矩阵区

3.添加画布内容

绘图时，可以添加标题、坐标轴刻度、坐标轴名称等标签用以标注图表的信息，pyplot 模块提供了为图形添加标签的函数，常用的如下表所示（matplotlib.pyplot 简写为 plt）：

表 9-8 NumPy 数组数据类型

函数名称	功能说明
plt.title('标题名称')	设置当前图表的标题
plt.xlabel('x 轴名称')	设置当前图表 x 轴的标签名称
plt.ylabel('y 轴名称')	设置当前图表 y 轴的标签名称
plt.xticks([刻标数组,刻度标签,rotation=旋转角度])	指定 X 轴刻度的标签与取值，刻度默认是数值，但可以替换成标签形式（刻度标签是可选参数），旋转角度指 x 轴标签在水平方向上顺时针旋转的角度
plt.yticks([刻标数组,刻度标签,旋转角度])	指定 X 轴刻度的标签与取值，刻度默认是数值，但可以替换成标签形式（刻度标签是可选参数），旋转角度值标签或刻度值在水平方向上顺时针旋转的角度
plt.xlim((x 轴范围))	设置或获取当前图表 x 轴范围
plt.ylim((y 轴范围))	设置或获取当前图表 y 轴范围
plt.legend([图例])	在轴上防止一个图例

3.图像的读取

```
img=plt.imread(path) # 读取图像
plt.imshow(img) # 显示图像
```

4.保存和显示图表

创建好图表后，可以将其直接显示，也可以以图片的形式保存在某目录下，这些用 Matplotlib 相关函数都可以实现。

（1）显示图表。显示图表的功能已经在前面的例题中应用过了，用法如下：

```
plt.show()
```

边学边练：

图像文件读取与显示

将当前工作路径下 data 文件夹的图像读取，并用子图形式展示出来。