“老师，我刚才写好的程序，重新运行后，之前的数据就都没有了。”

“我的也是这样。每次运行程序都需要把数据重新生成一次。如果能保留前一次的运行结果就好了。”

“小萌、小帅，你们有没有想过怎么可以把上一次运行的结果保留下来，等下一次运行时可以直接拿来用？”

小萌、小帅和老师讨论的问题，在 Python 中可以用文件来处理。

5.1　什么是文件

“小萌、小帅，你们能说出计算机中的文件都有哪些吗？”

“我知道 Word 文件、文本文件。”

“我还知道音乐文件 MP3，还有图片文件 JPG。”

文件用来存放数据。它可以存放在硬盘上，数据永久保留；可以包含任何数据内容。用文件来组织和表达数据更灵活，也更方便。文件有两种类型：文本文件和二进制文件。两种文件的数据组织方式不同。

文本文件是指存放形式为长字符串的文件，它们的编码方式统一，如 txt 文件；二进制文件是指存放形式不能看成是字符串，都是由 0 和 1 组成的文件，没有统一的字符编码，不能展示为字符，具体的组织形式与文件的用途有关，如音乐文件 MP3、图片文件 JPG 等。

【例 5-1】 输出二进制文件和文本文件。文件“祖国 .txt”的内容为“我爱我的祖国！”。

```
txtf=open('祖国 .txt','rt')      # 以文本文件形式读入文件
print('----- 输出文本文件 -----')
print(txtf.readline())
txtf.close()
binf=open('祖国 .txt','rb')      # 以二进制文件形式读入文件
print('----- 输出二进制文件 -----')
print(binf.readline())
binf.close()
```

运行结果为：

```
----- 输出文本文件 -----
我爱我的祖国!
----- 输出二进制文件 -----
b'\xce\xd2\xb0\xae\xce\xd2\xb5\xc4\xd7\xe6\xb9\xfa\xa3\xa1'
```

可以看到，文本文件经过编码，形成了字符串，而二进制文件由一串 0 和 1 组成，文件中的每个字符对应两个字节（16 位二进制，输出时为以 x 开头的十六进制数）。

Python 中，对文本文件和二进制文件的处理方式是一样的，包括：文件打开、文件关闭、读文件、写文件等。文件的处理流程也都一样：打开文件——操作文件——关闭文件。

【问题 5-1】 为什么程序中变量的值不能永久保留，而文件可以呢?

“小萌、小帅，当你们需要编辑一篇文章的时候，你们会做哪些操作？”

“先找到保存文章的文件，打开文件，就可以编辑了。”

“编辑完了之后，还要保存，最后关闭打开的文件。”

5.2　文件的打开和关闭

1. 打开和关闭文件

任何文件，都需要先打开才能进行下一步的操作，操作结束需要关闭文件。处于打开状态的文件只能由打开它的程序操作，其他程序不能操作，只有当该程序关闭后，其他程序才能打开并操作。

通过 open( ) 函数和 close( ) 函数，可以打开和关闭文件，改变文件的状态，如图 5-1 所示。

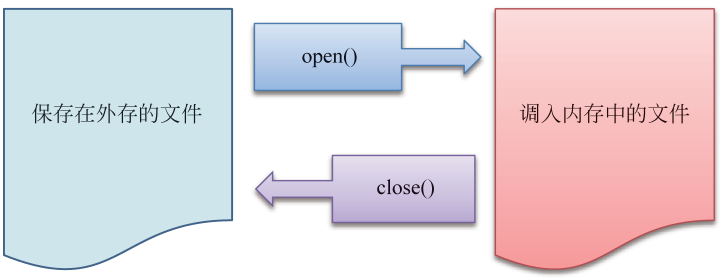

图 5-1　文件的打开和关闭操作

open( ) 和 close( ) 都是 Python 的内置函数，可以直接使用。

使用格式：

< 变量名 >=open(< 文件名 >, < 打开模式 >)

说明：文件名指的是需要打开的文件的名字，打开模式指的是以什么方式打开文件，即打开后文件可以做些什么操作，如表 5-1 所示。

表 5-1　文件的打开模式

打开模式	说　明
r	默认值，只读模式，如果文件不存在，则返回异常
w	覆盖写模式，如果文件不存在就创建一个新文件，否则就完全覆盖原来的内容
x	创建写模式，文件不存在就创建新文件，否则程序报错
a	追加写模式，文件不存在就创建新文件，否则在已有文件末尾追加新内容
b	二进制文件模式，可以与 r/w/x/a 组合使用
t	默认值，文本文件模式，可以与 r/w/x/a 组合使用
+	与 r/w/x/a 组合使用，在原功能基础上同时增加读写功能

【例 5-2】 文件操作过程。

```
f=open('古诗.txt')   # 打开文件，省略了"打开模式"，默认为"只读模式"
for i in f:
    print(i)
f.close()              # 文件操作完后，关闭文件
```

运行结果：

```
人生得意须尽欢，莫使金樽空对月
```

【问题 5-2】 下列有关 Python 文件操作的叙述，不正确的是（　　）。

A. open 是 Python 中打开文件的内置函数

B. 文件使用完毕，应使用打开文件对象的 close( ) 方法将其关闭

C. Python 能够以文本形式或二进制形式打开文件进行操作

D. Python 打开的文件不能既进行读操作，又进行写操作

2. 文件的路径

“老师，‘古诗 .txt’这个文件在哪儿？”

“嗯，小帅说的是文件的路径问题。这在使用文件的时候很重要。”

文件的路径其实就是文件存放的位置。我们都有过找东西的经历，寻找东西关键要知道东西在哪里，知道位置，找起来就又快又准了。计算机上有很多的文件，使用时，同样需要知道文件存放的位置。文件的位置有两种表示方式：相对路径和绝对路径。

1）相对路径

上面的程序中，用 open(' 古诗 .txt') 打开文件时，使用的就是相对路径。也即要打开的文件“古诗 .txt”和打开它的程序文件“5-2.py”在同一个文件夹中。

2）绝对路径

如果要打开的文件和使用它的文件不在同一个文件夹，需要写出要打开文件的存放位置，这就是绝对路径。

在资源管理器中，文件、文件夹、绝对路径的分布如图 5-2 所示。

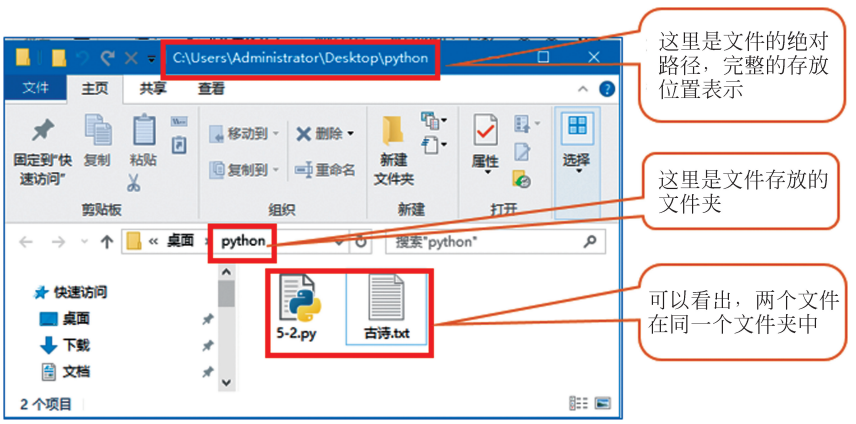

图 5-2　资源管理器中文件、文件夹、绝对路径的分布

5.3 文件的读写操作

文件的读写操作，指的是对文件的输入和输出。在前面的程序中，输入基本是从键盘或者直接在程序中指定，输出都是输出到屏幕上。文件的读写，都是从文件获取数据，再输入到文件中保存。

1. 读文件

读文件是从文件中获取程序运行需要的数据。根据获取数据的多少，有如表 5-2 所示几种读文件的方法。

表 5-2　读文件的方法

方　法	说　明
f.read(size=n)	从文件中读入前 n 个长度的字符串或字节流，若没有该参数，则读入全部文件内容
f.readline(size=n)	从文件中读入一行内容，如果有参数，则读入该行前 n 个长度的字符串或字节流
f.readlines()	从文件中读入所有行，以每行为元素，形成一个列表

【例 5-3】 从《赤壁怀古》中读出内容。

```
name=input('请输入需要打开的文件：')      # 输入需要打开的文件名称
f=open(name+'.txt','r')                  # 把用户输入的文件名加上扩展名
print('----- 带参数的情况 -----')
poem=f.read(3)
print(poem)
f.seek(0)                                # 把文件指针重新定位到文件开头
print('----- 不带参数的情况 -----')
poem=f.read()
print(poem)
f.close()
```

运行结果：

```
请输入需要打开的文件：赤壁怀古
----- 带参数的情况 -----
大江东
----- 不带参数的情况 -----
大江东去，浪淘尽，千古风流人物。
故垒西边，人道是，三国周郎赤壁。
乱石穿空，惊涛拍岸，卷起千堆雪。
江山如画，一时多少豪杰。
遥想公瑾当年，小乔初嫁了，雄姿英发。
羽扇纶巾，谈笑间，樯橹灰飞烟灭。
故国神游，多情应笑我，早生华发。
人生如梦，一尊还酹江月。
```

第一次使用 f.read(3) 带了参数 3，表示读入文件中前 3 个字符，因此输出“大江东”，第二次使用 f.read()，不带参数，此时读入了整个文件的内容。

还记得小时妈妈指读（见图 5-3）绘本吗?

“妈妈的手指头就像指针，指向文件中的字符。”

“f.seek()就用来移动指针，移动到需要的位置。”

图 5-3　指读

说明：

f.seek(0)——把文件指针移动到文件的开头。

f.seek(1)——文件指针指向当前文件的位置。

f.seek(2)——把文件指针移动到文件末尾。

【例 5-4】 读《春晓》并逐行打印。

```
name=input('请输入要打开的文件：')
f=open(name+'.txt','r')
print('----- 全部读入文件并逐行打印 -----')
poem=f.readlines()
print(poem)
for i in poem:
    print(i)
f.seek(0)
print('----- 直接处理 -----')
for i in f:
    print(i)
f.close()
```

运行结果：

```
请输入要打开的文件：春晓
----- 全部读入文件并逐行打印 -----
['春眠不觉晓，处处闻啼鸟。\n', '夜来风雨声，花落知多少。']
春眠不觉晓，处处闻啼鸟。
夜来风雨声，花落知多少。
----- 直接处理 -----
春眠不觉晓，处处闻啼鸟。
夜来风雨声，花落知多少。
```

使用 f.readlines( ) 时，把文件内容全部都读入到一个列表中，列表的每个元素是文件每一行的内容，再通过 for 循环来遍历列表并输出列表内容。这样做的缺点是，如果文件太大，内容很多时，全部一起调入会非常占用内存，而且影响程序的执行速度。程序中的第二种方式就是直接处理，用 for 循环遍历文件，一次读入一行并输出。

练一练

【问题 5-3】　文件 a.txt 中的一行数据为 "Hello,world."。

运行语句 f=open("a.txt","rt") 后，下列叙述正确的是（　　）。

A. f. readall( ) 的值为 'Hello,world.'

B. f. readlines( ) 的值为 'Hello,world.'

C. f. readable( ) 的值为 True

D. f. readline (1) 的值为 'Hello,world.'

2. 写文件

写文件就是把数据写入文件中保存，写入数据可以是程序的运行结果。Python 提供了两种写文件的方法，如表 5-3 所示。

表 5-3　写文件的方法

方　法	说　明
f.write(s)	向文件写入一个字符串
f.writelines(lines)	将一个元素全为字符串的列表写入文件

【例 5-5】　将古诗《绝句》写入指定文件中。

```
name=input('请输入需要操作的文件：')
f=open(name+'.txt','w')
poem=input('需要写入文件的内容是什么？ ')
f.write(poem)                    # 把需要写入的内容作为字符串写入文件
f=open(name+'.txt','r')     # 查看写入后文件内容，以读方式再打开文件
f.seek(0)
for i in f:
    print(i)
f.close()
```

运行结果：

```
请输入需要操作的文件：绝句
需要写入文件的内容是什么？两个黄鹂鸣翠柳，一行白鹭上青天。
两个黄鹂鸣翠柳，一行白鹭上青天。
```