第5章

输入输出与文件处理

在程序设计中，经常需要从键盘或者文件中获取数据，处理结束后又需要将运行结果输出到显示器或者文件中去。

5.1 标 准 输 入

Python 语言为标准输入提供了一个系统函数 input()，专门用来从键盘获取数据。

5.1.1 默认格式

```
变量 = input([提示信息])
```

功能：用来从键盘一次性读入一行数据，输入的数据以 Enter 键结束，但是不包括 Enter 键本身，读入的数据是字符串类型，然后赋值给变量。

提示信息是一个字符串，用来对用户进行输入前的必要提示，是一个可选项。

【程序源码】(LX0501.py)

```
#不断地从键盘给变量输入数据，然后输出变量的值和类型，直到直接按回车键结束

a = input("请输入一个数据:")
while a != '':
    print(a, type(a))
    a = input("请输入一个数据:")
```

【运行结果】

```
请输入一个数据:123
123 <class 'str'>
请输入一个数据:1234.56789
1234.56789 <class 'str'>
请输入一个数据:China
China <class 'str'>
请输入一个数据:1+2j
```

```
1+2j <class 'str'>
请输入一个数据:True
True <class 'str'>
请输入一个数据:
```

5.1.2 具体类型格式

(1) 整数类型。

```
变量 = int(input([提示信息]))
```

功能：为变量从键盘读入一个整数类型的数据，如果输入的数据格式有误，则抛出异常。

(2) 浮点数类型。

```
变量 = float(input([提示信息]))
```

功能：为变量从键盘读入一个浮点数类型的数据，如果输入的数据格式有误，则抛出异常。

(3) 复数类型。

```
变量 = complex(input([提示信息]))
```

功能：为变量从键盘读入一个复数类型的数据，如果输入的数据格式有误，则抛出异常。

(4) 布尔类型。

```
变量 = bool(input([提示信息]))
```

功能：为变量从键盘读入一个布尔类型的数据，如果输入的数据格式有误，则抛出异常。

这 4 个输入格式的实质是将输入的字符串通过类型转换函数转换成了相应的类型，再赋值给变量。

【程序源码】(LX0502.py)

```
1.  a1 = input("请输入一个字符串类型的数据:")
2.  print(a1, type(a1))
3.  a2 = int(input("请输入一个整数类型的数据:"))
4.  print(a2, type(a2))
5.  a3 = float(input("请输入一个浮点数类型的数据:"))
6.  print(a3, type(a3))
7.  a4 = complex(input("请输入一个复数类型的数据:"))
8.  print(a4, type(a4))
9.  a5 = bool(input("请输入一个布尔类型的数据:"))
10. print(a5, type(a5))
11. a6 = int(input("请输入一个整数类型的数据:"))
12. print(a6, type(a6))
```

【运行结果】

```
请输入一个字符串类型的数据:China
China <class 'str'>
请输入一个整数类型的数据:123
123 <class 'int'>
请输入一个浮点数类型的数据:1234.56789
1234.56789 <class 'float'>
请输入一个复数类型的数据:1+2j
(1+2j) <class 'complex'>
请输入一个布尔类型的数据:True
True <class 'bool'>
请输入一个整数类型的数据:1234.56789
```

当程序运行至第 11 行时,输入 1234.56789,不符合整数类型的格式要求,出现了错误,抛出了如下异常信息。

```
发生异常: ValueError       (note: full exception trace is shown but execution is
paused at: <module>)
invalid literal for int() with base 10: '1234.56789'
```

5.1.3 自动类型格式

```
变量 1, 变量 2, … ,变量 n = eval(input([提示信息]))
```

功能:

(1) 自动把从键盘读入的数据转换成合适的数据类型,并赋值给变量;

(2) 如果要给字符串变量输入数据,则需要在输入时给字符串加上定界符;

(3) 可以一次性给多个变量读入数据,在输入时数据之间默认用逗号分隔,也可用其他符号分隔,但是需要用字符串的 split()方法进行处理;

(4) 如果用逗号分隔,一次输入多个数据只赋值给一个变量,则这个变量是元组类型。

【程序源码】(LX0503.py)

```
1.  #不断地从键盘给变量输入数据,然后输出变量的值和类型,直到输入一个空字符串结束
2.
3.  a = eval(input("请输入一个数据:"))
4.  while a != '':
5.      print(a, type(a))
6.      a = eval(input("请输入一个数据:"))
```

【运行结果】

```
请输入一个数据:123
123 <class 'int'>
请输入一个数据:1234.56789
1234.56789 <class 'float'>
请输入一个数据:1+2j
```

```
(1+2j) <class 'complex'>
请输入一个数据:True
True <class 'bool'>
请输入一个数据:"China"
China <class 'str'>
请输入一个数据:""
```

【程序源码】(LX0504.py)

```
a1, a2, a3, a4, a5 = eval(input())
print(a1, type(a1))
print(a2, type(a2))
print(a3, type(a3))
print(a4, type(a4))
print(a5, type(a5))
```

【运行结果】

```
123,1234.56789,1+2j,False,"China"
123 <class 'int'>
1234.56789 <class 'float'>
(1+2j) <class 'complex'>
False <class 'bool'>
China <class 'str'>
```

【程序源码】(LX0505.py)

```
#input()函数一次输入多个数据时,数据之间也可以不用逗号分隔

a = eval(input())
print(a, type(a))

a1,b1 = eval(input())
print(a1, b1)
print(type(a1), type(b1))

a2,b2 = input().split(' ')
print(a2, b2)
print(type(a2), type(b2))

a3,b3 = input().split('#')
print(a3, b3)
print(type(a3), type(b3))
```

【运行结果】

```
123,456,789
(123, 456, 789) <class 'tuple'>
123,True
123 True
```

```
<class 'int'> <class 'bool'>
China Shanghai
China Shanghai
<class 'str'> <class 'str'>
China#Shanghai
China Shanghai
<class 'str'> <class 'str'>
```

5.2 标准输出

Python 语言为标准输出提供了一个系统函数 print(),专门用来向终端窗口输出程序的运行结果。

5.2.1 简单输出

```
print([表达式 1, 表达式 2, …, 表达式 n][, sep = 分隔符][, end = 结束符])
```

功能:

(1) 在终端窗口中依次输出表达式的值,值与值之间默认用一个空格分隔,输出所有表达式的值后默认输出一个“回车+换行”;

(2) sep 参数是一个字符串,用来分隔值与值;

(3) end 参数也是一个字符串,在所有表达式的值输出之后输出;

(4) print()只输出一个空行,即“回车+换行”。

【程序源码】(LX0506.py)

```
1.  a = 10
2.  b = 20
3.  c = 30
4.  print(a, b, c)
5.  print(a, b, c, sep='#')
6.  print(a)
7.  print(b)
8.  print(c)
9.  print(a, end=',')
10. print(b, end=',')
11. print(c)
```

【运行结果】

```
10 20 30
10#20#30
10
20
30
10,20,30
```

5.2.2 格式化输出

```
print(格式字符串. format(表达式 1, 表达式 2, …, 表达式 n))
```

格式字符串由两部分组成,即普通字符和控制字符。

1. 普通字符

普通字符会原样输出,用来修饰输出的内容。

2. 控制字符

```
{ [ 序号 | 键 ][ : 格式控制字符] }
```

(1) 槽:一组“{ }”叫作一个槽,用来匹配一个或多个表达式;

(2) 序号:用序号 0,1,2,…,n−1 分别对应表达式 1, 表达式 2, …, 表达式 n,序号可以自定义,从而改变表达式的输出顺序;

(3) 键:用“key = 表达式”的形式代替序号让槽匹配后面的表达式;

(4) 格式控制字符:以冒号开始引导一个字符串,用来组合控制输出格式,其功能如图 5-1 所示。

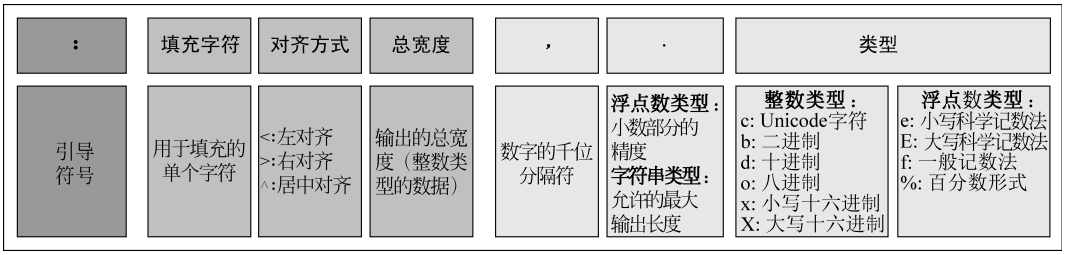

图 5-1 格式控制字符的功能

【程序源码】(LX0507.py)

```
1.  s1 = "China"
2.  d1 = 1023
3.  f1 = 1234.56789
4.
5.  print("{}, {}, {}".format(s1, d1, f1))
6.  print("{0}, {1}, {2}".format(s1, d1, f1))
7.  print("{2}, {1}, {0}".format(s1, d1, f1))
8.  print("{key1}, {key2}, {key3}".format(key1 = f1,key2 = d1, key3 = s1))
9.
10. print("{:#<20}, {:*^20}, {:>20}".format(s1, d1, f1))
```

```
11. print("{:#<20.3}, {:*^20,}, {:>20.2f}".format(s1, d1, f1))
12.
13. print("{0:^10c},{0:^10b},{0:^10d},{0:^10o},{0:^10x},{0:^10X}".format(d1))
14. print("{0:>10e}, {0:>10E}, {0:>10f}, {0:>10%}".format(f1))
15.
16. print("{:^10b},{:^10c},{:^10d},{:^10o},{:^10x},{:^10X}".format(d1,d1,d1,
    d1,d1,d1))
17. print("{:>10e}, {:>10E}, {:>10f}, {:>10%}".format(f1,f1,f1,f1))
```

【运行结果】

```
China, 1023, 1234.56789
China, 1023, 1234.56789
1234.56789, 1023, China
1234.56789, 1023, China
China#################, ********1023********,          1234.56789
Chi##################, *******1,023********,              1234.57
    ?     ,1111111111,   1023   ,   1777   ,   3ff    ,   3FF
1.234568e+03, 1.234568E+03, 1234.567890, 123456.789000%
1111111111,    ?     ,   1023   ,   1777   ,   3ff    ,   3FF
1.234568e+03, 1.234568E+03, 1234.567890, 123456.789000%
```

5.3 文件读写

在程序设计过程中，变量的值只是在内存中临时保存。当程序运行结束或者计算机断电时，内存中存储的所有数据将被自动清除。如果想永久性保存数据，就要使用文件进行保存。

5.3.1 文件

文件是存储在长期储存设备上（例如硬盘、光盘和 U 盘等）的相关信息的集合，其特点是所存信息可以长期、多次使用，也不会因为断电而消失。存储在计算机上的一个程序、一个文档、一张图片、一首音乐、一部电影等都是一个文件。

在 Python 语言中，支持读写的文件有两类，即文本文件和二进制文件。

1. 文本文件

文本文件是基于字符编码的文件，常用的编码有 ASCII 编码和 Unicode 编码。文本文件是一种典型的顺序文件，其逻辑结构属于流式文件。

2. 二进制文件

二进制文件是基于值编码的文件，用户一般不能直接读懂二进制文件，需要通过相应的软件才能展示。不同类型的文件，其编码格式不同，比如图像文件 JPEG（Joint Photographic Experts Group，联合图像专家组）格式和 PNG（Portable Network Graphics，

便携式网络图形)格式,其实质就是图像文件的编码格式不同。

文本文件和二进制文件的定义只是逻辑上的定义,不是物理的区分,计算机中存储的所有文件最终都是以 0 和 1 的二进制形式存储的。

5.3.2 文件处理流程

在程序设计中,文件的处理(读文件或写文件)都要遵循打开文件、处理文件、关闭文件的流程,如图 5-2 所示。

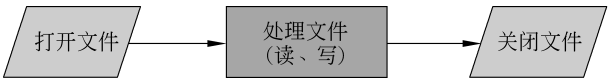

图 5-2 文件处理流程

5.3.3 打开和关闭文件

1. 打开文件

```
文件对象 = open(文件名[, 打开方式][, encoding="编码方式"])
```

功能:以指定的打开方式打开一个文件,生成一个文件对象,赋值给变量。

(1) 文件名参数是一个字符串,用来表示文件的路径信息,有两种表示方式,即绝对路径和相对路径。

绝对路径:从分区盘符开始一直到文件名为止的一个完整路径信息,如"D:\\Program Files\\Python36\\Lib\\os.py",需要强调的是使用转义字符"\\"表示"\"。

相对路径:如果打开的文件在当前程序所在文件夹下,或者当前程序所在文件夹的下级文件夹下,则可以使用相对路径,如"os.py"(比如程序文件和读写文件同在 Lib 文件夹下)、"Lib\\os.py"(比如程序文件在 Python36 文件夹下,读写文件在 Lib 文件夹下)。

注意:强烈建议不要使用带中文字符的路径。

(2) 打开方式如表 5-1 所示。

表 5-1 文件的打开方式

序号	打开方式	说明
1	w|wb	以只写方式打开 一个文本文件(w)或二进制文件(wb)。 如果文件已经存在,则会清空文件,如果文件不存在,则会自动创建新文件,文件指针指向文件头
2	r|rb	以只读方式打开一个文本文件(w)或二进制文件(wb)。 文件必须存在,不存在则抛出异常,文件指针指向文件头
3	a|ab	以追加方式打开 一个文本文件(w)或二进制文件(wb)。 如果文件已经存在,则打开文件,文件指针指向文件尾;如果文件不存在,则会自动创建新文件,文件指针指向文件头

续表

序号	打开方式	说　明
4	x|xb	以创建写方式打开 一个文本文件(w)或二进制文件(wb)。 如果文件不存在,则创建文件,文件指针指向文件头;如果文件已经存在,则抛出异常
5	w+|wb+	以读写方式打开 一个文本文件(w)或二进制文件(wb)
6	r+|rb+	以读写方式打开 一个文本文件(w)或二进制文件(wb)
7	a+|ab+	以读写方式打开 一个文本文件(w)或二进制文件(wb)

(3) encoding：文本文件所采用的编码方式。

2. 关闭文件

```
文件对象.close()
```

功能：关闭已经打开的文件。

3. 文件对象常用属性

(1) 文件对象.name：获取文件的文件名。
(2) 文件对象.mode：文件的打开方式。
(3) 文件对象.closed：当前文件是否关闭。

4. 文件指针

文件指针是指在打开文件进行读写时,用来指示其文件内容具体位置的一个指针,随着读写文件操作的进行,文件指针也随之自动发生变化,如图 5-3 所示。虽然文件指针指示位置可以发生变化,但是文件指针有且仅有一个。不同的打开方式,文件指针的初始位置也不相同。

图 5-3　文件指针

5.3.4　写文件

1. 写文本文件

文本文件的写入就是把字符串以某种字符编码方式写入文件。

(1) print([表达式 1,表达式 2,…,表达式 n][, sep=分隔符][, end=结束符], file=文件对象)。

功能：通过 print()函数的方式将表达式的值以字符串的形式写入文本文件中,不会在终端窗口中显示,字符串中的字符支持转义字符。

(2) 文件对象.write(字符串)。

功能：通过 write()方法将字符串写入文本文件中,字符串的字符支持转义字符。

(3) 文件对象.writelines(字符串类型列表)。

功能：通过 writelines()方法将一个字符串类型列表(列表中的每个元素均是字符串类型)中的每个元素依次写入文本文件中,字符串中的字符支持转义字符。

【程序源码】(LX0508.py)

```
1.  fi = open('data1.txt','w', encoding="utf-8")
2.  print("通过 print 写入新的数据!",file = fi)
3.  fi.write("通过 write 写入新的数据!\n")
4.  l1 = ["通过 writelines","写入新的数据","\n"]
5.  fi.writelines(l1)
6.  fi.close()
7.
8.  fi = open('data1.txt', 'r', encoding="utf-8")
9.  print("文件名称:",fi.name)
10. print("打开方式:",fi.mode)
11. print("是否关闭:",fi.closed)
12. fi.close()
13. print("是否关闭:",fi.closed)
```

【运行结果】

```
文件名称: data1.txt
打开方式: r
是否关闭: False
是否关闭: True
```

直接在操作系统的“资源管理器”中打开“data1.txt”文件,内容如图 5-4 所示。

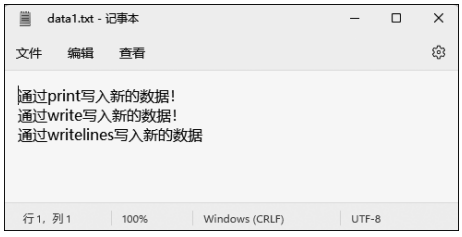

图 5-4 文本文件的内容

2. 写二进制文件

二进制文件以字节型字符串的形式进行读写。