第5章 列表与元组

5.1 本章语法

1. 列表赋值与输入输出

列表是将一组数据放在一对方括号([])中,以逗号分隔,即定义一个列表:

```
ls1 = ['physics', 'chemistry', 1997, 2000]
ls2 = [1, 2, 3, 4, 5 ]
ls3 = ["a", "b", "c", "d"]
```

列表中每个数据称为"元素",数据的个数称为"长度"。不包含任何元素的列表称为空列表,即[ ]。列表的元素可以是相同类型,也可以是不同的类型。

列表的输入和输出最简单的方式是:

- 输入:采用**列表=input()**,从键盘输入的是方括号([])括起来的列表,元素以逗号分隔,如图 5.1 所示。
- 输出:采用 **print 列表**,输出完整的列表,即方括号([])括起来的列表,元素以逗号+空格作分隔,如图 5.1 所示。

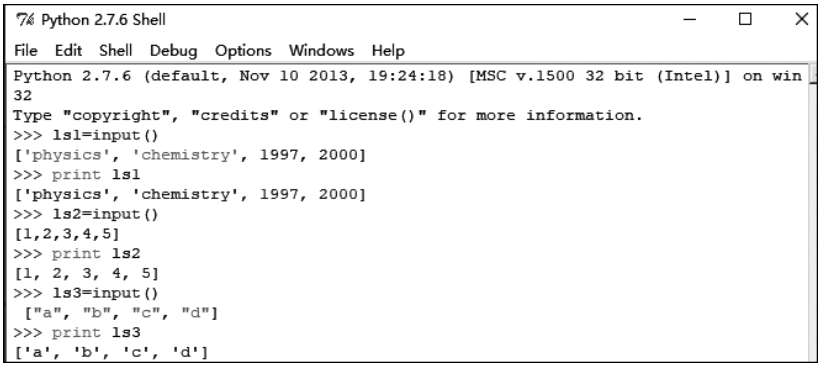

图 5.1 列表的整体输入输出界面

列表输入和输出比较复杂的方式是:

- 输入:使用 raw_input 函数以字符串形式接受数据,然后利用 map 函数和字符串的 split 方法将其转换为数值型列表,此时输入时不带方括号,以逗号或空格进行

分隔的多个数，详细用法参见例 5.1 的方法 2 和方法 3。

- 输出：列表的输出通过遍历方式，这与字符串遍历输出一样，可以用两种方式，详细用法参见例 5.1 的方法 4 和方法 5。

2. 列表操作

列表元素有序存放，列表的序号也分为正向序号和反向序号。列表元素可以按序号访问，列表的切片、内置的运算符（+、*、in、==）以及内置处理方法（例如 findindex 方法、count 方法等）与字符串操作一样，就不详细描述了。

这里介绍列表与字符串不同的操作。列表与字符串不同在于列表支持修改、添加和删除操作，即列表是可变的。

（1）修改元素。

语法格式：

列表名[索引] = 新值

注意：序号索引方向分为正向或反向。详细用法参见例 5.2。

（2）添加元素：有 2 个方法。

- 尾部追加元素：append 方法，详细用法参见例 5.3。
- 指定位置插入元素：insert 方法。

（3）删除元素：有 3 个方法。

- 按索引删除元素：del 命令。
- 按索引删除元素：pop 方法，pop 方法删除元素同时会返回该元素，缺省索引参数时，pop 默认删除最后一个元素。
- 按值删除元素：remove 方法，列表中包含多个待删除元素时，remove 删除索引值相对较小的那个。

注意：命令和方法的区别；已知待删除元素的索引时，可使用 del 命令和 pop 方法；pop 方法对于删除列表最末尾元素最为简单方便；明确知道待删除元素值时，用 remove 方法更为简单。与 del 命令和 remove 方法不同，pop 方法在删除元素的同时会“弹出”这个被删除的元素，如果需要可以用一个变量“接住”它，以便进行进一步的后期操作。

（4）其他常用操作：

- sum 内置函数：计算列表的元素和。注意：对列表使用 sum 函数时，列表必须是数值型的。
- 排序：python 中提供 sort 方法或 sorted 函数用于列表的排序，二者使用方法和所带参数（即 key 和 reverse）含义一致，但是 sort 方法排序会改变原来的列表，而 sorted 函数排序是生成新的有序列表，不改变原来的列表。详细用法参考例 5.4 的 4 种方法。

注意方法与函数的使用区别：方法是 a.b 的形式，a 是对象名，b 是方法名；而函数直接使用函数名。

- 复制：即生成“一模一样”的列表。使用 copy 方法深复制，两个列表有独立空间；通过列表之间的赋值操作是浅复制，两个列表共享空间。注意：copy 方法和赋值操作都能得到“一样”的列表，但是两者的实现机制有本质区别。
- 列表的删除：del 后直接跟列表名，则将彻底删除该列表对象。
- 列表的清空：del 列表名[：]，经过删除“所有元素”的 del 操作后，列表中不包含任何元素，但是仍保留其列表的本质。

3. 元组

元组与列表类似，也是用来存放一组相关的数据。两者的不同之处主要有两点：

- 元组使用圆括号()，列表使用方括号[]。
- 元组是不可变的序列，不能修改、删除和插入元素；而列表则是可变序列。

不带“[]”的输入转换为元组，即多个数输入以逗号隔开，默认为元组，如图 5.2 所示。

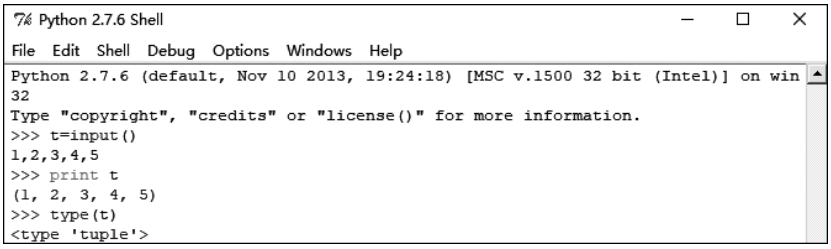

图 5.2　元组的整体输入输出界面

元组输入输出用法与列表类似，区别在于元组是用“()”括起来的多个元素，但是圆括号不是必须的，只要将各元素用逗号隔开，Python 就会将其视为元组。

元组的元素不能修改。可以将元组理解为不能修改的“列表”，因此列表中不涉及元素修改的操作都适用于元组。

使用元组的好处：

- 元组比列表操作速度快。如果定义了一个值的常量集，并且唯一的操作是不断地遍历它，那么使用元组代替列表。
- 如果对不需要修改的数据进行 “写保护”，可以使代码更安全，此时使用元组而不是列表，就如同拥有一个隐含的 assert 语句，说明这一数据是常量。如果必须要改变这些值，则需要执行元组到列表的转换。

4. 序列转换函数

本章介绍的列表、元组和前面学习的字符串都属于 Python 的一种基本数据类型——序列(sequence)。序列的最大特点是元素的有序性，所以序列都是通过序号来访问元素的。

序列是 Python 中最基本的数据结构。序列中的每个元素都分配一个数字表示它的位置或索引：第一个索引是 0，第二个索引是 1，依此类推。Python 已经内置确定序列的长度以及确定最大和最小的元素的函数。Python 有 6 个序列的内置类型，最常见的是列

表和元组。

序列之间可以通过转换函数进行互相转换：

(1) 元组通过 list 函数转换为列表，例如：list((1,2,3))结果是[1, 2, 3]。

(2) 列表通过 tuple 函数转换为元组。

(3) 字符串转换为列表有 3 种方法：

- 使用 list 函数：list 函数转换后字符串中的单个字符依次成为列表元素，例如 list("123")的结果是['1', '2', '3']。
- 使用字符串的 split 方法，利用 split 方法分割字符串生成字符串列表，语法格式为：字符串.split(str="", num=string.count(str))，参数 str 是分隔符，默认为所有的空字符，包括空格、换行(\n)、制表符(\t)等；参数 num 是分割次数，默认为-1，即分隔所有。设 s="1 2 3"，则 s.split()的结果是['1', '2', '3']。
- 内置函数 map 与字符串的 split 方法配合使用，生成数值列表，例如：map(int, '1 2 3 4'.split())的结果是[1, 2, 3, 4]。

5.2 本章示例

【例 5.1】 用不同的方法实现列表的输入输出：输入列表，然后输出列表。

方法 1：

```
ls1=input()
print ls1
```

用例输入：

```
[1,2,3,4,5]
```

用例输出：

```
[1, 2, 3, 4, 5]
```

代码解析：采用了最简单的列表输入输出方式。

方法 2：

```
ls2=map(int,raw_input().split())
```

用例输入：

```
1 2 3 4 5
```

用例输出：

```
[1, 2, 3, 4, 5]
```

代码解析：先使用 raw_input 函数以字符串形式接受数据，然后利用 map 函数和字符串的 split 方法将其转换为整数类型的数值列表。语句 map(int,raw_input().split())

输入多个数，空格分隔。

方法 3：

```
ls3=map(int,raw-input().split(','))
print ls3
```

用例输入：

```
1,2,3,4,5
```

用例输出：

```
[1, 2, 3, 4, 5]
```

代码解析：先使用 raw_input 函数以字符串形式接受数据，然后利用 map 函数和字符串的 split 方法将其转换为整数类型的数值列表。语句 map(int,raw_input().split(',')) 输入多个数，逗号分隔。

方法 4：

```
ls=input()
print ls
for c in ls:
    print c,
```

用例输入：

```
[1,2,3,4,5]
```

用例输出：

```
[1, 2, 3, 4, 5]
1 2 3 4 5
```

代码解析：采用列表 ls 作为迭代器进行遍历输出，比较简单。

方法 5：

```
ls=input()
print ls
for i in range(len(ls)):
    print ls[i],
```

用例输入：

```
[1,2,3,4,5]
```

用例输出：

```
[1, 2, 3, 4, 5]
1 2 3 4 5
```

代码解析：采用 range 函数作为迭代器，以列表 ls 长度作为 range 函数的取值范围，

取索引对应的元素进行遍历输出，虽然实现相对复杂，但是可以灵活地控制列表项全部或部分输出。

【例 5.2】 用修改列表元素的方法实现：输入年和月，计算某年某月天数并输出。

思路是 ls 列表中存放一年 12 个月对应的天数，默认 2 月是 28 天，如果闰年，要修改 2 月为 29 天。

```
ls=[31,28,31,30,31,30,31,31,30,31,30,31]
year,month=input()
if ((year % 4 == 0) and (year % 100 != 0)) or (year % 400 == 0):
    ls[1]=29
print ls[month-1]
```

用例输入：

```
2000,2
```

用例输出：

```
29
```

代码解析：通过列表索引方法实现，注意：修改 2 月天数时，ls[1]这条语句的“差 1”处理。

【例 5.3】 用列表的 append 方法实现斐波那契数列构建前 n 项并输出前 n 项。

```
n=input()
ls=[1,1]
for i in range(2,n+1):
    m=ls[i-1]+ls[i-2]
    ls.append(m)
print ls
```

用例输入：

```
5
```

用例输出：

```
[1, 1, 2, 3, 5, 8]
```

代码解析：列表的 append 方法用于在列表末尾添加新的对象。

【例 5.4】 用不同的排序方式实现对列表进行排序。

方法 1：

```
ls=input()
ls.sort()
print ls
```

用例输入：

```
[3,5,-4,-1,0,-2,-6]
```

用例输出：

```
[-6, -4, -2, -1, 0, 3, 5]
```

代码解析：从运行结果看，使用列表的 sort 方法排序 ls，ls 被修改，默认是升序排序。

方法 2：

```
ls=input()
print sorted(ls)
print ls
```

用例输入：

```
[3,5,-4,-1,0,-2,-6]
```

用例输出：

```
[-6, -4, -2, -1, 0, 3, 5]
[3, 5, -4, -1, 0, -2, -6]
```

代码解析：从运行结果看，使用内置函数 sorted，不修改列表 ls，只是对 ls 进行排序，返回一个排好序的新列表，因为运行结果的第二行输出的 ls 还是原来的值。

方法 3：

```
ls=input()
ls.sort(reverse=True)
print ls
```

用例输入：

```
[3,5,-4,-1,0,-2,-6]
```

用例输出：

```
[5, 3, 0, -1, -2, -4, -6]
```

代码解析：从运行结果看，使用列表的 sort 方法排序 ls，ls 被修改。默认是升序排序，这里参数 reverse＝True 表示降序排序。

方法 4：

```
ls=input()
ls.sort(key=abs)
print ls
```

用例输入：

```
[3,5,-4,-1,0,-2,-6]
```

用例输出：

```
[0, -1, -2, 3, -4, 5, -6]
```

代码解析：这里参数 key＝abs 表示按照列表中元素的绝对值大小升序排列，abs 是内置绝对值函数。该语句等价 ls.sort(key＝lambda x：abs(x))，匿名函数用法将在第 6 章介绍。

5.3 本章练习题

本章的练习题可以分为 2 次完成，第 1 次练习侧重基础语法掌握，第 2 次练习侧重复杂些的应用。

列表与元组练习 1

【练习 5.1】 月的英文。

题目描述：从键盘上输入一个数字，输入对应月的英文单词("January","February","March","April","May","June","July","August","September","October","November","December")，当数字不在 1～12 时，输出“Input error”。

用例输入：

```
1
```

用例输出：

```
January
```

用例输入：

```
13
```

用例输出：

```
Input error
```

【练习 5.2】 输出大于均值的数。

题目描述：从键盘上输入 10 个数，计算均值，然后输出均值以及其中大于均值的数。

提示：循环＋变量的方法虽然可以计算均值，但是要输出大于均值的数，必须要将输入的多个数都保存下来，以便与均值进行比较。因此，本题使用循环＋变量的方法无法实现输出大于均值的数。怎么办？如何将输入的多个数都保存下来？需要列表。

用例输入：

```
[21,42,13,54,75,96,77,98,29,100]
```

用例输出：

```
ave=60.5
```

```
75 96 77 98 100
```

【练习 5.3】 山洞取宝。

题目描述：设一个山洞中有 10 个箱子，每个箱子中放着一块重量不等的宝石。进入山洞后，最多只能拿一块宝石，如何实现？

用例输入：

```
[21,42,13,54,75,96,77,98,29,10]
```

用例输出：

```
max=98
```

【练习 5.4】 某年某月有几天。

题目描述：从键盘输入某年某月，然后计算并输出该年该月有几天。

提示：方法 1。闰年条件是四年一闰，百年不闰，四百年再闰，即 if ((year % 4 == 0) and (year % 100 != 0)) or (year % 400 == 0)：

方法 2。import calendar　然后 if calendar.isleap(year)：或者 calendar.monthrange 函数。

用例输入：

```
2020,3
```

用例输出：

```
31
```

【练习 5.5】 查找某个数。

题目描述：从键盘输入 8 个数，查找是否存在 13。如果找到 13，输出 yes，否则输出 no。

用例输入：

```
[21,42,13,54,75,96,77,98]
```

用例输出：

```
yes
```

用例输入：

```
[82,74,53,94,75,86,77,98]
```

用例输出：

```
no
```

【练习 5.6】 查找某个数出现的次数。

题目描述：从键盘输入若干个数，查找 13 出现几次，并输出次数，若无，则输出 no。

用例输入：

```
13,5,9,11,13,15,19
```

用例输出：

```
2
```

用例输入：

```
1,2,3,4
```

用例输出：

```
no
```

【练习 5.7】 删除数。

题目描述：已知列表[61,4,26,8,22,35,7,89,45,1]。从键盘输入整数 n，若 n 在列表中，则删除 n，并输出删除后的列表；若不存在 n 则输出 no。

用例输入：

```
4
```

用例输出：

```
[61, 26, 8, 22, 35, 7, 89, 45, 1]
```

用例输入：

```
13
```

用例输出：

```
no
```

【练习 5.8】 升序排序。

题目描述：从键盘输入若干个整数，然后将这些数进行升序排序，并输出排序后的数。

用例输入：

```
[1,0,4,8,12,65,-76,100,-45,123]
```

用例输出：

```
[-76, -45, 0, 1, 4, 8, 12, 65, 100, 123]
```

【练习 5.9】 插入数。

题目描述：已知一个排好序的列表[1,2,3,6,8,9,12,23,33]，从键盘上输入一个数 n，若列表中已经存在 n，则输出 no，否则将 n 插入到列表中。要求插入 n 后的列表依然有序，输出新列表。

用例输入：

```
15
```