项 目 5

列表与元组应用

在 Python 编程中，经常会出现需要存储、处理一系列相关数据项的情况，例如，一系列相关数字，一个人相关的姓名、年龄和地址等，用简单的数据类型处理起来难度较大，需要定义很多的变量，效率和性能都偏低。此时，可以借助 Python 中的列表(list)或者元组(tuple)来进行数据的批量处理。列表和元组是 Python 编程时经常使用的数据类型，它们提供了存储和操作一系列数据的能力，能有效支持各种应用开发需求。

在本项目中，通过完成三个具体的任务——演讲比赛评分系统、快递超市管理系统和中文数字对照表，来系统学习列表的创建、列表元素的访问、列表元素的修改、元组的创建、元组元素的访问等内容。通过任务的实践，全面而深入地掌握 Python 中列表与元组的应用。

【学习目标】

知识目标

1. 了解常用的组合数据类型。
2. 理解序列的基本概念及序列的特点。
3. 理解列表的基本概念。
4. 掌握列表的常用内置函数和方法。
5. 理解列表推导式的基本概念。
6. 理解元组的基本概念。
7. 理解列表与元组的区别。

能力目标

1. 能够掌握列表的创建。
2. 能够掌握列表元素的访问、列表的循环遍历。
3. 能够掌握使用常用内置函数和方法操作列表。
4. 能够完成列表中增加、修改、删除列表元素等操作。
5. 能够完成嵌套列表的创建、嵌套列表元素的访问。
6. 能够完成元组的创建。
7. 能够完成元组元素的访问。

【建议学时】

4 学时。

任务 1　演讲比赛评分系统设计

【任务提出】

运用 PyCharm 开发工具编写 Python 程序，设计一个简单的比赛评分系统。某演讲比赛有 11 位评委为选手打分，选手的最终得分计算规则是：去掉一个最高分、去掉一个最低分，然后计算剩下所有得分的平均分，即为该选手的最终得分。某位选手最终得分输出结果如图 5.1 所示。

```
-----演讲比赛评分系统-----
去掉一个最高分： 99
去掉一个最低分： 89
选手的最终得分： 95.67
```

图 5.1　演讲比赛评分

【任务分析】

本任务涉及对多个评委的打分进行处理，需要创建列表用于保存和处理所有的得分，具体的任务实施分析如下。

1. 创建 Python 程序 Score.py。
2. 创建一个列表用来保存选手的所有得分。
3. 找到列表中的最高分和最低分。
4. 计算列表中剩下的所有得分的平均值，即为选手的最终得分。
5. 测试运行程序，并通过输出结果检验程序。

【知识准备】

组合数据类型是将多个相同数据类型或不同数据类型的数据组成的一个整体，根据数据组合方式不同，Python 的组合数据类型可分为三种，分别是序列类型、集合类型和字典类型。


视频讲解

5.1　序列

5.1.1　常见的序列类型

在 Python 的内置数据类型中，序列是由一组有序的数据元素排列形成的集合，可以通过元素在序列中的位置对其操作，即通过索引（也称下标）的方式操作序列中的一个或若干数据元素。Python 中常见的序列类型有列表（list）和元组（tuple），也包括前面讲到的字符串（string）类型。

5.1.2 序列的特点

Python 中的序列支持双向索引：正向递增的正数索引和反向递减的负数索引。其中，正向递增的正数索引，自左向右，从 0 开始，第二个元素的索引是 1，以此类推；反向递减的负数索引，自右向左，从－1 开始，倒数第二个元素的索引为－2，以此类推。字符串与列表都有这样的双向索引，如图 5.2 和图 5.3 所示。

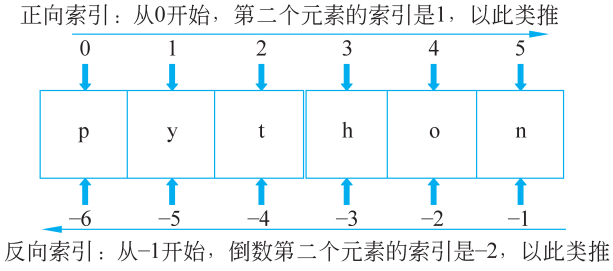

图 5.2 字符串中元素的索引

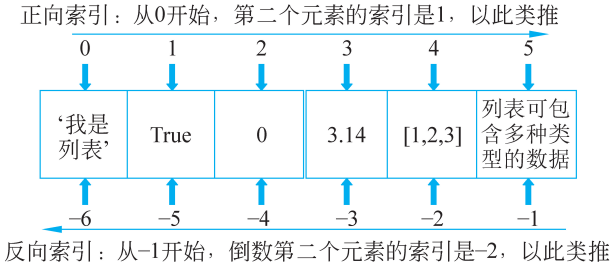

图 5.3 列表中元素的索引

5.1.3 常用的序列操作

序列类型有着一些通用的特定操作，即字符串、列表、元组这些序列类型有着一样的序列元素的操作方式，如下所示。

- 通过正向索引、反向索引访问序列中的一个元素。
- 通过切片访问序列中的部分元素。
- 多个序列相加、序列与数字相乘，然后得到新的序列。
- 使用成员运算符 in 和 not in 来判断序列中是否包含某元素。使用 in 运算符判断时，如果序列中包含某元素则返回结果为 True，否则为 False；而 not in 运算符则正好相反。
- 序列的常用内置函数和方法，如表 5.1 所示。

Python 中的内置函数无须定义，可以直接调用，例如 len(序列)；而方法则需要先创建序列，然后通过序列来调用方法，例如序列.index(元素)。

表 5.1　序列的常用内置函数和方法

函数/方法	语 法 格 式	说　　明
内置函数	len(序列)	计算序列的长度
	max(序列)	返回序列的最大元素
	min(序列)	返回序列的最小元素
	sum(序列，start=0)	计算序列中所有元素的和，第 2 个参数 start 为可选参数，指定和的起始值，默认为 0
方法	序列.index(元素)	查找元素在序列中第一次出现的位置索引
	序列.count(元素)	统计元素在序列中出现的次数


视频讲解

5.2　列表

Python 中列表(list)类似于其他语言的数组，可以由一组不同数据类型的元素组成，且数据元素可以是任意类型，既可以是整型、浮点、布尔等简单数据类型，也可以是字符串、列表、元组、字典等组合数据类型。

5.2.1　列表的创建

创建列表有三种常用的方式：使用中括号[]、内置函数 list()、列表推导式创建列表。

1. 使用中括号

使用中括号创建列表时，将每个列表元素用逗号分隔后，放到中括号“[]”中，语法格式如下所示。

```
列表变量名 =[元素 1,  元素 2,  …]
```

2. 使用内置函数 list()

使用内置函数 list()创建列表时，函数接收的参数必须是一个可迭代类型的数据，例如字符串、列表等类型的数据，语法格式如下所示。

```
列表变量名 =  list(可迭代类型的数据)
```

例 5.1　创建列表，示例代码如下所示。

```
#使用中括号创建列表
list1 =[]                                    #创建一个空列表
list2 =['python', 'java', 'javascript']      #创建元素都是字符串类型的列表
list3 =['我是列表', True, 0, 3.14, [1, 2, 3]]     #创建元素类型不同的列表
#使用内置函数 list()创建列表
list4 =list()                                #使用 list()函数创建一个空列表
list5 =list("abc")                           #使用 list()函数创建三个字母元素的列表
print(list5)                                 #输出: ['a', 'b', 'c']
```

3. 使用列表推导式

列表推导式(list comprehension)可以利用 range()函数、元组、字典和集合等可以迭代循环的数据类型,快速生成一个满足指定需求的列表,语法格式如下所示。

```
[表达式 for  迭代变量  in 可迭代对象 [if 条件表达式]]
```

其中,最外层是中括号[],表示用于生成一个列表;可迭代对象可以是 range()函数、元组、字典和集合;[if 条件表达式]是可选部分。

列表推导式的执行过程是,用 for 循环遍历可迭代对象,逐一将迭代对象的数据元素赋值给迭代变量,然后由包含迭代变量的表达式进行运算,最后将运算结果追加到列表中。如果使用 if 条件表达式,则是仅当 if 条件表达式为 True 时,才将迭代变量用于表达式运算。

例 5.2 创建列表,包含数字 0～9 的平方数,示例代码如下所示。

```
list1 =[x**2 for x in range(10)]
print("list1:", list1)            #输出: list1: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

例 5.3 创建列表,包含整数 0～100 中可以被 3 和 5 整除的所有数字,示例代码如下所示。

```
list1 =[i for i in range(101) if i %3 ==0 and i %5 ==0]
print("list1:", list1)            #输出: list1: [0, 15, 30, 45, 60, 75, 90]
```

5.2.2 访问列表元素

1. 使用索引

通过正向索引、反向索引的方式可以访问列表中一个指定位置的元素,语法格式如下所示。

```
列表变量名[索引]
```

例 5.4 使用索引方式访问列表元素,示例代码如下所示。

```
list =['我是列表', True, 0, 3.14, [1, 2, 3], '列表可包含多种类型的数据']
#列表与字符串的索引规则一样,有正向递增的正数索引,自左向右,从 0 开始
print(list[0])          #输出第一个元素: '我是列表'
print(list[1])          #输出第二个元素: True
#反向递减的负数索引,自右向左,从-1 开始
print(list[-1])         #输出最后一个元素: '列表可包含多种类型的数据'
print(list[-2])         #输出倒数第二个元素: [1, 2, 3]
```

注意:第 1 个元素的索引值为 0,索引的取值范围是 0 至列表的长度减 1,使用的索引值超出索引范围时,Python 会报“index out of range”的索引越界错误。

2. 使用切片

列表支持切片截取列表中的部分元素,得到一个新的列表,语法格式如下所示。

```
列表变量名[起始 : 结束 : 步长]
```

切片截取从起始索引到结束索引的元素，如果步长是大于 1 的数，那么会跳过某些元素。注意，选取的区间左闭右开，不包含结束索引的元素。

例 5.5　使用切片访问列表元素，示例代码如下所示。

```
list =['我是列表', True, 0, 3.14, [1, 2, 3], '列表可包含多种类型的数据']
#在列表后面加上切片[1:5:2]，切片选取索引为 1 到 5，步长为 2，取索引为 1、3 的元素
print(list[1:5:2])          #输出：[True, 3.14]
```

5.2.3　常用列表操作

列表类型是序列的一种，支持使用常用内置函数和序列的方法操作列表，也可以使用列表相加、列表与数字相乘等操作。此外，Python 还提供了 in 和 not in 运算符，用于判断列表中是否包含某元素，使用 in 运算符判断时，如果存在则返回结果为 True，否则为 False。

例 5.6　常用的列表操作，示例代码如下所示。

```
list =['我是列表', True]
list2 =[99, 96, 93, 89, 95]
#1. 多个列表相加、列表与数字相乘，然后得到新的列表
print("list +list2: ", list +list2)
                        #输出：list +list2:  ['我是列表', True, 99, 96, 93, 89, 95]
print("list * 2: ", list * 2)  #输出：list * 2:  ['我是列表', True, '我是列表', True]
#2. 使用 in 和 not in 运算符，判断列表中是否包含某元素
print("3.14 in list: ", 3.14 in list)              #输出：3.14 in list:  False
print("True in list: ",True  in list)              #输出：True in list:  True
print("3.14 not in list: ", 3.14 not in list)      #输出：3.14 not in list:  True
#3. 常用内置函数和方法
print("len(): ", len(list2))                       #输出：len():  5
print("min(): ", min(list2))                       #输出：min():  89
print("max(): ", max(list2))                       #输出：max():  99
print("sum(): ", sum(list2))                       #输出：sum():  472
print("index(): ", list2.index(99))                #输出：index():  0
print("count(): ", list2.count(99))                #输出：count():  1
```


视频讲解

5.3　列表的循环遍历

除了使用索引、切片方式访问列表的一个元素或部分元素外，还可以通过 while 和 for 循环遍历列表，依次访问列表元素。

5.3.1　使用 while 循环

在使用 while 循环遍历列表时，通常在 while 循环前面定义列表和循环用的变量 i，然后使用 i 是否小于列表的长度做判断条件，来限制循环执行的次数。

例 5.7　使用 while 循环遍历列表，示例代码如下所示。

```
list =['我是列表', True, 0]
i =0
while i <len(list):
    print(list[i])                    #当满足条件时,执行 while 循环结构中的循环语句
    i +=1
```

运行结果如下所示。

```
我是列表
True
0
```

5.3.2 使用 for 循环

在使用 for 循环遍历列表时,依次将当前循环对应的列表元素赋值给 for 后面的变量,语法格式如下所示。

```
for 变量名 in 列表:
    循环语句
```

例 5.8 使用 for 循环遍历列表,示例代码如下所示。

```
list =['我是列表', True, 0]
for temp in list:
    print(temp)
```

使用 for 循环除了可以遍历整个列表,也可以遍历使用切片截取后的列表。

例 5.9 使用 for 循环遍历切片截取后的列表,示例代码如下所示。

```
list =['我是列表', True, 0, 3.14, [1, 2, 3], '列表可包含多种类型的数据']
for temp in list[1:5:2]:
  print(temp)
```

切片[1:5:2]的截取区间是从索引 1 到 5,步长为 2,注意选取的区间左闭右开,不包含结束索引 5 对应的元素,其运行结果如下所示。

```
True
3.14
```

5.4 列表的排序


视频讲解

列表中的元素可以进行升序、降序或者逆序排列。其中,逆序就是将元素前后位置反转,最前面的元素放到最后面,最后面的元素放到最前面。例如,列表[6 ,5 ,3, 2, 8 ,9],升序排列后变为[2, 3, 5, 6, 8, 9]、降序排列后变为[9, 8, 6, 5, 3, 2]、逆序排列后变为[9, 8, 2, 3, 5, 6]。

列表的升降序排列可以使用 sort()方法或者 sorted()函数，逆序可以使用 reverse()方法或者切片，具体说明如表 5.2 所示。

表 5.2　列表排序常用方式

函数/方法	语 法 格 式	说　　明
sort()方法	升序：列表.sort() 降序：列表.sort(reverse＝True)	对原列表进行排序，即原列表会被排序后的列表覆盖
sorted()函数	升序：sorted(列表) 降序：sorted(列表,reverse＝True)	有返回值，会返回一个新的有序列表，原列表没有改变
reverse()方法	列表.reverse()	对原列表进行逆序排列，即原列表会被逆序后的列表覆盖
使用切片	列表[::－1]	使用省略起始和结束的索引、步长为－1 的切片，原列表没有改变

例 5.10　列表的排序操作，示例代码如下所示。

```
num_list =[6, 5, 3, 2, 8, 9]
str_list =['16 辽宁舰', '18 福建舰', '17 山东舰']
#使用 sort 方法和 sorted 函数对列表排序
num_list.sort()                                    #升序排列
str_list.sort(reverse=True)                        #降序排列
new_num_list=sorted(num_list, reverse=True)        #降序排列
print(num_list)                                    #输出：[2, 3, 5, 6, 8, 9]
print(str_list)                          #输出：['18 福建舰', '17 山东舰', '16 辽宁舰']
print(new_num_list)                                #输出：[9, 8, 6, 5, 3, 2]
```

例 5.11　列表的逆序操作，示例代码如下所示。

```
num_list =[6, 5, 3, 2, 8, 9]
str_list =['16 辽宁舰', '17 山东舰', '18 福建舰']
str_list.reverse()              #使用 reverse 方法对列表逆序
new_num_list =num_list[::-1]    #使用切片对列表逆序
print(str_list)                 #输出：['18 福建舰', '17 山东舰', '16 辽宁舰']
print(new_num_list)             #输出：[9, 8, 2, 3, 5, 6]
```

【任务实现】


视频讲解

1. 分析代码

通过分析任务要求可知，任务实现的步骤是：创建一个列表 scores 用来保存选手的所有得分，找到列表中的最高分和最低分，计算列表中剩下的所有得分的平均值，即为选手的最终得分。其中，找列表中的最高分和最低分有以下两种办法。

(1) 使用内置函数 max()和 min()找出列表中最大值和最小值，即最高分和最低分。

(2) 对分数列表 scores 用 sort()方法进行升序排列，第一个元素即为最低分，最后一个

元素为最高分。

2. 编写代码

(1) 启动 PyCharm,选择菜单 File→New Project,指定项目位置为 D:\Chapter05。

(2) 右击项目文件夹 Chapter05,在弹出的快捷菜单中选择 New→Python File,在弹出的新建 Python 文件对话框中输入文件名 Score,类别为 Python file。

(3) 在 Score.py 文件的代码编辑窗口,输入如下代码。

```
print("-----演讲比赛评分系统-----")
scores =[99, 96, 93, 96, 96, 98, 95, 99, 93, 89, 95]
max =max(scores)
min =min(scores)
print('去掉一个最高分: ', max)
print('去掉一个最低分: ', min)
sum =sum(scores)          #使用 sum()函数计算列表中所有元素的和
score =(sum -max -min) / (len(scores) -2)
#round()函数 对浮点数进行四舍五入,第 2 个参数指定小数点后要保留的位数
print('选手的最终得分: ', round(score, 2))
```

(4) 再创建第 2 个程序文件,在对话框中输入文件名 Score2。在 Score2.py 文件的代码编辑窗口,输入如下代码。

```
print("-----演讲比赛评分系统-----")
scores =[99, 96, 93, 96, 96, 98, 95, 99, 93, 89, 95]
scores.sort()
print('去掉一个最高分: ', scores[-1])
print('去掉一个最低分: ', scores[0])
#使用 sum()函数计算切片截取后的列表中所有元素的和
sum =sum(scores[1:-1])
score =sum / (len(scores) -2)
#round()函数 对浮点数进行四舍五入,第 2 个参数指定小数点后要保留的位数
print('选手的最终得分: ', round(score, 2))
```

(5) 按快捷键 Ctrl+Shift+F10 分别运行 Score.py 和 Score2.py,可以在底部的结果窗格中查看运行结果,如图 5.1 所示。

【任务总结】

通过本任务的学习,读者可以全面掌握 Python 列表的创建、列表元素的访问、常见的列表操作、列表的循环遍历和列表的排序等内容。在使用时需要注意以下几点。

- 创建列表除了可以使用中括号、内置函数 list()、列表推导式以外,还可以通过复制现有列表、使用 extend()或运算符+连接列表、使用 itertools 模块等方式来创建列表。
- Python 中列表的索引是从 0 开始,而不是从 1 开始。索引的取值范围是 0 至列表长

度－1，如果使用的索引值超出索引范围，程序会报索引越界错误。

- 列表切片不包括结束索引对应的元素。如果起始索引大于或等于结束索引，并且没有指定负步长，那么切片将为空。不带任何参数的切片将返回整个列表。
- 默认情况下，列表排序是基于列表中元素的升序排列。Python 中还支持通过 key 参数来自定义排序依据，例如，按照字符串长度、元素的某个属性或其他复杂逻辑进行排序。

【巩固练习】

一、选择题

1. 下列选项中，用于返回列表的元素个数的函数是(　　)。

A. len　　B. count　　C. list　　D. length

2. 下列选项中，用于返回列表的元素最大值的函数是(　　)。

A. len　　B. min　　C. index　　D. max

3. 阅读下面的程序，程序运行的正确结果是(　　)。

```
list =[0, 1, 2, 3, 4, 5]
for i in list:
    if i %3 ==0:
        continue
    print(i, end="")
```

A. 012345　　B. 01245　　C. 1245　　D. 12

4. 阅读下面的程序，程序运行的正确结果是(　　)。

```
list =[7, 0, 3, 4, 5]
list.sort()
print(list[:-3])
```

A. [7，0，3，4，5]　　B. [0，3，4，5，7]

C. [0，3]　　D. [0，3，4]

5. 下面代码的输出结果是(　　)。

```
list =[7, 0, 3, 4, 5]
list.reverse()
print(list)
```

A. [7，0，3，4，5]　　B. [0，3，4，5，7]

C. [5，4，3，0，7]　　D. 以上都不正确

二、判断题

1. 在 Python 中，使用 index 方法查找元素在序列中第一次出现的位置索引时，找到则返回索引值，若没找到返回－1。(　　)