教案（分教案）

课 次： 3 学 时： 2

章 节	第 3 章 列表和元组 3.1 序列 3.2 列表 3.3 元组
教学目的和教学要求	Python 数据结构是数据组织的常用方式，本章的教学目的是要求学生根据实际数据的格式需求能够选择适合的数据结构，并能够熟练进行各种操作。 要求学生了解序列、列表和元组的概念和区别，掌握列表和元组的增加、删除、修改和查找、排序等方法，能够利用列表实现简易的各种数据管理系统。
教学重点难点	**教学重点：** 序列，列表，元组，列表结构的增、删、改、查、排等功能实现 **教学难点：** 列表及操作，简易管理系统的算法设计
教学方法和辅助手段	**教学方法：**课堂讲解、案例分析，实际操作，提问讨论，反思质疑 **辅助手段：**电脑、投影仪、教科书

第 3 章 列表和元组

列表（list）、元组（tuple）和字典（dict），这三种数据结构都可用于保存多个数据项，这对于编程而言是非常重要的。因为程序不仅需要使用单个变量来保存数据，还需要使用多种数据结构来保存大量数据，而列表、元组和字典就可满足保存大量数据的需求。

列表和元组比较相似，它们都按顺序保存元素， 每个元素都有自己的索引，因此列表和元组都可通过索引访问元素。二者的区别在于元组是不可修改的，但列表是可修改的。

3.1 序列

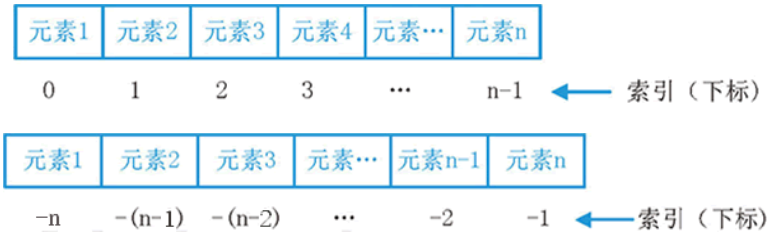
- 数据结构是通过某种方式组织在一起（按顺序排列）的元素的集合，Python 中有四种内建的数据结构，即列表、元组、字典、集合。
- 容器（Container）是一种 Python 的数据结构，基本上是包含其他对象的任意对象。序列和映射（如字典）是两类主要的容器，集合（Set）也是容器类型。
- 序列（Sequence）是最基本的数据结构，是通过对数据元素进行编号将它们组织在一起的数据元素的集合。
- 序列是一块用于存放多个值的连续内存空间，并且按一定顺序排列，每一个值（称为元素）都分配一个数字，称为索引或位置。通过该索引可以取出相应的值。例如，可以把一家酒店看作一个序列，那么酒店里的每个房间都可以看作是这个序列的元素。而房间号就相当于索引，可以通过房间号（索引）找到该酒店（序列）对应的房间（内存地址）。
- Python 有 6 中种内建的序列：列表、元组、字符串、Unicode 字符串（字符串前面加 u,定义标准 unicode 字符串）、buffer 对象、xrange 对象。
  - 最通俗的讲 Unicode 字符串和就是将普通字符串给标准化了，它为每个字符设定了统一并且唯一的二进制编码，以满足跨语言、跨平台进行文本转换、处理的要求。
  - xrange 方法返回的是 xrange 对象，它是一个序列对象，但并不保存序列中的元素。而 range 方

法返回的是一个 list 对象，它需要开辟专门的空间保存序列中所有的元素。因此，如果只对序列进行读操作，xrange 方法效率较高；但是如果需要改变序列的元素，或者需要往序列增删元素，那只能通过 range 方法生成一个 list 对象。

对于序列结构，通用的操作有索引、切片、序列相加、序列相乘以及和序列相关的内置函数：

1、索引

索引是从 0 开始递增，即下标为 0 表示第一个元素，下标为 1 表示第二个元素，以此类推。



Python 还支持索引值是负数，此类索引是从右向左计数，换句话说，从最后一个元素开始计数，从索引值 -1 开始

注意：在使用负值作为列序中各元素的索引值时，是从 -1 开始，而不是从 0 开始。

2、切片

切片操作是访问序列中元素的另一种方法，它可以访问一定范围内的元素，通过切片操作，可以生成一个新的序列。

序列实现切片操作的语法格式如下：**sname[start : end : step]**，其中，各个参数的含义分别是：

- ✧ sname：表示序列的名称；
- ✧ start：表示切片的开始索引位置（包括该位置），此参数也可以不指定，会默认为 0，也就是从序列的开头进行切片；
- ✧ end：表示切片的结束索引位置（不包括该位置），如果不指定，则默认为序列的长度；
- ✧ step：表示在切片过程中，隔几个存储位置（包含当前位置）取一次元素，也就是说，如果 step 的值大于 1，则在进行切片去序列元素时，会“跳跃式”的取元素。如果省略设置 step 的值，则最后一个冒号就可以省略。

3、序列相加

Python 中，支持两种类型相同的序列使用“+”运算符做相加操作，它会将两个序列进行连接，但不会去除重复的元素。

这里所说的“类型相同”，指的是“+”运算符的两侧序列要么都是列表类型，要么都是元组类型，要么都是字符串。

4、序列相乘

Python 中，使用数字 n 乘以一个序列会生成新的序列，其内容为原来序列被重复 n 次的结果。

比较特殊的是，列表类型在进行乘法运算时，还可以实现初始化指定长度列表的功能。例如如下的代码，将创建一个长度为 5 的列表，列表中的每个元素都是 None，表示什么都没有。

5、检查元素是否包含在序列中

Python 中，可以使用 in 关键字检查某元素是否为序列的成员，其语法格式为：**value in sequence**，其中，value 表示要检查的元素，sequence 表示指定的序列。和 in 关键字用法相同，但功能恰好相反的，还有 not in 关键字，它用来检查某个元素是否不包含在指定的序列中。

6、和序列相关的内置函数

Python 提供了几个内置函数（表 3-1 所示），可用于实现与序列相关的一些常用操作。

表 3-1 序列相关的内置函数

函数	功能
len()	计算序列的长度，即返回序列中包含多少个元素。len(iterable)
max()	找出序列中的最大元素。max(iterable)
min()	找出序列中的最小元素。min(iterable)
list()	将序列转换为列表。list(iterable)
str()	将序列转换为字符串。
sum()	计算元素和。sum(iterable) 注意，对序列使用 sum() 函数时，做“加和操作”的必须都是数字，不能是字符或字符串，否则该函数将抛出异常，因为解释器无法判定是要做连接操作（+ 运算符可以连接两个序列），还是做加和操作。
sorted()	sorted(iterable,key,reverse)对元素进行排序，产生一个新的列表。此外，用对 List 的成员函数 sort 进行排序。 ✧ sort()与 sorted()的不同在于，sort 是在原位重新排列列表，而 sorted()是产生一个新的列表。 ✧ sort 是应用在 list 上的方法，sorted 可以对所有可迭代的对象进行排序操作。 ✧ list 的 sort 方法返回的是对已经存在的列表进行操作，而内建函数 sorted 方法返回的是一个新的 list，而不是在原来的基础上进行的操作。
reversed()	反向序列中的元素。iterable. reversed()
enumerate()	将序列组合为一个索引序列，多用在 for 循环中。

案例分析：

```
# -*- coding: UTF-8 -*-
"""
@author:Lenovo
@file_name:sequence_phone_demo.py
@time:2021-09-13 10:42
@IDE:PyCharm
@copyright: http://maths.xynu.edu.cn
"""
from pypinyin import lazy_pinyin


# lazy_pinyin：不考虑音调，返回一维列表，而非二维数组


def sequence_demo(phone):
    """
    根据手机品牌数据序列进行各种操作，主要是增删改查排
```

```
    :param phone: 手机品牌数据序列
    :return: None
    """
    # 1、索引
    print("1. 第一个手机品牌是：", phone[0])  # 下标从零开始
    print("2. 最后一个手机品牌是：", phone[-1])

    # 2、切片：sname[start: end: step]
    print("3. 手机品牌前十名为：\n", phone[0:10])
    print("4. 手机品牌第 2、4、6、8、10 位是：\n", phone[0:10:2])  # step = 2
    last5 = phone[-10::2]  # 省略中间 end 参数，表示到最后一个值
    last5.reverse()
    print("5. 手机品牌倒数第 2、4、6、8、10 名是：\n", last5)

    # 3、序列相加
    print("6. 手机品牌前五名与后五名：")
    first5 = phone[:5]  # 表示从第一个元素开始的五个值
    end5 = phone[-5:]  # 表示从最后一个值到倒数第五个值
    print(first5 + end5)

    # 4、序列乘法
    print("7. 我爱【" + phone[0] * 3 + "】，重要的事情说三遍！")

    # 5、检查某个元素是否是序列的成员（元素）
    print("8. 诺基亚曾是我的最爱，是否在手机品牌中：", end="")
    if "诺基亚" in phone:
        print("是")
    else:
        print("否")

    # 6、计算序列的长度
    print("9. 全球手机品牌共有：%d 个。" % len(phone))

    # 7、反转，元素内部顺序颠倒，只在内部翻转，并不返回什么
    print("10. 反转后前五名为：", end="")
    phone.reverse()  # phone[::-1]
    print(phone[0:6])
    phone.reverse()  # 再次反转，恢复原来顺序

    # 8、增删改
    phonef5 = phone[0:6]
    phonef5.append("华为 5G")
    print("11. 序列后追加：", phonef5)
    phonef5.insert(2, "OPPO5G")  # 第 3 个位置插入元素
    print("12. 在第 3 个位置插入元素：", phonef5)
    del phonef5[-1]
    phonef5.remove("OPPO5G")
```

```
    print("13. 删除后：", phonef5)

    # 9、排序，按照拼音
    phone_seq.sort(key=lambda phone: lazy_pinyin(phone))  # 对原序列操作，不返回新对象
    # phone_seq2 = sorted(phone_seq, key=lambda phone: lazy_pinyin(phone))  # 返回新的对象
    print("14. 排序后，输出前五名和后五名：")
    print(phone_seq[0:6] + phone_seq[-6:-1])

    # 10、清空列表
    phonef5.clear()
    if ~len(phonef5):
        print("15. 清空元素后，序列为空！")
    else:
        print("15. 清空元素后，序列不空！")


def read_phone_data():
    """
    读取文件数据，并适当对数据进行处理，并打印输出
    :return:
    """
    phone_seq = []  # 定义一个序列：列表
    with open("../data/phone.txt", 'r', encoding="utf8") as fr:
        phones = fr.readlines()  # 按行读取
    for phone in phones:
        phone = phone[:-1]  # 删除回车符\n
        phone_seq.append(phone)
    print("=" * 25 + " >> 全球手机列表 << " + "=" * 25, end="")
    for k, ph in enumerate(phone_seq):  # enumerate 函数用法
        if k % 10 == 0:  # 每十个元素一行
            print()  # 输出内容为空，但打印换行符
        if k == len(phone_seq) - 1:  # 最后一个元素，不打印“, ”
            print(ph)
        else:
            print(ph, end=", ")  # 非最后一个元素，用“, ”号分隔
    print("=" * 70)
    return phone_seq


if __name__ == '__main__':
    phone_seq = read_phone_data()
    sequence_demo(phone_seq)
```

程序运行结果：

```
========================= >> 全球手机列表 << ========================
华为, OPPO, vivo, 苹果, 三星, 荣耀, 小米, 一加, 魅族, realme,
努比亚, 红米, 诺基亚, 中兴, 索尼移动, 联想, 黑鲨, Moto, 锤子科技, ROG,
```

```
金立, 谷歌, LG, 360, 海信, 中国移动, 美图, 酷派, 黑莓, 飞利浦,
AGMVERTU, 小辣椒, HTC8848, 华硕, 夏普, TCL, 国美手机, 朵唯, 天语, SUGAR,
长虹, 欧奇, 纽曼, 詹姆士, 雷蛇, 征服, 柔宇, MANN, 康佳, 海尔,
创星, ivvi, 索野, 格力, 传音, YotaPhone, 多亲, 微软, Librem, 百合,
波导, 守护宝, 青橙, 21 克, 卡布奇诺, 誉品, 酷比, 大神, 乐目, 尼凯恩,
私人医生, 保千里, sonim, 红鸟, E 人 E 本, 铂爵, 独影, 天幕, 联想, ZUKioutdoor,
神舟, 朗界, 邦华, 小格雷, 阿尔卡特, 克里特, 易百年
==========================================================================
1. 第一个手机品牌是： 华为
2. 最后一个手机品牌是： 易百年
3. 手机品牌前十名为：
['华为', 'OPPO', 'vivo', '苹果', '三星', '荣耀', '小米', '一加', '魅族', 'realme']
4. 手机品牌第 2、4、6、8、10 位是：
['华为', 'vivo', '三星', '小米', '魅族']
5. 手机品牌倒数第 2、4、6、8、10 名是：
['克里特', '小格雷', '朗界', 'ZUKioutdoor', '天幕']
6. 手机品牌前五名与后五名：
['华为', 'OPPO', 'vivo', '苹果', '三星', '邦华', '小格雷', '阿尔卡特', '克里特', '易百年']
7. 我爱【华为华为华为】，重要的事情说三遍！
8. 诺基亚曾是我的最爱，是否在手机品牌中：是
9. 全球手机品牌共有：87 个。
10. 反转后前五名为：['易百年', '克里特', '阿尔卡特', '小格雷', '邦华', '朗界']
11. 序列后追加： ['华为', 'OPPO', 'vivo', '苹果', '三星', '荣耀', '华为 5G']
12. 在第 3 个位置插入元素： ['华为', 'OPPO', 'OPPO5G', 'vivo', '苹果', '三星', '荣耀', '华为 5G']
13. 删除后： ['华为', 'OPPO', 'vivo', '苹果', '三星', '荣耀']
14. 排序后，输出前五名和后五名：
['21 克', '360', 'AGMVERTU', 'E 人 E 本', 'HTC8848', 'LG', '一加', '誉品', '詹姆士', '征服', '中国移动']
15. 清空元素后，序列为空！
```

序列封包和序列解包

Python 还提供了序列封包（ Sequence Packing ）和序列解包（ Sequence Unpacking ）的功能。简单来说， Python 允许支持以下两种赋值方式。

- 程序把多个值赋给一个变量时， Python 会自动将多个值封装成元组。这种功能被称为序列封包。
- 程序允许将序列（元组或列表等）直接赋值给多个变量，此时序列的各元素会被依次赋值给每个变量（要求序列的元素个数和变量个数相等）。这种功能被称为序列解包。

案例说明：

```
import random as rd


def pack_unpack():
    """
    序列封包与解包各种用法详解
    :return: None
    """
    # 序列封包：将十个整数封装成元组后赋值给 score
    score = 90, 75, 57, 70, 57, 88, 78, 68, 59, 81
```

```
print("1. 序列封包：",score)
print("1.1. 序列封包后类型：",type(score)) # <class 'tuple'>
print("1.2. 以序列方式读取第 2 到 5 的元素：",score[1:5])
print("="*70)

score_tuple = tuple(rd.randint(50,90) for i in range(5))
print("2. 随机成绩整数为：",score_tuple)
# 序列解包：将 score_tuple 元组的各元素依次赋值给 a、b、c、d、e 变量
a, b, c, d, e = score_tuple
print("2.1. 序列解包 a,b,c,d,e = score_tuple：",a, b, c, d, e)
print("=" * 70)

a_list = ['寒江孤影', '江湖故人','相逢何必曾相识']
# 序列解包 fk：将 a_list 序列的各元素依次赋值给 a_str、b_str 变量
astr, bstr, cstr = a_list
print("3. 序列解包 astr,bstr,cstr = a_list：",astr, bstr,cstr)
print("=" * 70)

# 将 66、88、99 依次赋值给 x、y、z
x, y, z = 66, 88, 99
print("4. 将 66、88、99 依次赋值给 x、y、z：",x, y, z) # 66 88 99
# 将 y,z, x 依次赋值给 x、y、z
x, y, z = y, z, x
print("4.1 将 y,z,x 依次赋值给 x、y、z：",x, y, z) # 88 66 99
print("=" * 70)

# first、second 保存前 2 个元素，rest 列表包含剩下的元素
character = ((rd.randint(65, 90) + rd.randint(0, 1) * 32) for i in range(10))
first, second, *rest = character
print("5. first, second, *rest，输出第一个元素：",chr(first))
print("5.1. first, second, *rest，第二个元素：",chr(second))
print("5.2. 剩余元素：",end="")
for r in rest:
    print(chr(r), end=" ")
print("\n" + "=" * 70)

# last 保存最后一个元素，begin 保存前面剩下的元素
*begin, last = score
print("6. *begin, last，输出 begin：",begin)
print("6.1. *begin, last，输出 last：",last)
print("=" * 70)

# first 保存第一个元素，last 保存最后一个元素，middle 保存中间剩下的元素
first, *middle, last = score
print("7. first, *middle, last，输出第一个元素：",first)
print("7.1. 保存中间剩下的元素：",middle)
print("7.2. 输出最后一个元素：",last)
```

```
if __name__ == '__main__':
    pack_unpack()
```

程序运行结果：

```
1. 序列封包： (90, 75, 57, 70, 57, 88, 78, 68, 59, 81)
1.1. 序列封包后类型： <class 'tuple'>
1.2. 以序列方式读取第 2 到 5 的元素： (75, 57, 70, 57)
======================================================================
2. 随机成绩整数为： (68, 59, 69, 82, 58)
2.1. 序列解包 a,b,c,d,e = score_tuple： 68 59 69 82 58
======================================================================
3. 序列解包 astr,bstr,cstr = a_list： 寒江孤影 江湖故人 相逢何必曾相识
======================================================================
4. 将 66、88、99 依次赋值给 x、y、z： 66 88 99
4.1 将 y,z,x 依次赋值给 x、y、z： 88 99 66
======================================================================
5. first, second, *rest，输出第一个元素： x
5.1. first, second, *rest，第二个元素： D
5.2. 剩余元素：Y D y k M Q X c
======================================================================
6. *begin, last，输出 begin： [90, 75, 57, 70, 57, 88, 78, 68, 59]
6.1. *begin, last，输出 last： 81
======================================================================
7. first, *middle, last，输出第一个元素： 90
7.1. 保存中间剩下的元素： [75, 57, 70, 57, 88, 78, 68, 59]
7.2. 输出最后一个元素： 81
```

3.2 列表

列表是 Python 中内置**有序、可变序列**，列表的所有元素放在一对中括号“[]”中，并使用逗号分隔开；当列表元素增加或删除时，列表对象自动进行扩展或收缩内存，保证元素之间没有缝隙；在 Python 中，一个列表中的数据类型可以各不相同，可以同时分别为整数、实数、字符串等基本类型，甚至是列表、元组、字典、集合以及其他自定义类型的对象：

- [10, 20, 30, 40]
- ['crunchy frog', 'ram bladder', 'lark vomit']
- ['spam', 2.0, 5, [10, 20]]
- [['file1', 200,7], ['file2', 260,9]]

列表与元组之间的转换：元组转换为列表函数 list()，列表转换为元组函数 tuple()。

```
# -*- coding:UTF-8 -*-
# 开发时间：2020/9/22 16:00
```

```
def tupleToList():
    """
    元组转换为列表函数 list()，列表可变
    :return: None
    """
    a_tuple = ('Python', "龟叔", 1989)
    mylist = list(a_tuple) # 将元组转换成列表
    mylist.append("优雅简洁高效")
    print(mylist)

    for ind,ml in enumerate(mylist):  #使用 for 循环和 enumerate()函数实现
        print(ind + 1, ml)
    print("-"*50)

def listToTuple():
    """
    列表转换为元组函数 tuple()，元组不可变
    :return: None
    """
    a_list = ['平凡的世界', "路遥", 115.8]
    mytuple = tuple(a_list)# 将列表转换成元组
    print(mytuple)
    for ind,tp in enumerate(mytuple):  #使用 for 循环和 enumerate()函数实现
        print(ind + 1, tp)

if __name__ == '__main__':
    tupleToList()
    listToTuple()
```

程序运行结果：

```
['Python', '龟叔', 1989, '优雅简洁高效']
1 Python
2 龟叔
3 1989
4 优雅简洁高效
--------------------------------------------------
('平凡的世界', '路遥', 115.8)
1 平凡的世界
2 路遥
3 115.8
```

表 3-2 列表常用方法

方法	说明
lst.append(x)	将元素 x 添加至列表 lst 尾部

lst.extend(L)	将列表 L 中所有元素添加至列表 lst 尾部
lst.insert(index, x)	在列表 lst 指定位置 index 处添加元素 x，该位置后面的所有元素后移一个位置
lst.remove(x)	在列表 lst 中删除首次出现的指定元素，该元素之后的所有元素前移一个位置
lst.pop([index])	删除并返回列表 lst 中下标为 index（默认为-1）的元素
lst.clear()	删除列表 lst 中所有元素，但保留列表对象
lst.index(x)	返回列表 lst 中第一个值为 x 的元素的下标，若不存在值为 x 的元素则抛出异常
lst.count(x)	返回指定元素 x 在列表 lst 中的出现次数
lst.reverse()	对列表 lst 所有元素进行逆序
lst.sort(key=None, reverse=False)	对列表 lst 中的元素进行排序，key 用来指定排序依据，reverse 决定升序（False），还是降序（True）
lst.copy()	返回列表 lst 的浅复制，只能复制一层。深复制可用 copy.deepcopy(list)

案例分析：学生成绩简易管理系统，使用列表存储数据，并实现增加、删除、修改、查询、排序、遍历访问、过滤筛选、数据基本统计量计算等功能。

```
# -*- coding: UTF-8 -*-
"""
@author:Lenovo
@file_name:student_score_simple_manage_system.py
@time:2021-09-17 9:21
@IDE:PyCharm
@copyright: http://maths.xynu.edu.cn
"""
# 案例分析： 学生成绩 简易 管理系统，使用列表 存储数据 ，并实现增加、删除、修改、查询、排序、遍历访
# 问、过滤 筛选 、数据基本统计量计算 等功能。

import numpy as np

# soores 为全局数据列表
scores = [['刘备'], ['关羽'], ['张飞'], ['诸葛亮'], ['赵云'], ['马超'], ['黄忠'], ['司马懿'],
          ['周瑜'], ['黄盖'], ['许褚'], ['典韦'], ['姜维'], ['夏侯敦'], ['张辽'], ['吕布'],
          ['吕蒙'], ['曹操'], ['孙权'], ['夏侯渊']]  # 列表中的每个元素也是列表



# 选择什么样的数据结构存储数据，决定了此种数据结构的操作，以及算法的效率。

def create_init_list(scores):
    """
    初始化创建列表，为每名同学随机生成三个科目的成绩，成绩为整数
```

```
    :param scores: 学生全局成绩数据列表，该参数可不须要，因为是全局变量，这里统一对每个方法添加
    :return:
    """
    np.random.seed(1123)
    for student in scores:  # 遍历列表中的每个元素
        # 列表中的每个元素也是列表，为每个学生追加三个成绩，并转换为列表
        # 其每隔元素形式为：['刘备', [88, 97, 77]]，每个元素本质上还是列表，
        # 其中第一个元素是字符串，第二个元素是列表（三个成绩）
        student.append(list(np.random.randint(60, 99, 3)))  # ['刘备', [88, 97, 77]]


def show_list(scores):
    """
    显示列表中所有人的成绩，注意：索引的使用，遍历元素的方法
    :param scores: 学生全局成绩数据列表
    :return:
    """
    print("%2s %6s %8s %8s %8s" % ("学号", "姓名", "科学计算", "数据分析", "机器学习"))
    for idx, score in enumerate(scores):
        # score 表示列表中的每个元素，形式为：['刘备', [88, 97, 77]]
        print("%2d %8s" % (idx + 1, score[0]), end="")
        if len(list(score[0])) == 3:
            # score[1]表示取第二个元素，其为列表[88, 97, 77]
            print("%9d %12d %12d" % (score[1][0], score[1][1], score[1][2]))
        else:
            print("%10d %12d %12d" % (score[1][0], score[1][1], score[1][2]))


def name_list(scores):
    """
    姓名列表，及架设了姓名为唯一 id 信息，各种操作，姓名不能重复，故取姓名列表
    :param scores:
    :return:
    """
    namelist = []  # 存储所有学生的姓名
    for score in scores:
        namelist.append(score[0])
    return namelist


def insert_list(scores):
    """
    增加学生及其成绩，并追加到最后或者根据用户需求选择插入的位置
    假设姓名是唯一 id 信息，不允许重复
    :param scores:
    :return:
    """
```

```
    namelist = name_list(scores)  # 获取姓名列表
    while True:
        name = input("请输入姓名（#结束）")
        if name == "#":
            break
        # 判断输入的姓名是否已经存在
        if name in namelist:
            print("该同学【%s】已存在，不允许添加，可修改！" % name)
        else:
            sc = int(input("请输入《科学计算》成绩："))
            da = int(input("请输入《数据分析》成绩："))
            ai = int(input("请输入《机器学习》成绩："))
            # 追加成绩，把学生信息构成列表形式，即[姓名, [sc, da, ai]]
            scores.append([name, [sc, da, ai]])
            namelist.append(name)
            print("添加成功，信息：", scores[-1])


def delete_list(scores):
    """
    删除指定学生的信息：先查询，若存在则删除，不存在则提是
    :return:
    """
    namelist = name_list(scores)
    while True:
        name = input("请输入要删除的学生姓名（#结束）")
        if name == "#":
            break
        # 判断输入的姓名是否已经存在
        if name in namelist:
            # 返回列表 lst 中第一个值为 x 的元素的下标，若不存在值为 x 的元素则抛出异常
            idx = namelist.index(name)  # 要删除的学生的索引下标
            del scores[idx]
            print("删除成功，剩余学生数量为【%d】名。" % len(scores))
        else:
            print("你删除的学生【%s】不存在！" % name)


def search_list(scores):
    """
    根据项目进行查询，并输出查询到的学生成绩信息
    :param scores:
    :return:
    """
    namelist = name_list(scores)
    name = input("请输入要查询的学生姓名")
    # 判断输入的姓名是否已经存在
```

```
    if name in namelist:
        idx = namelist.index(name)  # 要查找的学生的索引下标
        print(name + " 的成绩为：", scores[idx][1])
    else:
        print("你查找的学生【%s】不存在！" % name)


def modify_list(scores):
    """
    修改学生成绩，首先根据姓名进行查找，存在则修改。
    注意：对输入的成绩要做健壮性判断，此处未作。
    :param scores:
    :return:
    """
    namelist = name_list(scores)
    name = input("请输入要修改的学生姓名：")
    if name in namelist:
        idx = namelist.index(name)
        select = int(input("请选择修改科目序号（1 科学计算，2 数据分析，3 机器学习，4 All）："))
        if select == 1:
            sc = int(input("请输入要修改的《科学计算》成绩："))
            scores[idx][1][0] = sc  # 索引 idx 代表整个学生对象，1 索引代表该学生成绩列表，0 索引代表科学计算成
绩
            print("成绩更新成功，结果为：", scores[idx])
        elif select == 2:
            da = int(input("请输入要修改的《数据分析》成绩："))
            scores[idx][1][1] = da
            print("成绩更新成功，结果为：", scores[idx])
        elif select == 3:
            ai = int(input("请输入要修改的《机器学习》成绩："))
            scores[idx][1][2] = ai
            print("成绩更新成功，结果为：", scores[idx])
        elif select == 4:
            sc = int(input("请输入要修改的《科学计算》成绩："))
            scores[idx][1][0] = sc
            da = int(input("请输入要修改的《数据分析》成绩："))
            scores[idx][1][1] = da
            ai = int(input("请输入要修改的《机器学习》成绩："))
            scores[idx][1][2] = ai
            print("成绩更新成功，结果为：", scores[idx])
    else:
        print("您要修改的的学生【%s】不存在！" % name)


def sort_by_sumscore_list(scores):
    """
    按照总成绩进行排序，要计算每个学生的总成绩
```

```
    :param scores:
    :return:
    """
    sum_score = []  # 存储每个学生的总成绩
    for score in scores:
        sum_score.append([score[0], sum(score[1])])
    select = int(input("升序选择 1，降序选择 2："))
    # 对列表 lst 中的元素进行排序，key 用来指定排序依据，reverse 决定升序
    # （False），还是降序（True）
    if select == 1:
        sum_score.sort(key=lambda score: score[1], reverse=False)  # ['张飞', 233]
    elif select == 2:
        sum_score.sort(key=lambda score: score[1], reverse=True)  # ['张飞', 233]
    else:
        print("选择有误！")
        return
    print("排序后的结果为：")
    for score in sum_score:
        print("name =%4s，sum score =%4d" % (score[0], score[1]))


def partial_subject_score_list(scores):
    """
    筛选偏科学生名单。偏科：一科目小于 70 分，另外两科目均大于等于 85 分
    :param scores:
    :return:
    """
    ps_student = []  # 存储偏科学生名单
    for score in scores:
        score_sort = sorted(score[1])  # 先升序排序[94, 87, 60]-->[60, 87, 94]
        if score_sort[0] < 70 and score_sort[1] >= 85 and score_sort[2] >= 85:
            ps_student.append([score[0], score[1]])
    print("偏科学生名单如下：")
    show_list(ps_student)


def subject_score_stats_list(scores):
    """
    统计各学科成绩的数字特征
    :param scores:
    :return:
    """
    sc, da, ai = [], [], []
    for score in scores:
        sc.append(score[1][0])  # 科学计算
        da.append(score[1][1])  # 数据分析
        ai.append(score[1][2])  # 机器学习
```

```
    subject_stats = []  # 存储各科目的数字特征
    subject_stats.append([max(sc), min(sc), sum(sc) / len(sc), np.std(sc), np.median(sc), score_mode(sc)])
    subject_stats.append([max(da), min(da), sum(da) / len(da), np.std(da), np.median(da), score_mode(da)])
    subject_stats.append([max(ai), min(ai), sum(ai) / len(ai), np.std(ai), np.median(ai), score_mode(ai)])
    print("各学科统计量结果如下：")
    subject = ["科学计算", "数据分析", "机器学习"]
    print("%14s %6s %6s %6s %8s %5s" % ("min", "max", "mean", "std", "median", "mode"))
    for k, s in enumerate(subject_stats):
        print(subject[k], end=":")
        for val in s:
            print("%6.1f" % val, end=" ")
        print()


def score_mode(sub_socre):
    """
    计算各科目成绩的众数：参考第 2 章案例数据统计量计算
    :param sub_socre: 各科目成绩列表
    :return:
    """
    mode_dict = dict()  # 字典存储
    for s in sub_socre:
        if s not in mode_dict:
            mode_dict[s] = 1  # 新的数，频数为 1
        else:
            mode_dict[s] += 1  # 已存在，则加一
    mode_val = list(mode_dict.values())  # 获取频数，并转换为列表
    idx = int(np.argmax(mode_val))  # 在频数列表中查询最大频数的索引
    score_key = list(mode_dict.keys())  # 获取成绩，并转换为列表
    return score_key[idx]


def menu():
    """
    菜单功能：根据用户选择，实现不同的功能
    :return:
    """
    print("=" * 10 + ">>学生成绩管理菜单<<" + "=" * 10)
    print("\t1.显示学生列表成绩信息\n"
          "\t2.添加学生及其成绩\n"
          "\t3.删除学生及其成绩\n"
          "\t4.查询学生及其成绩\n"
          "\t5.修改学生及其成绩\n"
          "\t6.按照学生总成绩排序\n"
          "\t7.筛选成绩偏科学生名单\n"
          "\t8.各学科成绩统计分析\n"
          "\t9.退出系统")
```

```
    print("=" * 40)

    select = int(input("请输入您的选择："))
    if 1 <= select <= 9:
        return select
    else:
        print("您输入的选择序号有误！")
        return 0


if __name__ == '__main__':
    create_init_list(scores)
    while True:
        select = menu()
        if select == 1:
            show_list(scores)
        elif select == 2:
            insert_list(scores)
        elif select == 3:
            delete_list(scores)
        elif select == 4:
            search_list(scores)
        elif select == 5:
            modify_list(scores)
        elif select == 6:
            sort_by_sumscore_list(scores)
        elif select == 7:
            partial_subject_score_list(scores)
        elif select == 8:
            subject_score_stats_list(scores)
        elif select == 9:
            break
        else:
            print("系统退出.....")
            exit(0)
```

程序运行结果部分如下：

```
==========>>学生成绩管理菜单<<==========
    1.显示学生列表成绩信息
    2.添加学生及其成绩
    3.删除学生及其成绩
    4.查询学生及其成绩
    5.修改学生及其成绩
    6.按照学生总成绩排序
    7.筛选成绩偏科学生名单
```

```
    8.各学科成绩统计分析
    9.退出系统
==========================================
请输入您的选择：6
升序选择 1，降序选择 2：2
排序后的结果为：
name =  周瑜，sum score = 261
name = 夏侯敦，sum score = 260
name = 诸葛亮，sum score = 253
name =  姜维，sum score = 253
name =  马超，sum score = 248
name =  典韦，sum score = 246
name =  张辽，sum score = 244
name = 司马懿，sum score = 240
name =  张飞，sum score = 237
name =  赵云，sum score = 237
name =  刘备，sum score = 235
name =  许褚，sum score = 235
name =  曹操，sum score = 235
name =  关羽，sum score = 234
name =  孙权，sum score = 233
name = 夏侯渊，sum score = 229
name =  吕蒙，sum score = 220
name =  黄忠，sum score = 216
name =  吕布，sum score = 214
name =  黄盖，sum score = 206
==========>>学生成绩管理菜单<<==========
    1.显示学生列表成绩信息
    2.添加学生及其成绩
    3.删除学生及其成绩
    4.查询学生及其成绩
    5.修改学生及其成绩
    6.按照学生总成绩排序
    7.筛选成绩偏科学生名单
    8.各学科成绩统计分析
    9.退出系统
==========================================
请输入您的选择：1
学号      姓名      科学计算      数据分析      机器学习
 1        刘备        68          92          75
 2        关羽        88          60          86
```

3	张飞	84	74	79
4	诸葛亮	97	75	81
5	赵云	94	71	72
6	马超	71	81	96
7	黄忠	92	61	63
8	司马懿	89	70	81
9	周瑜	67	97	97
10	黄盖	77	62	67
11	许褚	77	73	85
12	典韦	77	76	93
13	姜维	83	74	96
14	夏侯敦	97	90	73
15	张辽	63	96	85
16	吕布	70	74	70
17	吕蒙	69	63	88
18	曹操	90	78	67
19	孙权	84	75	74
20	夏侯渊	63	69	97

==========>>学生成绩管理菜单<<==========

1.显示学生列表成绩信息

2.添加学生及其成绩

3.删除学生及其成绩

4.查询学生及其成绩

5.修改学生及其成绩

6.按照学生总成绩排序

7.筛选成绩偏科学生名单

8.各学科成绩统计分析

9.退出系统

==========

请输入您的选择：8

各学科统计量结果如下：

	min	max	mean	std	median	mode
科学计算:	97.0	63.0	80.0	11.0	80.0	77.0
数据分析:	97.0	60.0	75.5	10.7	74.0	74.0
机器学习:	97.0	63.0	81.2	10.7	81.0	81.0

列表的主要操作功能如下：

（1）创建和删除列表

（2）访问列表元素

（3）遍历列表，与序列遍历方法相同

（4）添加、修改和删除列表元素

（5）对列表进行统计计算

1）获取指定元素出现的次数，语法：listname.count(obj)

2）获取指定元素首次出现的下标，语法：listname.index(obj)

3）统计数值列表的元素和，语法：sum(iterable[, start])

（6）对列表进行排序：listname.sort(key = None, resverse = False)，默认升序，True 为降序

1）使用列表对象的 sort()方法实现

2）使用内置的 sorted()函数实现：sorted(iterable, key = None, resverse = False)

（7）列表推导式

1）生成指定范围的数值列表

语法：list = [Expression for var in range]

2）根据列表生成指定需求的列表

语法：newlist = [Expression for var in list]

3）从列表中选择符合条件的元素组成新的列表

语法：newlist = [Expression for var in list if condition]

代码示例：

```
# -*- coding: UTF-8 -*-
"""
@author:Lenovo
@file_name:list_reduction.py
@time:2021-09-17 15:46
@IDE:PyCharm
@copyright: http://maths.xynu.edu.cn
"""
import random  # 导入 random 标准库


def list_reduction():
    # 生成指定范围的数值列表，语法：list = [Expression for var in range]
    random_number = [random.randint(10, 100) for i in range(10)]
    print("生成的随机数为：", random_number)
    print('-' * 70)

    # 根据列表生成指定需求的列表，语法：newlist = [Expression for var in list]
    price = [random.randint(2000, 10000) for i in range(10)]
    sale = [int(x * 0.5) for x in price]
    print("原价格：", price)
    print("五折价格：", sale)
    print('-' * 70)

    # 从列表中选择符合条件的元素组成新的列表，语法：newlist = [Expression for var in list if condition]
    sale = [x for x in price if x > 5000]
    print("原列表：", price)
    print("价格高于 5000 的：", sale)
```

```
if __name__ == '__main__':
    list_reduction()
```

程序运行结果如下（未设置随机种子，每次随机生成的数据不同）：

```
生成的随机数为： [31, 90, 36, 98, 32, 87, 59, 34, 57, 34]
------------------------------------------------------------------------
原价格： [5939, 9781, 6168, 9684, 5290, 5984, 5650, 6315, 4260, 2129]
五折价格： [2969, 4890, 3084, 4842, 2645, 2992, 2825, 3157, 2130, 1064]
------------------------------------------------------------------------
原列表： [5939, 9781, 6168, 9684, 5290, 5984, 5650, 6315, 4260, 2129]
价格高于 5000 的： [5939, 9781, 6168, 9684, 5290, 5984, 5650, 6315]
```

3.3 元组

从元组和列表的定义上看，这两种结构比较相似，那么它们之间有哪些区别？主要区别在于元组是不可变序列，列表是可变序列，即元组中元素不可以单独修改，而列表则可以任意修改。

元组创建语法：**tuplename = (element 1, element 2, element 3, … ,element n)**

元组与列表的区别

- ✧ 列表是动态数组，它们不可变且可以重设长度（改变其内部元素的个数）。
- ✧ 元组是静态数组，它们不可变，且其内部数据一旦创建便无法改变。
- ✧ 元组缓存于 Python 运行时环境，这意味着我们每次使用元组时无须访问内核去分配内存。

两者在设计哲学上的不同：

- ✧ 列表可被用于保存多个互相独立对象的数据集合
- ✧ 元组用于描述一个不会改变的事务的多个属性

代码示例：

```
# -*- coding: UTF-8 -*-
"""
@author:Lenovo
@file_name:tuple_demo.py
@time:2021-09-17 15:46
@IDE:PyCharm
@copyright: http://maths.xynu.edu.cn
"""



def tuple_demo():
    num = (7, 14, 21, 28, 35, 42, 49, 56, 93)
    team = ("马刺", "火箭", "勇士", "湖人")
    untitle = ('Python', 28, "人生苦短，我用 Python", ["爬虫", "自动化运维", "云计算", "WEB 开发"])
    language = ('Python', "C++", '''JAVA''')
    team2 = "马刺", "火箭", "勇士", "湖人"  # 可以不用小括号括起来元素
    print("1、数值元组：", num)
```

```
    print("2、字符串元组：", team)
    print("3、复杂元组", untitle)
    print("4、访问：输出第一个元素：", untitle[0])
    print("5、访问：输出前四个元素：", untitle[:3])
    print("6、字符串元素使用单、双、三引号的元组：", language)
    print("7、不加小括号创建的元组：", team2)
    print("=" * 70)

    verse1 = ("世界杯冠军",)  # 包含只有一个元素的元组，元素后加逗号
    print("8、包含只有一个元素的元组，元素后加逗号，类型为：", type(verse1))
    verse2 = ("世界杯冠军")  # 不加逗号，表示字符串
    print("9、不加逗号，表示字符串，类型为", type(verse2))
    print("=" * 70)

    empty_tuple = ()  # 空元组
    print("10、输出空元组：", empty_tuple)
    del team2  # 删除元组
    print("=" * 70)

    language = language + ("R 语言", "SAS 语言")
    print("11、修改元素：", language)
    # TypeError: 'tuple' object does not support item assignment
    # language[0] = "I love Python"
    # TypeError: can only concatenate tuple (not "list") to tuple
    # language = language + ["R 语言", "SAS 语言"]
    print("=" * 70)
    print("12、JAVA 所在元组位置索引（从 0 开始）：", language.index("JAVA"))
    print("13、元组 untitle 中共有元素 Python 个数为：", untitle.count("Python"))


if __name__ == '__main__':
    tuple_demo()
```

程序运行结果如下：

```
1、数值元组： (7, 14, 21, 28, 35, 42, 49, 56, 93)
2、字符串元组： ('马刺', '火箭', '勇士', '湖人')
3、复杂元组 ('Python', 28, '人生苦短，我用 Python', ['爬虫', '自动化运维', '云计算', 'WEB 开发'])
4、访问：输出第一个元素： Python
5、访问：输出前四个元素： ('Python', 28, '人生苦短，我用 Python')
6、字符串元素使用单、双、三引号的元组： ('Python', 'C++', 'JAVA')
7、不加小括号创建的元组： ('马刺', '火箭', '勇士', '湖人')
======================================================================
8、包含只有一个元素的元组，元素后加逗号，类型为： <class 'tuple'>
9、不加逗号，表示字符串，类型为 <class 'str'>
======================================================================
```

```
10、输出空元组： ()
======================================================================
11、修改元素： ('Python', 'C++', 'JAVA', 'R 语言', 'SAS 语言')
======================================================================
12、JAVA 所在元组位置索引（从 0 开始）： 2
13、元组 untitle 中共有元素 Python 个数为： 1
```

（4）元组推导式子

- ✧ 生成器推导式的结果是一个生成器对象，而不是列表，也不是元组。
- ✧ 使用生成器对象的元素时，可以根据需要将其转化为列表或元组。
- ✧ 可以使用__next__()或者内置函数访问生成器对象，但不管使用何种方法访问其元素，当所有元素访问结束以后，如果需要重新访问其中的元素，必须重新创建该生成器对象。即生成器的对象好比一份蛋糕，第一个人吃完了就没了，必须重新制作另外一个蛋糕。
- ✧ 生成器对象创建与列表推导式不同的地方就是，生成器推导式是用圆括号创建。

代码示例：

```
# -*- coding: UTF-8 -*-
"""
@author:Lenovo
@file_name:tuple_reduction.py
@time:2021-09-17 15:54
@IDE:PyCharm
@copyright: http://maths.xynu.edu.cn
"""
import random  # 导入 random 标准库

rn = (random.randint(10, 100) for i in range(10))
print("生成的元组为：", rn)
print('-' * 50)

rn = tuple(rn)  # 转换为元组
print("转换为元组后：", rn)
print('-' * 50)

rn2 = (random.randint(10, 100) for _ in range(5))
print("输出第 1 个元素：", rn2.__next__())  # 输出第 1 个元素
print("输出第 2 个元素：", rn2.__next__())  # 输出第 2 个元素
print("输后第 3 个元素：", rn2.__next__())  # 输出第 3 个元素
rn2 = tuple(rn2)  # 转换为元组
print("转换后剩余元素：", rn2)
print('-' * 50)

number = (i for i in range(50, 90, 5))  # 生成生成器对象
for i in number:  # 遍历生成器对象
    print(i, end=" ")  # 输出每个元素的值
```

```
print("\n", tuple(number))  # 转换为元组输出
```

程序运行结果如下：

```
生成的元组为： <generator object <genexpr> at 0x0000024F3258CAC0>
------------------------------------------------
转换为元组后： (23, 52, 94, 44, 13, 67, 41, 21, 34, 78)
------------------------------------------------
输出第 1 个元素： 25
输出第 2 个元素： 74
输后第 3 个元素： 97
转换后剩余元素： (56, 74)
------------------------------------------------
50 55 60 65 70 75 80 85
 ()
```

实战小任务：QQ 运动周报

```
# -*- coding: UTF-8 -*-
"""
@author:Lenovo
@file_name:qq_sport_report.py
@time:2021-09-17 15:58
@IDE:PyCharm
@copyright: http://maths.xynu.edu.cn
"""



def sport_report(cur_week_sports, last_week_sports):
    print("本周运动周报：", cur_week_sports)
    print("上周运动周报：", last_week_sports)
    print('--' * 35)
    collact_sports = []
    for cur, last in zip(cur_week_sports, last_week_sports):
        sum_sports = cur + last
        collact_sports.append(sum_sports)  # 同步汇总步数
    print("周报汇总结果 ：", collact_sports)
    collact_sports.sort()
    print("升序周报汇总：", collact_sports)
    collact_sports.sort(reverse=True)
    print("降序周报汇总：", collact_sports)
    print('--' * 35)
    weeks = ["周日", "周一", "周二", "周三", "周四", "周五", "周六"]
    weeks2 = weeks.copy()
    max_sports = max(cur_week_sports)
    min_sports = min(cur_week_sports)
    index_max = cur_week_sports.index(max_sports)
```

```
    index_min = cur_week_sports.index(min_sports)
    weeks.insert(index_max + 1, max_sports)
    weeks.insert(index_min + 1, min_sports)
    print("周运动列表，包含最大步数和最小步数：")
    print(weeks)
    print('--' * 35)
    high_steps_val, high_steps_date = [], []
    for item in cur_week_sports:
        if item > 8500:
            high_steps_val.append(item)
            i = cur_week_sports.index(item)
            high_steps_date.append(weeks2[i])
    print("本周高于 8000 步的步数值：", high_steps_val)
    print("本周高于 8000 步的日期为：", high_steps_date)
    del collact_sports
    del high_steps_val
    del high_steps_date


if __name__ == '__main__':
    current_week_sports = [4235, 10111, 8447, 9566, 9788, 8951, 9808]  # 本周运动周报
    last_week_sports = [4235, 5612, 8447, 11250, 9211, 9985, 3783]  # 上周运动周报
    sport_report(current_week_sports, last_week_sports)
```

程序运行结果如下：

```
本周运动周报：  [4235, 10111, 8447, 9566, 9788, 8951, 9808]
上周运动周报：  [4235, 5612, 8447, 11250, 9211, 9985, 3783]
----------------------------------------------------------------------
周报汇总结果 ：  [8470, 15723, 16894, 20816, 18999, 18936, 13591]
升序周报汇总：  [8470, 13591, 15723, 16894, 18936, 18999, 20816]
降序周报汇总：  [20816, 18999, 18936, 16894, 15723, 13591, 8470]
----------------------------------------------------------------------
周运动列表，包含最大步数和最小步数：
['周日', 4235, '周一', 10111, '周二', '周三', '周四', '周五', '周六']
----------------------------------------------------------------------
本周高于 8000 步的步数值：  [10111, 9566, 9788, 8951, 9808]
本周高于 8000 步的日期为：  ['周一', '周三', '周四', '周五', '周六']
```

本章小结：

本章讲解了序列容器，包括列表和元组。列表和元组最大的区别为可变与不可变，但其中的元素可以重复，这与集合不同。本章以案例的形式讲解了列表和元组的各种操作方法，主要包括创建、删除、索引、查找、遍历、列表推导式等。列表与元组作为数据的一种组织形式或数据结构，是 Python 课程的重要内容，后续章节会经常使用。

此外，本章节主要以案例形式讲解了列表序列的主要操作函数，对于简易的数据管理系统进行设计和实

现，要求学生掌握列表的基本操作，理解简易系统算法设计的核心思路，会使用列表进行 Python 程序设计，实现各种功能。

作　业	课下完成教案中代码，并仿照修改。完成实验指导书相关实验。
主要 参考资料	[1] 明日科技,王国辉,李磊,冯春龙. Python 从入门到项目实践. 吉林：吉林大学出版社，2018. [2] 明日科技. Python 项目开发案例集锦. 吉林：吉林大学出版社，2019. [3] 李刚. 疯狂 Pyhton 讲义[M]. 电子工业出版社，2019
课后自我总 结分析	
备注	