

2018 年春季《大学计算机基础》（理科）实验指导书

实验 3 问题的描述—数据结构（1）

1. 实验目的

- （1）了解如何用简单的数据结构描述问题
- （2）掌握用 Python 语言内置的几种基本数据结构（列表、元组、字典）解决问题的方法。

2. 实验任务

实验任务 3-1 GPA 计算

题目描述：

GPA，或称平均学分绩点，是比较不同学生学习整体成果的指标。北航的 GPA 采用 4 分制，算法如下：

设百分制成绩为 x ，相应的 $GPA = 4 - 3 \times \frac{(100-x)^2}{1600}$ ($60 \leq x \leq 100$)，60 分 GPA 为 1，60 分以下为 0。现输入 N 个百分制成绩 $x_1, x_2 \dots x_N$ ($1 \leq N \leq 100$, N 为正整数) 对应的学分分别为 $h_1, h_2 \dots h_N$ 。请编程计算总 GPA，保留到小数点后三位 ($0 \leq x \leq 100$, x 为整数; $0.5 \leq h \leq 6$, h 为 0.5 的整数倍)。总 GPA 的计算公式为

$$\frac{GPA_1 h_1 + GPA_2 h_2 + \dots + GPA_N h_N}{h_1 + h_2 + \dots + h_N}$$

输入格式：

输入数据包含 $2N+1$ 行。

第 1 行为一个正整数 N ，表示百分制成绩的个数。

第 2 至 $N+1$ 行为 N 个百分制成绩 $x_1, x_2 \dots x_N$ 。

第 $N+2$ 至第 $2N+1$ 行为对应的学分 $h_1, h_2 \dots h_N$ 。

输出格式：

输出数据包含一行，为计算出的总 GPA（保留到小数点后三位）。

实验指导：

1. 列表的使用方法

python 中的列表（list）是最基本的数据结构，用于存储数据。序列中的每一个元素都被分配了一个数字，作为索引地址，默认第一个索引是 0，第二个索引是 1，以此类推。

创建一个列表，只要把逗号分隔的不同的数据项使用方括号括起来即可。如下所示：

```
list1 = ['physics', 'chemistry', 1997, 2000]
```

```
list2 = [1, 2, 3, 4, 5 ]
```

如果要创建一个空列表，可以直接使用函数 `list()` 或使用一组空的方括号，例如：

```
list3 = []
```

或

```
list3 = list()
```

同一个列表内的各项数据可以是不同类型的，例如整数、字符串等。如果需要访问列表内的数据，使用索引地址进行，例如：

```
print(list1[0], list1[1])
```

显示得到的结果为：

```
physics chemistry
```

如果要向列表内添加数据，可以使用 `append()` 函数，数据会被自动添加到列表末尾，并顺次赋予一个新的索引地址，例如：

```
list1.append('math')
```

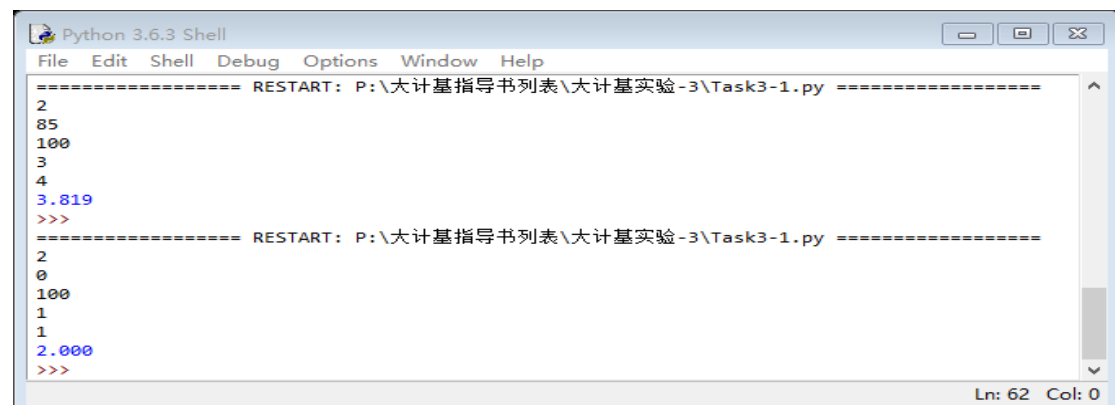
```
print(list1[4])
```

显示得到的结果为：

```
math
```

2.

参考运行结果：



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: P:\大计基指导书列表\大计基实验-3\Task3-1.py =====
2
85
100
3
4
3.819
>>>
===== RESTART: P:\大计基指导书列表\大计基实验-3\Task3-1.py =====
2
0
100
1
1
2.000
>>>
```

实验任务 3-2 学生信息

题目描述:

每个学生都有一个学号和姓名。请你编写一个程序，输入 n 个学生的学号和姓名，记录下来，再输入 k 个学号进行查询，对于每个学号，如果在查询中则输出对应学生姓名，否则输出错误信息“Not Found!”

输入格式:

输入数据包含 $2n+k+2$ 行:

第一行为一个整数 n ，表示学生数量;

接下来 $2n$ 行，为 n 个学生的信息，每两行依次为一个学生的学号和姓名，学号保证由大小写字母和数字组成，姓名保证由大小写字母、数字、空格、下划线组成;

接下来一行，为一个整数 k ，表示待查询的学号个数;

接下来 k 行，为查询的 k 个学号，保证由大小写字母和数字组成。

输出格式:

输出数据包含 k 行，每行包含一个字符串，对每个查询，分别输出对应学生姓名或“Not Found!”。

实验指导:

python 中的字典(dict)是基本数据结构之一，用于存储映射型数据。创建一个字典，可以将用逗号分隔的映射对用大括号包住，如:

```
dic = {'one': 1, 'two': 2, 'three': 3}
```

其中，key（即每个映射对冒号前的部分）不能为列表、字典这种易于增删的值，而应是字符串、元组或数值；value（即每个映射对冒号后的部分）没有类型要求。

如果要创建一个空字典，可以使用 dict() 或者一对空的大括号:

```
dic1 = dict()
```

```
dic2 = {}
```

```
print(dic1 == dic2) #输出 True
```

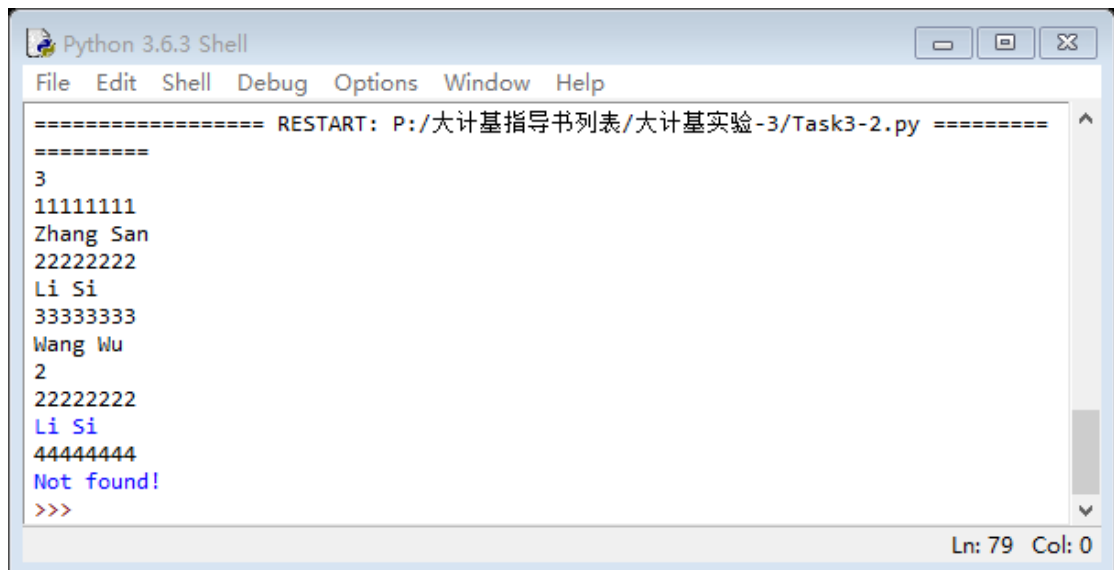
添加字典元素时，可直接使用方括号，如 `dic['zero'] = 0`；修改某个 key 对应的 value 也可以使用相同的方法。

当使用 `dic[key]` 获取(而不是添加、修改)字典中的值，也就是说 `dic[key]` 出现在=左

端以外的地方的时候，如果字典中没有对应的键值，程序会报错中止。为了避免这种状况，可以利用 `dic.get(key,default)` 函数查找对应键的值，若存在则返回对应值，不存在则返回 `default` 参数的值。在这个函数中 `default` 参数可以省略，默认为 `None`，如：

```
a = dic.get('one',-1) # a 为 1
b = dic.get('seven',-1) # b 为 -1
dic['seven'] = 7
c = dic.get('seven',-1) # c 为 7
```

参考运行结果：



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: P:/大计基指导书列表/大计基实验-3/Task3-2.py =====
=====
3
11111111
Zhang San
22222222
Li Si
33333333
Wang Wu
2
22222222
Li Si
44444444
Not found!
>>>
```

注意：在网络 OJ 评测时，评测系统只检验输出内容是否正确，与你输出的时间无关；也就是说，无论你是读入储存所有查询再依次计算，还是对每次查询立即计算，其结果对于 OJ 系统来说均是一致的。

实验任务 3-3 命名风格

题目描述:

在编程中,保持良好的编程习惯是很重要的,这其中包括一点就是变量的命名风格。合适的变量命名风格能使代码可读性更高。一种命名风格是将每个单词的首字母大写,例如 `DataBaseUser`,由于大小写字母连在一起看起来特别像骆驼的驼峰,所以这种命名风格被称作驼峰命名法(`CamelCase`)。另外一种下划线命名方法是保持单词小写,在单词之间用下划线连接,例如 `data_base_user`。请你编写一个函数,实现将驼峰命名法转变为下划线命名法。

输入格式:

输入数据包含一行,为一个用驼峰命名法命名的变量名。

输出格式:

输出数据包含一行,为转换后的用下划线命名法命名的变量名。

实验指导:

1. 设计思路

观察两种命名法,可以发现两种命名法互相转化的方法。于是,我们只需将输入的字符串转化为列表,逐个字符处理后,将列表中重新调整的各项连接即可。

2. 字符串常用处理方法

通过 `list()` 可以将输入的字符串转换为列表来进行操作。`list("abc")` 的结果为 `['a', 'b', 'c']`。将列表各项连接成字符串的函数语法为: `sep.join(seq)`

参数说明

`sep`: 分隔符,可以为空 (`''`)

`seq`: 要链接的元素序列,可以为字符串、元组、列表等。

以 `sep` 作为分隔符,将 `seq` 所有的元素合并成一个新的字符串。

返回值: 返回一个以分隔符 `sep` 连接各个元素后生成的字符串。

如以下程序:

```
l=['2','3','4']
```

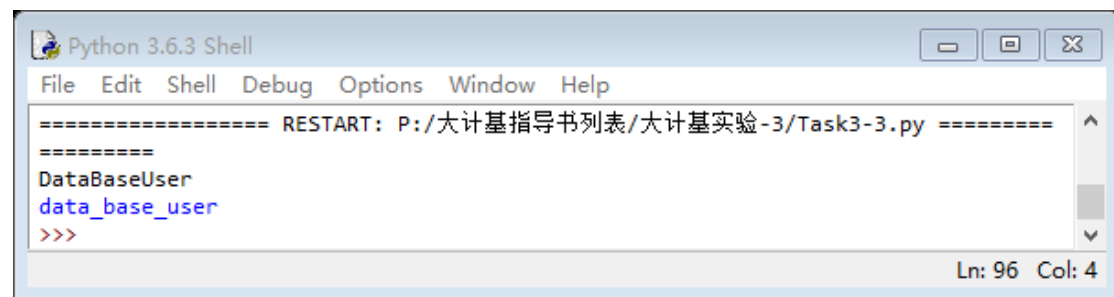
```
t=' '      #此处为一个空格构成的字符串,不是空串
```

```
s=t.join(l)
```

```
print(s)
```

你将获得“2 3 4”作为输出结果。

参考运行结果:



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: P:/大计基指导书列表/大计基实验-3/Task3-3.py =====
=====
DataBaseUser
data_base_user
>>>
Ln: 96 Col: 4
```

实验任务 3-4 简单数据管理

题目描述:

```
political_divisions = [  
    {  
        'name': 'Alabama', 'abbreviation': 'AL', 'counties': [  
            'Autauga', 'Baldwin', 'Barbour', 'Bibb'  
        ]  
    },  
    {  
        'name': 'Alaska', 'abbreviation': 'AK', 'counties': [  
            'Aleutians East', 'Anchorage', 'Bristol Bay'  
        ]  
    }  
]
```

上面的程序段定义了一个列表，包含美国阿拉巴马州、阿拉斯加州的部分郡级行政单位信息。我们提供了 `political_divisions.py` 程序，定义了 `political_divisions` 列表，以类似的格式包含了美国七个州的部分行政区信息。

`political_divisions` 列表包含若干项目，每个项目为一个字典项，表示一个州份。每个州份分别包含三个字典项：

`name` 项对应的值为一个字符串，表示州的名称；

`abbreviation` 项对应的值为一个字符串，表示州的缩写；

`counties` 项对应的值为一个列表，表示州所含的部分郡级行政单位名称。

现在要求你编写一个程序，可以对 `political_divisions.py` 程序中的 `political_division` 列表之内的信息进行两种查询操作：

操作 1：给定一个州的名称或缩写，并给出一个字母，输出字典内包含的所有首字母为该字母的郡（`counties`）的信息；

如：1 AK B 表示搜索所有阿拉斯加州以 B 开头的郡级行政单位名称；

操作 2：给出一个郡名称，输出字典内所有包含同名郡的州的缩写；

如：2 Washington 表示搜索所有包含名为 Washington 的郡级行政单位的州份缩写。

请你对用户的 n 个查询操作，分别完成查询操作。

输入格式：

输入数据共 $n+1$ 行：

第一行为一个整数 n ，表示所有查询的总次数；

接下来的 n 行，每行包含一个查询：

对于第一种查询，数字 1 后有一个空格，其后为表示州名称或缩写的字符串，由大小写字母组成，其后再跟一个空格，为查询的首字母，大小写均有可能；

对于第二种查询，数字 2 后有一个空格，其后为表示郡级行政单位名称的字符串，由大小写字母和空格组成。

输出格式：

输出包含 n 行，每行包含数个以英文逗号+空格隔开的字符串，包含每次查询的所有结果，以字典序排列；最后一个结果后没有逗号。

实验指导：

1. 设计思路

对于两种查询，我们的思路都是类似的，即遍历所有可能的选项，筛选出其中符合条件的，然后使用相应格式输出。对于第一种查询，我们可以遍历州份，找到对应的州，再从其郡的名单里筛选出符合要求首字母的郡名；对于第二种查询，我们需要遍历所有郡名，找到对应的州的缩写。

本题内容较为综合，不论是输入处理，还是最后排序输出，都包含了一些值得关注的细节。依写法不同，可能需要多重循环嵌套。

2. for 循环与数据结构配合使用

for 循环不仅可以和 range 结构搭配，也可以用于遍历数据结构，具体用法为 `for element in structure`。当 structure 是列表或元组时，element 依次被赋值为 structure 内部的各项内容的值；当 structure 是字典时，element 依次被赋值为字典的每个 key 值。

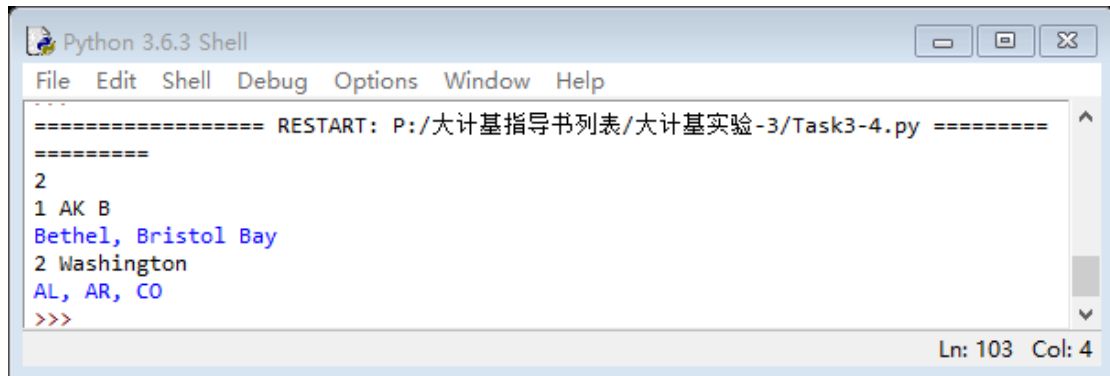
试研究以下程序：

```
odds = [1,3,5,7]
for num in odds:
    print(num)          #输出 1, 3, 5, 7
dishes = {'eggs': 2, 'sausage': 1}
for obj in dishes:
    print(obj)          #输出 eggs, sausage
for obj in dishes:
    print(dishes[obj])  #输出 2, 1
```


3. 内置的排序方法

对一个包含相同数据类型的列表，Python 提供了内置的排序方法。如对于列表 `lis`，`lis.sort()` 会直接调整 `lis` 的内部元素顺序，使其顺序从小到大。使用 `sorted(lis)` 则可以直接获得一份 `lis` 的有序拷贝。具体结果可以自行试验完成。

参考运行结果：



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: P:/大计基指导书列表/大计基实验-3/Task3-4.py =====
=====
2
1 AK B
Bethel, Bristol Bay
2 Washington
AL, AR, CO
>>>
Ln: 103 Col: 4
```

实验任务 3-5 质数的和（选做）

题目描述：

给定 2 个整数 a, b ，求出它们之间（含 a, b ）所有质数的和。

输入格式：

输入数据包含一行， 包含两个数字， 分别为 $a, b(a \leq b \leq 500000)$

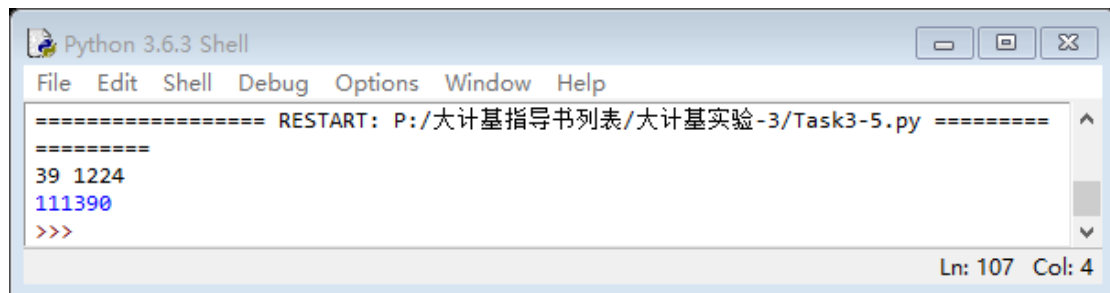
输出格式：

输出数据包含一行， 为一个整数， 表示 a, b 之间（含 a, b ）所有质数的和。

实验指导：

使用搜索引擎（如百度），查找在值很大的情况下，较快批量寻找质数的方法，理解该方法，并以此为基础编程。

参考运行结果：



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: P:/大计基指导书列表/大计基实验-3/Task3-5.py =====
39 1224
111390
>>>
Ln: 107 Col: 4
```

提示：本题测试数据很大，采用效率较低的方法很可能出现 TLE 错误。可以采用题目范围内较大的数据进行测试，如果运行时间较长，尽量考虑其他方法。效率较低但答案正确的方法也可以拿到 40%~80% 分数。