



第5章 组合数据类型

授课老师：刘国旭

潍坊科技学院

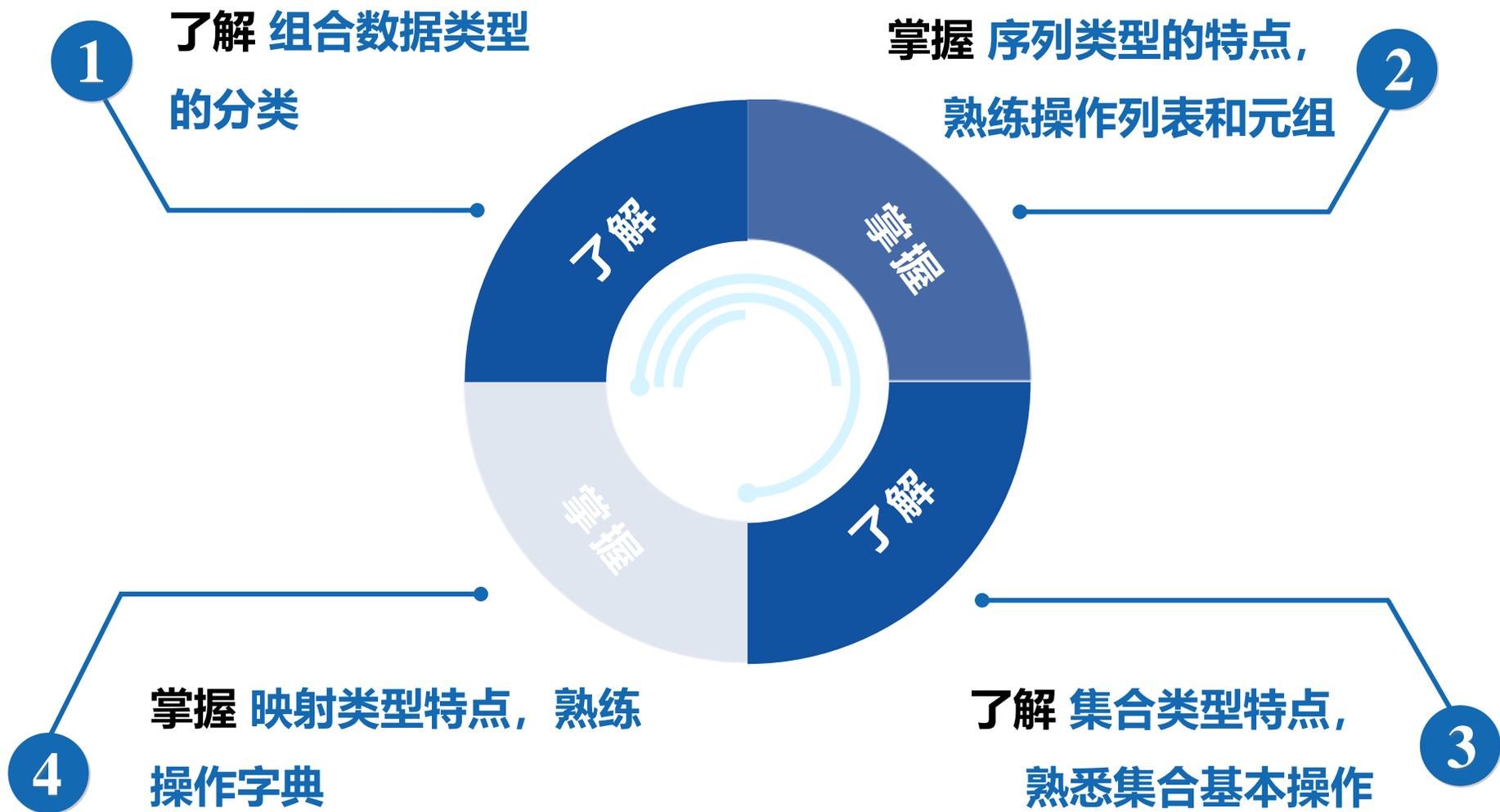


- 认识组合数据类型
- 列表
- 元组

- 集合
- 字典
- 组合数据类型与运算符



学习目标





目录页



潍坊科技学院
Weifang University of Science and Technology



5.1 认识组合数据类型

5.2 列表

5.3 元组

5.4 实训案例

5.5 集合



目录页



潍坊科技学院
Weifang University of Science and Technology



5.6 字典

5.7 实训案例

5.8 组合数据类型与运算符



目录页



潍坊科技学院
Weifang University of Science and Technology



5.1 认识组合数据类型

5.2 列表

5.3 元组

5.4 实训案例

5.5 集合



5.1 认识组合数据类型



思考：

什么是组合数据类型？



5.1 认识组合数据类型



组合数据类型可将多个相同类型或不同类型的数据组织为一个整体，根据数据组织方式的不同，Python的组合数据类型可分成三类：**序列类型**、**集合类型**和**映射类型**。

序列
类型

集合
类型

映射
类型



5.1 认识组合数据类型



- Python中常用的序列类型有**字符串 (str)**、**列表 (list)** 和**元组 (tuple)**。
- Python中的序列支持双向索引：**正向**递增索引和**反向**递减索引。正向递增索引**从左向右**依次**递增**，第一个元素的索引为**0**，第二个元素的索引为**1**，以此类推；反向递减索引**从右向左**依次**递减**，从右数第一个元素的索引为**-1**，第二个元素的索引为**-2**，以此类推。





5.1 认识组合数据类型



Python集合具备**确定性**、**互异性**和**无序性**三个特性。

Python要求放入集合中的元素必须是不可变类型，Python中的整型、浮点型、字符串类型和元组属于不可变类型，列表、字典及集合本身都属于可变的数据类型。

- **确定性**：给定一个集合，那么任何一个元素是否在集合中就确定了。
- **互异性**：集合中的元素互不相同。
- **无序性**：集合中的元素没有顺序，顺序不同但元素相同的集合可视为同一集合。



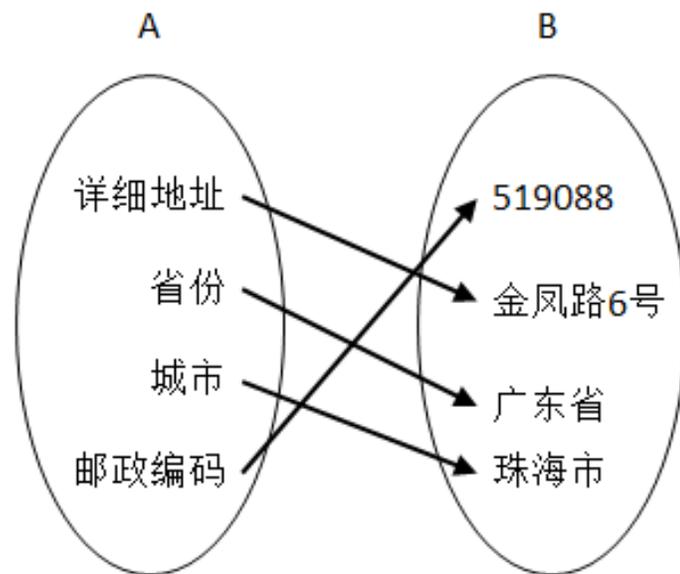
5.1 认识组合数据类型



映射类型以**键值对**的形式存储元素，键值对中的键与值之间存在映射关系。

字典（dict）是Python唯一的内置映射类型，字典的键必须遵守以下两个原则：

- 每个键只能对应一个值，不允许同一个键在字典中重复出现。
- 字典中的键是不可变类型。





目录页



潍坊科技学院
Weifang University of Science and Technology



5.1 认识组合数据类型

5.2 列表

5.3 元组

5.4 实训案例

5.5 集合



5.2.1 创建列表



Python列表的创建方式非常简单，既可以直接使用中括号“[]”创建，也可以使用内置的list()函数快速创建。

```
list_one = [] # 使用[]创建空列表  
list_two = ['p', 'y', 't', 'h', 'o', 'n']  
li_two = list() # 使用list()创建空列表  
li_two = list('python')
```

示例





多学一招：可迭代对象



- 在Python中，支持通过for...in...语句迭代获取数据的对象就是可迭代对象。
- 我们学习过可迭代的类型有字符串和列表，后续学习的集合、字典、文件也是可迭代类型的对象。
- 使用**isinstance()**函数可以判断目标是否为可迭代对象，返回True表示为可迭代对象。

```
from collections.abc import Iterable  
ls = [3,4,5]  
print(isinstance(ls, Iterable))
```

示例





5.2.2 访问列表元素



列表中的元素可以通过索引或切片这两种方式进行访问，也可以在循环中依次访问。

```
list_one = ["Java", "C#", "Python", "PHP"]
```

示例

```
print(list_one[1])
```

索引

```
print(list_one[1:])
```

切片

```
for li in list_one:  
    print(li, end=' ')
```

循环





5.2.3 添加列表元素



向列表中添加元素是非常常见的一种列表操作，Python提供了`append()`、`extend()`和`insert()`这几个方法向列表末尾、指定位置添加元素。

```
list_one = ["Java", "C#", "Python", "PHP"]
```

示例

```
list_one.append("C++")
```

append

在列表**末尾**添加元素

```
list_one.extend(["Android", "IOS",])
```

extend

在列表**末尾**添加另一个序列的所有元素

```
list_one.insert(2, "HTML")
```

insert

按照索引将元素插入列表的**指定位置**





5.2.4 元素排序



列表的排序是将元素按照某种规定进行排列。列表中常用的排序方法有 `sort()`、`reverse()`、`sorted()`。

```
li_one = [6, 2, 5, 3]
```

示例

```
list_one.sort(reverse=True)
```

sort

有序的元素会覆盖原来的列表元素，
不产生新列表

```
li_two = sorted(li_one)
```

sorted

产生排序后的新列表，排序操作不会对原列表产生影响

```
li_one.reverse()
```

reverse

逆置列表，即把原列表中的元素从右至左依次排列存放





5.2.5 删除列表元素



删除列表元素的常用方式有del语句、remove()方法、pop()方法和clear()方法。

```
li_one = [6, 2, 5, 3, 3]
```

示例

```
del li_one[0]
```

del

删除列表中**指定位置**的元素

```
li_one.remove(3)
```

remove

移除列表中匹配到的**第一个元素**

```
li_one.pop()
```

pop

移除列表中的**某个元素**，若未指定具体元素，则移除列表中的**最后一个元素**

```
li_one.clear()
```

clear

清空列表





5.2.6 列表推导式



- 列表推导式是符合Python语法规则的**复合表达式**，它用于以简洁的方式根据已有的列表构建**满足特定需求的列表**。

```
[exp for x in list]
```

格式

- 列表推导式还可以结合**if判断语句**或**for循环嵌套**，生成更灵活的列表。
 - 带有if语句的列表推导式
 - for循环嵌套的列表推导式
 - 带有if语句与for循环嵌套的列表推导式





5.2.6 列表推导式



```
ls = [1,2,3,4,5,6,7,8]
ls = [data*data for data in ls]
print(ls)
```

示例



目录页



潍坊科技学院
Weifang University of Science and Technology



5.1 认识组合数据类型

5.2 列表

5.3 元组

5.4 实训案例

5.5 集合



5.3 元组



- 元组的表现形式为一组包含在圆括号 “()” 中、由逗号分隔的元素，元组中元素的个数、类型不受限制。
- 使用圆括号可以直接创建元组，还可以使用内置函数 `tuple()` 构建元组。

```
t1 = ()           # 空元组
t2 = (1,)         # 包含单个元素的元组
t1 = tuple()      # 创建空元组
t2 = tuple([1,2,3]) # 利用列表创建元组
```

示例





5.3 元组



当使用圆括号 “()” 创建元组时，如果元组中只包含一个元素，那么需要在该**元素的后面添加逗号**，从而保证Python解释器能够识别其为元组类型。

```
t1 = ('python')
```

```
t2 = ('python',)
```

```
print(type(t1))
```

```
print(type(t2))
```

示例

```
<class 'str'>
```

```
<class 'tuple'>
```

结果





5.3 元组



Python支持通过索引与切片访问元组的元素，也支持在循环中遍历元组。

```
tuple_demo = ('p','y','t','h','o','n')
```

示例

```
tuple_demo[2]
```

使用索引

```
tuple_demo[2:5]
```

使用切片

```
for i in tuple_demo:  
    print(i)
```

遍历元组





目录页



潍坊科技学院
Weifang University of Science and Technology



5.1 认识组合数据类型

5.2 列表

5.3 元组

5.4 实训案例

5.5 集合



5.4.1 十大歌手



为丰富校园文化生活，学校拟组织一场歌手大赛，从参赛选手中选拔出十名相对突出的学生，授予“校园十大歌手”称号。比赛之中设置有评委组，每名选手演唱完毕之后会由评委组的十名评委打分。为保证比赛公平公正，防止作弊和恶意打分，计算得分时会先去掉最高分和最低分，再计算平均分。

本实例要求编写程序，实现根据需求计算平均分的功能。





5.4.2 神奇魔方阵



魔方阵又称纵横图，是一种 n 行 n 列、由自然数 $1 \sim n \times n$ 组成的方阵，该方阵中的数符合以下规律：

1. 方阵中的每个元素都不相等。
2. 每行、每列以及主、副对角线上的个元素之和都相等。

本实例要求编写程序，输出一个5行5列的魔方阵。



目录页



潍坊科技学院
Weifang University of Science and Technology



5.1 认识组合数据类型

5.2 列表

5.3 元组

5.4 实训案例

5.5 集合



5.5 集合



- Python的集合 (**set**) 本身是**可变类型**，但Python要求放入集合中的元素必须是不可变类型。
- 集合类型与列表和元组的区别是：集合中的元素无序但必须**唯一**。
- 集合的表现形式为一组包含在大括号 “{}” 中、由逗号 “,” 分隔的元素。使用 “{}” 可以直接创建集合，使用内置函数**set()**也可以创建集合。

```
s1 = {1}
s2 = {1,'b',(2,5)}
s = set()
s3 = set('python')
```

示例





5.5 集合



需要注意，使用{}不能创建空集合（不包含元素的{}创建的是字典变量），空集合只能利用set()函数创建。

```
set_demo1 = {}  
set_demo2 = set()  
print(type(set_demo1))  
print(type(set_demo2))
```

示例

```
<class 'dict'>  
<class 'set'>
```

结果





5.5 集合



集合是可变的，集合中的元素可以动态**增加**或**删除**。Python提供了一些内置方法来操作集合，常见内置方法如下：

常见操作	说明
<code>S.add(x)</code>	向集合 <code>S</code> 中添加元素 <code>x</code> ， <code>x</code> 已存在时不做处理
<code>S.remove(x)</code>	删除集合 <code>S</code> 中的元素 <code>x</code> ，若 <code>x</code> 不存在则抛出 <code>KeyError</code> 异常
<code>S.discard(x)</code>	删除集合 <code>S</code> 中的元素 <code>x</code> ，若 <code>x</code> 不存在不做处理
<code>S.pop()</code>	随机返回集合 <code>S</code> 中的一个元素，同时删除该元素。若 <code>S</code> 为空，抛出 <code>KeyError</code> 异常
<code>S.clear()</code>	清空集合 <code>S</code>
<code>S.copy()</code>	拷贝集合 <code>S</code> ，返回值为集合
<code>S.isdisjoint(T)</code>	判断集合 <code>S</code> 和 <code>T</code> 中是否没有相同的元素，没有返回 <code>True</code> ，有则返回 <code>False</code>



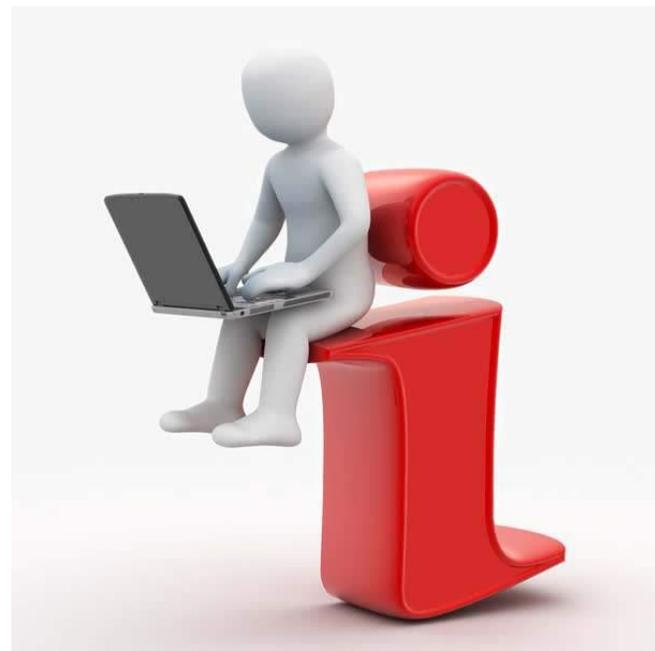
5.5 集合



集合也可以利用推导式创建，集合推导式的格式与列表推导式相似，区别在于**集合推导式外侧为大括号“{}”**。

```
{exp for x in set if cond}
```

格式





5.5 集合



```
ls = [1,2,3,4,5,6,7,8]
```

```
s = {data for data in ls if data%2==0}
```

```
print(s)
```

```
{8,2,4,6}
```

示例



目录页



潍坊科技学院
Weifang University of Science and Technology



5.6 字典

5.7 实训案例

5.8 组合数据类型与运算符



5.6.1 创建字典



提到字典这个词相信大家都不会陌生，学生时期碰到不认识的字时，大家都会使用字典的部首表查找对应的汉字。Python中的字典数据与学生使用的字典有类似的功能，它以“**键值对**”的形式组织数据，利用“**键**”快速查找“**值**”。通过“**键**”查找“**值**”的过程称为**映射**，Python中的**字典**是典型的映射类型。





5.6.1 创建字典



字典的表现形式为一组包含在大括号 “{}” 中的**键值对**，每个键值对为一个字典元素，每个元素通过逗号 “,” 分隔，每对键值通过 “:” 分隔。

```
{键1:值1, 键2:值2,...,键N:值N}
```

格式

字典的值可以是任意类型，但键不能是列表或字典类型。字典像集合一样使用 “{}” 包裹元素，它也具备类似集合的特点：字典**元素无序**，**键值必须唯一**。



5.6.1 创建字典



使用 “{}” 可以直接创建字典，还可以使用内置函数 `dict()` 创建字典。

```
d1 = {}           # 创建空字典  
d2 = {'A': '123', 'B': '135', 'C': '680'}  
d3 = {'A': 123, 12: 'python'}
```

示例

```
d4 = dict()  
d5 = dict({'A': '123', 'B': '135'})
```

示例



5.6.2 字典的访问



字典的值可通过“键”或内置方法`get()`访问。

```
d2 = dict({'A': '123', 'B': '135'})
```

示例

```
d2['A']
```

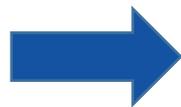
键



'123'

```
d2.get('B')
```

get()



'135'





5.6.2 字典的访问



字典涉及的数据分为**键**、**值**和**元素**（键值对），除了直接利用键访问值外，Python还提供了内置方法**keys()**、**values()**和**items()**。

```
info = {'name': 'Jack', 'age': 23, 'height': 185}
```

示例

```
info.keys()
```

获取所有键

```
dict_keys(['name', 'age', 'height'])
```

```
info.values()
```

获取所有值

```
dict_values(['Jack', 23, 185])
```

```
info.items()
```

获取所有元素

```
dict_items([('name', 'Jack'), ('age', 23), ('height', 185)])
```



5.6.3 字典元素的添加和修改



字典支持通过为**指定的键赋值**或使用**update()**方法添加或修改元素。

- 通过键添加元素：**字典变量[键] = 值**
- 使用update()添加元素：**dict.update(key=value)**





5.6.3 字典元素的添加和修改



添加字典元素

```
add_dict = {'name': 'Jack', 'age': 23, 'height': 185}
```

示例

```
add_dict['sco'] = 98
```

通过键添加

```
add_dict.update(sco=98)
```

使用update方法添加





5.6.3 字典元素的添加和修改

修改字典元素的本质是通过键获取值，再重新对元素进行赋值。修改元素的操作与添加元素的操作相同。

```
modify_dict = {'stu1': '小明', 'stu2': '小刚', 'stu3': '小兰'}
```

示例

```
modify_dict['stu3'] = '刘婷'
```

通过键修改

```
modify_dict.update(stu2='张强')
```

使用update方法修改





5.6.4 字典元素的删除



Python支持通过`pop()`、`popitem()`和`clear()`方法删除字典中的元素。

- ✓ `pop()`: 根据指定键值删除字典中的指定元素
- ✓ `popitem()`: 随机删除字典中的元素
- ✓ `clear()`: 清空字典中的元素





5.6.4 字典元素的删除



```
per_info = { '001' : '张三' , '002' : '李四' ,  
            '003' : '王五' , '004' : '赵六' }  
print(per_info.pop( '001' ))  
print(per_info)
```

示例



5.6.5 字典推导式



- 字典推导式的格式、用法与列表推导式类似，区别在于字典推导式**外侧为大括号“{}”**，且内部需**包含键和值两部分**。

```
{new_key:new_value for key,value in dict.items()}
```

格式

- 利用字典**推导式**可快速**交换**字典中的**键和值**。

```
old_dict = {'name': 'Jack','age':23,'height':185}  
new_dict = {value:key for key,value in old_dict.items()}  
print(new_dict)
```

示例



```
{'Jack': 'name',  
23: 'age',  
185: 'height'}
```



目录页



潍坊科技学院
Weifang University of Science and Technology



5.6 字典

5.7 实训案例

5.8 组合数据类型与运算符



5.7.1 青春有你



如今两年偶像选秀节目风头正盛，吸引了许多喜欢唱跳、有一颗明星梦想的少年少女参加，青春有你正是节目之一。青春有你采用**计票机制**，选手获得的票数越多，排名就越靠前。

本实例要求编写程序，接收选手的姓名和票数，输出**排序后的成绩**。



5.7.2 手机通讯录



通讯录是记录了联系人姓名和联系方式的名录，手机通讯录是最常见的通讯录之一，人们可以在通讯录中通过姓名查看相关联系人的联系方式等信息，也可以在其中新增联系人，或修改、删除联系人信息。

本实例要求编写程序，实现具备**添加**、**查看**、**修改**以及**删除**联系人信息功能的手机通讯录。



目录页



潍坊科技学院
Weifang University of Science and Technology



5.6 字典

5.7 实训案例

5.8 组合数据类型与运算符



5.8 组合数据类型与运算



Python中针对数字类型的运算符对组合数据类型同样适用，包括+、*、in、not in。

- 字符串、列表和元组使用 “+” 运算符，会对数据进行拼接。
- 字符串、列表和元组使用 “*” 运算符，会对数据进行整数倍拼接。
- “in” “not in” 运算符称为成员运算符，用于判断某个元素是否属于某个变量。





5.9 本章小结



本章首先带领大家简单认识了Python中的**组合数据类型**，然后分别介绍了Python中常用的组合数据类型：**列表**、**元组**、**集合**和**字典**的创建和使用，并结合**实例案例**帮助大家巩固这些数据类型，最后介绍了**组合数据类型与运算符**的相关知识。通过本章的学习，希望大家能掌握并熟练运用Python中的组合数据类型。