

Python 语言程序设计

第五章

组合数据结构



5.1 组合类型简介

01

02

5.2 列表

5.3 元组

03

04

5.4 字典

5.5 集合

05

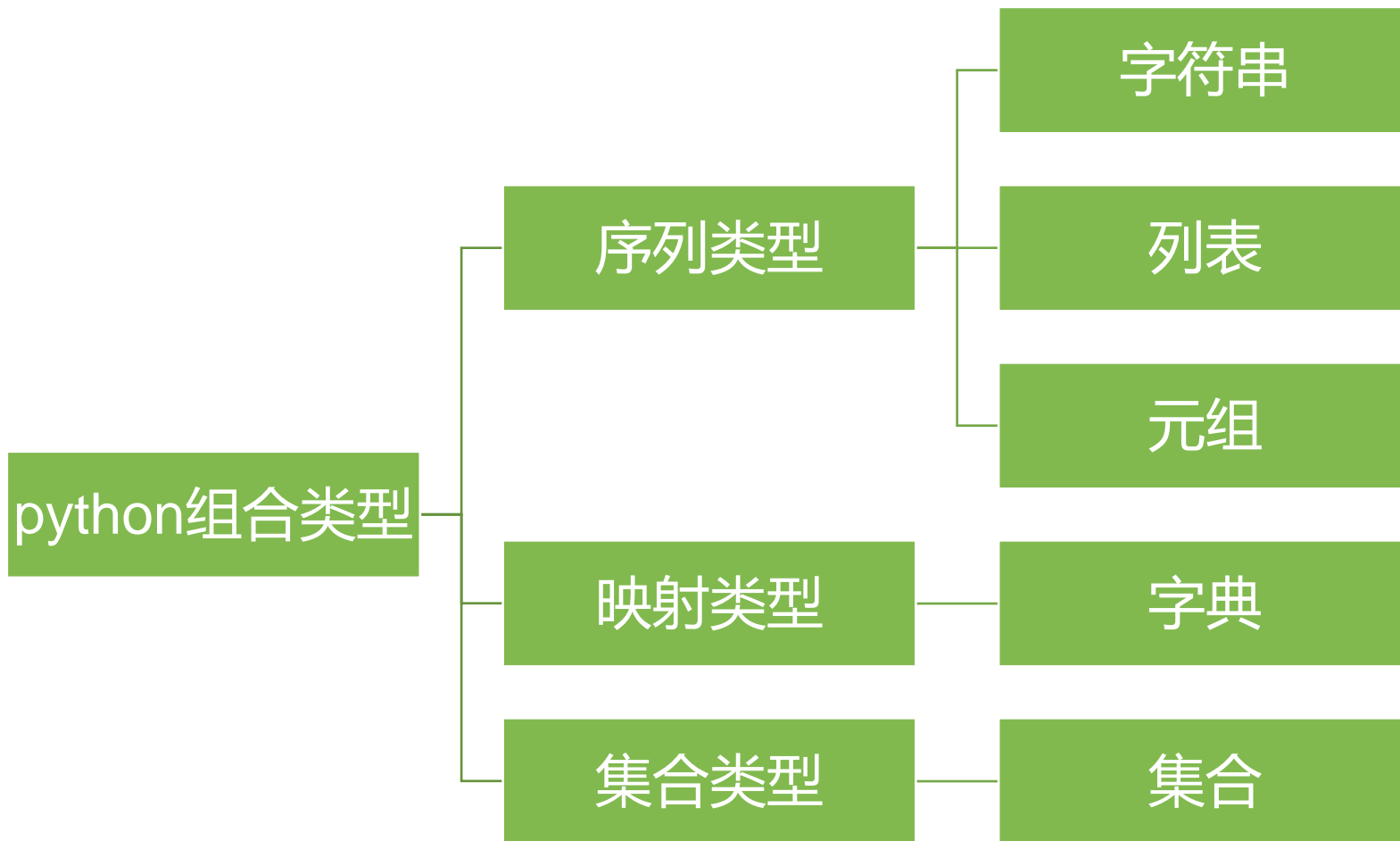


5.1 组合类型简介

5.1 组合类型简介

- Python可以处理的数据如整型、字符串类型和逻辑型等只能表示1个单一的数据，称为基本数据类型。
- 在处理多个有关联的数据的时候，仅仅使用基本数据类型是不够的。除了这些简单数据类型外，Python语言还可以处理一些复杂的数据类型，称为组合数据类型。

Python组合类型分类



5.2列表



列表简介

- 列表是一种可变序列数据类型。列表将数据元素放在一对中括号之间，并使用逗号作为数据元素的分割。一个列表中的数据元素可以是基本数据类型，也可以是组合数据类型或自定义数据类型的数据。例如下面的列表都是合法的列表对象：
- [1,2,3,4,5,6,7]
- [1, "10086","中国移动",True]
- ["日期","中国",[2017,5,1]]
- [1,[2,3,4],(5,6),{"a":7,"b":8,"c":9}]

5.2.1 创建列表

- 创建一个列表，可以通过使用方括号，并且把方括号内每一个列表元素用逗号分开进行创建，或者使用list()函数进行创建。

```
>>> list1=[]                #产生一个空列表
```



```
>>> list1
```

```
[]
```

```
>>> list2=[1,2,3,4,5,6]
```

```
>>> list2
```

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



```
>>> list3=['Python','c++','Java','VB','Perl']
```

```
>>> list3
```

```
['Python', 'c++', 'Java', 'VB', 'Perl']
```

```
>>> list4=list()          #产生一个空列表，等价于list1=[]
```

```
>>> list4
```

```
[]
```

```
>>> list5=list(range(1,10,2))    #将range函数产生的序列变为列表
```




```
>>> list5
```

```
[1, 3, 5, 7, 9]
```

```
>>> list6=list("沈阳师范大学")  #每字符作为列表中的一个数据元素
```

```
>>> list6
```

```
['沈', '阳', '师', '范', '大', '学']
```



例5.1使用列表推导式创建列表

```
>>> list1=[x*x for x in range(1,10)]
```

```
>>> list1
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
>>> list2=[i for i in list1 if i%2==0]
```

```
>>> list2
```

```
[4, 16, 36, 64]
```

```
>>> list3=[i for i in range(100,1000) if (int(i/100))**3 + (int(i/10)%10)**3 + (i%10)**3==i]
```

```
>>> list3
```

```
[153, 370, 371, 407]
```

5.2.2 访问列表

正向索引，从0开始递增					
0	1	2	3	4	5
['沈',	'阳',	'师',	'范',	'大',	'学']
-6	-5	-4	-3	-2	-1
逆向索引，从-1开始递减					



```
>>> list1=['a','b','c','d','e','f','g']
```

```
>>> list1[0]          #访问列表list1中正向索引序号为0的列表元素。
```

```
'a'
```

```
>>> list1[2:4]        #访问列表list1中正向索引序号从2到4（不包括）的列表元素。
```

```
['c', 'd']
```

```
>>> list1[-2]         #访问列表list1中逆向索引序号为-2的列表元素。
```

```
'f'
```

```
>>> list1[4:]         #访问列表list1中正向索引序号从4到最后的列表元素。
```

```
['e', 'f', 'g']
```




```
>>> list1[: -4]       #访问列表list1中从开始到逆向索引序号为-4（不包含）的列表元素。
```

```
['a', 'b', 'c']
```

```
>>> list1[8]
```

```
...
```

```
IndexError: list index out of range
```



例5.2 遍历二维列表

```
list1=['20200001','赵明','男',19,'物理学院','流体力学']
list2=['20200002','钱小红','女',20,'化学学院']
list3=['20200003','孙强','男',20,'信息学院','计算机科学与技术','3班']
list4=['20200004','李丽','女',19,'外语学院']
list=[list1,list2,list3,list4]
for i in range(len(list)):
    for j in range(len(list[i])):
        print("{}\t".format(list[i][j]),end="")
    print()
```

5.2.3 更新列表

1. 修改列表元素值

修改列表元素值可以用赋值语句，语法格式如下：

<列表名> [<索引>]=<数据值>

其中，列表名为一个已经存在的列表，索引为该列表的正向或逆向索引序号，数据值为任意数据值。

2. 添加列表元素

虽然在对列表赋值的时候可以添加数据元素，但通常还是使用专门的函数来对列表进行添加元素操作。`append()`函数可以向列表的最后添加列表元素；`insert()`函数可以向列表中指定索引序号处插入列表元素。

3. 删除列表元素

删除列表元素通常可以使用`remove()`函数或者`del`语句进行操作。
`remove()`函数用来按值删除列表中的列表元素，`del`语句可以按索引序号删除列表中的列表元素。

5.2.4 列表常用其它操作

操作	功能
<code>list1+list2</code>	连接两个列表，新列表的列表元素个数为list1和list2列表元素个数之和。
<code>list1*n</code> 或 <code>n*list1</code>	将list1重复n次。
<code>x in list1</code>	如果列表list1包含x，则返回True，否则返回False。
<code>x not in list1</code>	如果列表list1中不包含x对象则返回True，否则返回False。
<code>len(list1)</code>	函数返回list1中列表元素的个数。
<code>max(list1)</code>	函数返回list1列表中元素的最大值，要求list1中列表元素类型相同。
<code>min(list1)</code>	函数返回list1列表中元素的最小值，要求list1中列表元素类型相同。
<code>sorted(list1)</code>	函数返回1个新列表，将list1中的元素进行排序，关键字reverse=False或省略为升序排列；关键字reverse=True为降序排列。
<code>sum(list1)</code>	如果list1中所有列表元素都是数字，函数返回列表元素之和。

5.3元组



元组简介

- 元组和列表类似，但元组的元素是不可变的，元组一旦创建，不可以修改其元素，也不能添加或者删除元素。创建一个元组可以使用一对小括号，在小括号内用逗号分隔开元组元素。一个元组中的数据元素可以是基本数据类型，也可以是组合数据类型或自定义数据类型的数据。例如下面的元组都是合法的：

- `(1,2,3,4,5)` `(“Python”, “C#”, “Java”, “Go”, “VB”)`
- `()` `(5,)` `((1,2,3), (“a”, “b”, “c”), (True, False))`

5.3.1 创建元组

- 创建一个元组，可以通过使用小括号，并且把小括号内每一个元组元素用逗号分开进行创建，或者使用tuple()函数进行创建。

```
>>> tup1=()          #产生一个空元组
```

```
>>> tup1
```

```
()
```

```
>>> tup2=(1,2,3,4,5,)
```

```
>>> tup2
```

```
(1, 2, 3, 4, 5)
```

```
>>> tup3=('Python','c++','Java','VB','Perl')
```



```
>>> tup3
```

```
('Python', 'c++', 'Java', 'VB', 'Perl')
```

```
>>> tup4=tuple() #产生一个空元组，等价于tup1=()
```

```
>>> tup4
```

```
()
```

```
>>> tup5=tuple(range(1,10,2)) #将range函数产生的序列变为元组
```


```
>>> tup5
```

```
(1, 3, 5, 7, 9)
```

```
>>> tup6=tuple("沈阳师范大学") #每字符作为元组中的一个数据元素
```

```
>>> tup6
```

```
('沈', '阳', '师', '范', '大', '学')
```





元组推导式

- 和列表一样，元组一样也可以使用推导式来生成，语法也近似，只是不用中括号而用小括号：
- (表达式 for 变量 in 序列)
- 但是和列表推导式不同的是，这种推导式叫做生成器推导式，它的结果是一个生成器对象，而不是元组。生成器对象可以使用next()函数依次访问其中的元素，也可以使用list()函数或者tuple()函数转化为列表或者元组后使用。

5.3.2 访问元组

- 元组属于不可变序列，一旦创建，元组中的值就固定下来不可变，也无法为元组增加元素或者删除元素。列表可以通过切片访问为列表中增加或者删除元素，但是元组不可以。访问一个元组和访问列表的方法类似。例如：



```
>>> tuple1=('a','b','c','d','e','f','g')
```

```
>>> tuple1[1]          #访问元组tuple1中正向索引序号为1的元素。
```

```
'b'
```



```
>>> tuple1[3:5]        #访问元组tuple1中正向索引序号从3到4的元素。
```

```
('d', 'e')
```

```
>>> tuple1[-1]         #访问元组tuple中逆向索引序号为-1的元组。
```



```
'g'
```

```
>>>
```



元组常用操作

- 元组常用操作也是 “+”和 “*” ，元组常用函数也和列表常用函数基本一致，请参考列表操作

5.4 字典

字典简介

- 字典是一种映射型的组合数据结构，由键值对组成。
- 在一个字典结构中，一个键只能对应一个值，但是多个键可以对应相同的值。
- 字典类型和序列类型的区别在于存储和访问方式都不同。序列类型通常通过索引序号来访问，而且只采用整数作为索引序号；字典可以用任意不可变类型如数字、字符串、元组等作为值的索引序号，也就是“键”。

字典简介

- 字典将键值对放在一对大括号之间，并使用逗号作为分隔，每个键值对之间用冒号分隔。例如下面的字典都是合法的：

`{}`

`{1:"Python",2:"C++",3:"Java",4:"VB"}`

`{"No1":"Python","No2":"C++","No3":"Java","No4":"VB"}`

`{("三班",15):["张晓红","女"],("四班",22):["董强","男"],("五班",7):["张晓红","女"]}`

5.4.1 字典的创建

1. 通过赋值语句创建字典

```
>>> dict1={}
```

```
>>> dict1
```

```
{}
```

```
>>> dict2={'优':90,'良':80,'中':70,'及':60}
```

```
>>> dict2
```

```
{'优': 90, '良': 80, '中': '70', '及': 60}
```

2. 通过dict()函数创建字典

dict()函数可以将键值对形式的列表或元组创建为字典。例如：

```
>>> dict3=dict(['优', 90], ['良', 80], ['中', '70'], ['及', 60])
```

```
>>> dict3
```

```
{'优': 90, '良': 80, '中': '70', '及': 60}
```

```
>>> dict4=dict((('优', 90), ('良', 80), ('中', '70'), ('及', 60)))
```

```
>>> dict4
```

- 在dict()函数中调用zip()函数，可以将多个序列作为参数，返回由元组构成的列表。
- >>> dict5=dict(zip(['优', '良', '中', '及'],[90,80,70,60]))
- >>> dict5
- {'优': 90, '良': 80, '中': 70, '及': 60}

3. 通过fromkeys()函数创建字典

调用fromkeys()函数可以创建值都相同的字典，可以将一个序列作为键，然后指定一个统一的值。formkeys()函数也可以不指定值，创建的字典默认为None空值。例如：

```
>>> dict6={}.fromkeys(['优', '良', '中', '及'], "大于60分")
```

```
>>> dict6
```

```
{'优': '大于60分', '良': '大于60分', '中': '大于60分', '及': '大于60分'}
```

```
>>> dict7={}.fromkeys(['优', '良', '中', '及'])
```

```
>>> dict7
```

```
{'优': None, '良': None, '中': None, '及': None}
```

4. 通过推导式创建字典

也可以使用推导式创建字典，语法格式如下：

{键:值 for 变量 in 序列}

例如：

```
>>> dict8 = {n: n**2 for n in range(1,5)}
```

```
>>> dict8
```

```
{1: 1, 2: 4, 3: 9, 4: 16}
```

5.4.2 访问字典

- 1. 通过键访问值
- 因为字典的键值对是一种映射关系，根据“键”就可以访问到“值”。所以访问字典中键值对的值可以通过方括号并指定键的方式进行访问。如果“键”不存在，则访问错误。
- 字典对象也提供了一个get()方法来通过键访问对应的值：当键存在时，返回该键对应的值，而键不存在的时候也不会出错，而返回指定值。

2. 访问字典的所有键值对、所有键、所有值

通过调用字典的`items()`方法可以返回所有的键值对；调用`keys()`方法可以返回所有的键；而调用`value()`方法可以返回所有的值。

3. 遍历字典

一般通过for语句循环可以遍历字典的元素。

5.4.3 更新字典

1. 添加元素

①通过赋值语句可以向字典添加键值对元素。

②使用`setdefault()`函数向字典添加元素。

2. 合并字典

使用`update()`函数可以将一个字典中的元素添加到当前字典中，如果两个字典的键有重名，则用另一个字典中的值对当前字典进行更新。

3. 修改元素的值

修改字典中的元素的值可以用过赋值语句实现。

4. 删除元素

删除字典中的元素可以用del()函数、del语句、pop()函数、popitem()函数或者clear()函数进行。

5.4.4 字典的常见其他操作

操作	功能
<code>key in dict1</code>	如果字典dict1包含键key，则返回True，否则返回False。
<code>(key,value) in dict1.items()</code>	如果字典dict1包含键值对(key,value)，则返回True，否则返回False。
<code>value in dict1.values()</code>	如果字典dict1包含值value，则返回True，否则返回False。
<code>len(dict1)</code>	求字典dict1中的元素个数。
<code>max(dict1)</code>	求字典dict1键的最大值，要求所有键数据类型相同。
<code>min(dict1)</code>	求字典dict1键的最小值，要求所有键数据类型相同。
<code>list(dict1)</code>	返回由字典dict1中的键组成的列表
<code>list(dict1.items())</code>	返回由字典dict1的元素组成的列表，键值对转化为元组(key,value)。
<code>list(dict1.values())</code>	返回由字典dict1的值组成的列表。

例5.3随机产生500个小写字母，统计每个字母出现的频率。

```
from random import choice
str='abcdefghijklmnopqrstuvwxyz'
dict1=dict()
for i in range(500):
    r=choice(str)
    dict1[r]=dict1.get(r,0)+1
print('一共产生的字母为：',sum(dict1.values()))
print('每个字母产生的词频为：',dict1)
```

5.5 集合



集合简介

- 集合是无序的可变序列，集合元素放在一对大括号间（和字典一样），元素之间用逗号分隔。在一个集合中，元素不允许重复。集合的元素类型只能是固定的数据类型，如整型、字符串、元组等，而列表、字典等是可变数据类型，不能作为集合中的数据元素。例如下面的集合都是合法的：

`{1,2,3,4,5,6,7}`

`{"a","b","c","d","e"}`

`{(1, 2), (1, 3), (2, 1), (2, 3), (2, 2), (1, 1)}`

5.5.1 创建集合

- 创建可变集合可以使用赋值语句或者set()函数创建；创建不可变集合可以使用frozenset()创建。

1. 通过赋值语句创建集合

```
>>> set1={1,2,3,4,3,2,1}           #创建集合对象时，重复元素自动去掉
>>> set1
{1, 2, 3, 4}
>>> set2={(1,1),(1,2),(2,1),(2,2)}
>>> set2
{(1, 2), (1, 1), (2, 1), (2, 2)}
```

2. 通过set()函数创建可变集合

>>> set3=set() #赋值语句set3={}创建的是空字典，空集只能
通过没有参数的set()函数创建。

>>> set3

set()

>>> set4=set([1,2,3,4,3,2,1])

#将列表对象创建为集合

>>> set4

{1, 2, 3, 4}

>>> set5=set((1,2,3,4,3,2,1))

#将元组对象创建为集合

>>> set5

{1, 2, 3, 4}

3. 通过frozenset()创建不可变集合

```
>>> set6=frozenset([1,2,3,4,3,2,1])
```

创建后不可更新

```
>>> set6
```

```
frozenset({1, 2, 3, 4})
```

```
>>>
```

#不可变集合

5.5.2 访问字典

- 集合中的元素是无序的，而且也没有任何的键与集合元素对应，所以无法取指定的某个元素，只能遍历整个集合访问其中的元素。例如：

```
>>> set1={1,7,2,6,5,4,3,7,1}
```

```
>>> for i in set1:
```

```
    print(i,end=',')
```

```
1,2,3,4,5,6,7,
```

5.5.3 更新集合

1. 添加元素

通过`add()`函数可以向集合中添加1个元素；通过`update()`函数可以向集合中添加多个元素。在向集合中添加元素时，如果新元素与集合中原有的元素重复，将不会被添加。

2. 删除元素

删除一个集合中的元素可以使用`remove()`函数，`pop()`函数，`discard()`函数或者`clear()`函数。

5.5.4 集合常用的其他操作

操作	功能
<code>x in set1</code>	元素x在set1中, 返回True, 否则返回False。
<code>x not in set1</code>	元素x不在set1中, 返回True, 否则返回False。
<code>set1==set2</code>	set1与set2元素个数, 元素值完全相同, 返回True, 否则返回False。
<code>set1!=set2</code>	set1与set2不同, 返回True, 否则返回False。
<code>set1<set2</code>	set1是set2的真子集, 返回True, 否则返回False
<code>set1<=set2</code>	set1是set2的子集, 返回True, 否则返回False
<code>set1 & set2</code>	求set1与set2的交集
<code>set1 set2</code>	求set1与set2的并集
<code>set1-set2</code>	求set1减set2, 即求属于set1不属于set2的元素集合
<code>set1^set2</code>	求set1和set2的对称差
<code>len(set1)</code>	求set1的元素个数

谢谢观看