

第 5 章

字典和集合

前面章节讲到的列表、元组和字符串都属于序列类型,序列的特点是数据元素之间有先后的顺序关系,通过位置编号(索引)来访问数据元素。本章介绍的字典和集合中的数据元素之间没有确定的顺序关系,属于无序的数据集合体,与序列类型的操作方式有很大的区别。

本章介绍字典与集合的概念、操作以及相关的应用。

5.1 字典

字典(dict)是以“{ }”为界限符,以“,”分隔的键值对的集合。字典是无序键值对的集合。每个键值对都包含两部分:键(key)和值(value)。键相当于索引,它对应的值就是数据,数据是根据键存储的,这种对应关系是唯一的。字典内的键必须是唯一的,值可以相同。

字典中的键必须是不可变的数据类型,如整数、实数、字符串、元组等。不允许使用列表、集合、字典作为字典的键,因为这些类型的数据是可变的。值可以是任意数据类型。

字典与序列类型的区别:

- (1) 存取和访问数据的方式不同。序列类型通过编号存取数据,而字典是根据键存取。
- (2) 序列类型是有序的数据集合,字典是无序的数据集合。字典中的键值对没有明确的顺序。
- (3) 字典是可变类型,序列类型是不可变类型。

5.1.1 字典的常用操作

1. 创建字典

Python 提供了多种创建字典的方法。

(1) 创建字典并赋值

使用赋值运算符“=”将一个字典赋值给一个变量即可创建一个字典变量,字典用大括号将字典的键值对括起来,每个键值对之间用逗号“,”分隔,对应的键和值之间用冒号“:”分隔。其格式为:

```
字典名 = {[键 1:值 1[, 键 2:值 2, ..., 键 n:值 n]]}
```

字典中的键在字典中必须是唯一的,而值可以不唯一。当键值对都省略时生产一个空字典。

【例 5-1】 创建字典、测试类型并显示字典内容。

参考代码如下：

```
>>> dict1 = {}
>>> type(dict1)
<class 'dict'>
>>> dict2 = {'id': '001', 'sex': '男'}
>>> dict2
{'id': '001', 'sex': '男'}
```

(2) 用 dict() 函数创建

使用 dict() 函数创建字典共有三种方法，具体如下：

① dict() 函数创建空字典，格式为：dict()。

② 用列表或元组作为 dict() 函数的参数。

【例 5-2】 把列表[['a', 1], ['b', 2]] 和元组 (('c', 3), ('d', 4)) 转换为字典。

参考代码如下：

```
>>> dict4 = dict(['a', 1], ['b', 2])
>>> dict4
{'a': 1, 'b': 2}
>>> dict5 = dict (('c', 3), ('d', 4))
{'c': 3, 'd': 4}
```

③ 将数据按照“关键字=值”的形式作为参数传递给 dict() 函数。代码如下：

```
>>> dict6 = dict(id = '002', sex = '男')
>>> dict6
{'id': '002', 'sex': '男'}
```

另外，用 zip() 函数可以把两个列表或元组对应位置的元素作为一个键值对，生成一个字典，函数 zip() 的参数要求是多个序列，格式为 zip(序列 1, 序列 2·····)。这里用 zip() 函数生成字典，所以参数需给定两个序列数据(列表或元组)。

【例 5-3】 用列表['a', 'b', 'c', 'd'] 和列表[1, 2, 3, 4] 生成字典{'a': 1, 'b': 2, 'c': 3, 'd': 4}。

参考代码如下：

```
>>> list1 = ['a', 'b', 'c', 'd']
>>> list2 = [1, 2, 3, 4]
>>> dict = dict(zip(list1, list2))
>>> dict
{'a': 1, 'b': 2, 'c': 3, 'd': 4}
```

(3) 用函数 fromkeys(seq[, value]) 创建字典

fromkeys() 是字典的函数，用来创建一个新的字典，其中参数 seq 是可迭代对象，参数 value 表示新建字典中的值，如果 value 缺省，默认值为 None。该方法适用于创建一个所有

值都相等的字典。

【例 5-4】 用 fromkeys() 函数创建值相等的字典。

参考代码如下：

```
>>> dict8 = {}.fromkeys(('a', 'b', 'c'))
>>> dict8
{'a': None, 'b': None, 'c': None}
>>> dict9 = {}.fromkeys(('a', 'b', 'c'), 3)
>>> dict9
{'a': 3, 'b': 3, 'c': 3}
```

2. 字典的基本操作

(1) 字典的访问

用下标来访问字典中的元素，下标是字典中的“键”，格式为：字典名[键名]。

如果键在字典中，则返回该键对应的值，否则引发一个 KeyError 错误。举例如下：

【例 5-5】 输出字典{'name': '张三', 'id': '001', 'sex': '男'}中键'name'和'age'的值。

```
>>> D1 = {'name': '张三', 'id': '001', 'sex': '男'}
>>> D1['name']
'张三'
>>> D1['age']
Traceback (most recent call last):
  File "<pyshell# 23>", line 1, in <module>
    D1['age']
KeyError: 'age'
```

'name'对应的值正常输出，但字典中没有键'age'，所以访问'age'引出错误。

如果字典中包含列表，则字典中列表的访问参照例 5-6 所示。

【例 5-6】 访问字典{'name': '张三', 'id': '001', 'sex': '男', 'score': [78, 89, 95, 88]}中列表的第 2 个元素。

参考代码如下：

```
>>> D1 = {'name': '张三', 'id': '001', 'sex': '男', 'score': [78, 89, 95, 88]}
>>> D1['score']
[78, 89, 95, 88]
>>> D1['score'][1]
89
```

D1['score']访问到列表，D1['score'][1]访问列表的第 2 个元素。字典也可以嵌套元组，操作方式与字典嵌套列表相同。

(2) 字典的更新

字典的更新指的是更新字典中的值，不是键，字典中的键是不能修改的。更新字典值的语句格式为：

字典名[键名] = 值

若该键在字典中存在,该语句是对该键对应的值进行更新操作,如果该键不在字典中,则表示添加一个新的“键值对”,也就是字典中添加了一个新元素。

【例 5-7】 把字典{'name': '张三', 'id': '001', 'sex': '男'}中的'001'改为'002',把键值对“'age':18”添加到字典中。

参考代码如下:

```
>>> D1 = {'name': '张三', 'id': '001', 'sex': '男'}
>>> D1['id'] = '002'
>>> D1
{'name': '张三', 'id': '002', 'sex': '男'}
>>> D1['age'] = 18
>>> D1
{'name': '张三', 'id': '002', 'sex': '男', 'age': 18}
```

(3) 字典的判断

判断某些键是否存在于字典中,用 in 或 not in 来判断。

【例 5-8】 判断'name'、'001'、'男'是否在字典{'name': '张三', 'id': '001', 'sex': '男'}中。

```
>>> D1 = {'name': '张三', 'id': '001', 'sex': '男'}
>>> 'name' in D1
True
>>> '001' in D1
False
>>> 'name' not in D1
False
>>> '男' not in D1
True
```

注意: 判断某些值是否在字典中指的是判断这些值是否是字典中的键,而不是值。

(4) 字典的长度

字典的长度指的是字典中键值对的数量,用的是 len() 函数,函数的参数是字典,如 len(D1)。

(5) 字典的运算

字典中常用的运算只有“==”和“!=”,用来判断两个字典是否相等。

【例 5-9】 判断{'a':1, 'b':2} 和 d2={'b':2, 'a':1} 是否相等。

参考代码如下:

```
>>> d1 = {'a':1, 'b':2}
>>> d2 = {'b':2, 'a':1}
>>> d1 == d2
True
```

这个例子也可以说明字典中的键值对是无序的。

(6) 字典的删除

字典的删除用 del 命令实现,包括删除字典和字典中的元素。

删除字典中元素的命令为：

```
del 字典名[键名]
```

删除字典的命令为：

```
del 字典名
```

【例 5-10】 先删除{'name': '张三', 'id': '001', 'sex': '男'}中“'name': '张三'”键值对，再删除字典。

```
>>> D1 = {'name': '张三', 'id': '001', 'sex': '男'}
>>> del D1['name']
>>> D1
{'id': '001', 'sex': '男'}
>>> del D1
>>> D1
Traceback (most recent call last):
  File "<pyshell#77>", line 1, in <module>
    D1
NameError: name 'D1' is not defined
```

5.1.2 字典的常用方法

1. keys()、values()和 items()方法

命令格式分别为：

```
字典名.keys()
字典名.values()
字典名.items()
```

keys()方法返回字典中的所有键，values()返回字典中的所有值，items()返回字典中的所有键值对。

【例 5-11】 分别显示{'name': '张三', 'id': '001', 'sex': '男'}中所有的键、所有的值以及所有的键值对。

参考代码如下：

```
>>> D1 = {'name': '张三', 'id': '001', 'sex': '男'}
>>> D1.keys()
dict_keys(['name', 'id', 'sex'])
>>> D1.values()
dict_values(['张三', '001', '男'])
>>> D1.items()
dict_items([('name', '张三'), ('id', '001'), ('sex', '男')])
```

2. copy()和 clear()方法

命令格式分别为：

字典名.copy(),功能是对字典的复制。

字典名.clear(), 功能是删除字典中全部的键值对, 使之变成空字典。

copy()方法的功能是实现字典的复制。clear()方法的功能是删除字典中全部的键值对, 使之变成空字典。

3. get()方法

命令格式为:

字典名.get(key, default)

功能是返回字典中键 key 对应的值, 若 key 不存在, 则返回 default 值, default 的缺省值是 None。

```
>>> D1 = {'name': '张三', 'id': '001', 'sex': '男'}
>>> print(D1.get('sex'))
男
>>> print(D1.get('brithday', 'no message'))
no message
>>> print(D1.get('age'))
None
```

键 'sex' 在字典中存在, 返回的是 'sex' 对应的值 '男'。字典 D1 中没有 'brithday', default 值为 'no message', 所以返回 'no message'。'age' 同样在字典 D1 中不存在, 但 get() 方法中没有指定 default 值, 所以返回 default 的缺省值 None。get() 方法与字典名[键名]方法获取键值有所不同, 'age' 在字典中不存在, get() 方法返回的是 None, 而字典名[键名]会直接报错。

4. update()方法

命令格式为:

字典 1.update(字典 2)

该命令的功能是用字典 2 中的键值对更新字典 1 中的键值对。

【例 5-12】 现有字典 D1 = {'name': '张三', 'id': '001', 'sex': '男'}, D2 为空字典, 实现下列操作:

- 复制 D1 到 D2。
- 把 D1 中的 '001' 改为 '002', 用 D1 更新 D2。

参考代码如下:

```
>>> D1 = {'name': '张三', 'id': '001', 'sex': '男'}
>>> D2 = {}
>>> D2.update(D1)
>>> D2
{'name': '张三', 'id': '001', 'sex': '男'}
>>> D1['id'] = '002'
>>> D2.update(D1)
>>> D2
{'name': '张三', 'id': '002', 'sex': '男'}
```

5. setdefault()方法

命令格式为:

字典名.setdefault(key,[value])

功能是如果字典中存在 key,则返回 key 对应的值;若 key 不存在,则返回键值对 key:value,同时把 key:value 添加到字典中,value 的缺省值是 None。

【例 5-13】 已知集合{'name': '张三', 'id': '001', 'sex': '男'},用 setdefault()方法实现下列操作:

- 返回键'id'的对应值。
- 把键值对“'age':18”添加进字典中。
- 把'birthday'作为键添加到字典中。

参考代码如下:

```
>>> D1 = {'name': '张三', 'id': '001', 'sex': '男'}
>>> D1.setdefault('id')
'001'
>>> D1.setdefault('age', 18)
18
>>> D1
{'name': '张三', 'id': '001', 'sex': '男', 'age': 18}
>>> D1.setdefault('birthday')
>>> D1
{'name': '张三', 'id': '001', 'sex': '男', 'age': 18, 'birthday': None}
```

6. pop()方法

命令格式为:

字典名.pop(key,[value])

功能是若字典中存在 key,则返回 key 对应的值,同时将该键值对在字典中删除;若字典中不存在该 key,则返回 value 值。

【例 5-14】 用 pop()方法删除{'name': '张三', 'id': '001', 'sex': '男'}中以'name'为键的键值对,再删除'age'时显示“不存在”。

参考代码如下:

```
>>> D1 = {'name': '张三', 'id': '001', 'sex': '男'}
>>> D1.pop('name')
'张三'
>>> D1
{'id': '001', 'sex': '男'}
>>> D1.pop('age', '不存在')
'不存在'
```

7. popitem()方法

命令格式为:

字典名.popitem()

功能是删除字典中的键值对,返回键值对构成的元组。

```
>>> D1 = {'name': '张三', 'id': '001', 'sex': '男'}
>>> D1.popitem()
('sex', '男')
>>> D1
{'name': '张三', 'id': '001'}
>>> type(D1.popitem())
<class 'tuple'>
>>> D1
{'name': '张三'}
```

5.2 集 合

集合(set)是一组对象的组合,是一个不重复的、无序的数据集合体。类似数学中的集合,可以进行交、并、差等运算。Python 中集合包含两种类型:可变集合(set)和不可变集合(frozenset)。可变集合中的元素既可以添加也可以删除,可变集合中的元素必须是不可变的数据类型,所以可变集合中的元素可以是数值、字符串或元组,但不能是列表和字典。可变集合中不能包含可变集合,因为可变集合属于可变数据类型,不能作其他集合的元素,同理也不能作为字典的键。

一个集合中包含的元素可以是不同数据类型。

5.2.1 集合的创建

1. 集合的创建

集合的创建有两种方法:第一种方法是用一对大括号“{}”将多个用逗号“,”分隔的数据括起来。{}不能表示空集合,因为{}表示空字典。如果要生成空集合需要用到第二种方法。第二种创建集合的方法是用 set() 函数,该方法可以将字符串、列表、元组等类型的数据转换成对应的集合类型。

(1) 生成空集合,代码如下:

```
>>> set2 = set()
>>> set2
set()
>>> type(set2)
<class 'set'>
```

(2) 用“{}”直接生成集合,代码如下:

```
>>> set3 = {0, 1, 2, 3, 'a', 'b', (34, 56)}
>>> set3
{0, 1, 2, 3, 'b', 'a', (34, 56)}
```

注意: 集合中可以包含数值、字符串、元组等不可变类型的数据。

(3) 利用 set() 函数将字符串转换成集合。

参考代码如下：

```
>>> set4 = set('hello world')
>>> set4
{'l', 'w', ' ', 'e', 'd', 'r', 'h', 'o'}
```

注意：set() 函数在把字符串 'hello world' 转换为集合时会把重复的元素删除，这是一个非常重要的特性。set() 函数也可以把列表和元组转化为集合。

下面介绍用 set() 函数把列表转化为集合，再用 list() 函数把集合转为列表的过程。

【例 5-15】 去掉列表 [89, 78, 90, 65, 59, 88, 79, 90, 80, 89] 中的重复值，再返回去重之后的列表。

参考代码如下：

```
>>> list1 = [89, 78, 90, 65, 59, 88, 79, 90, 80, 89]
>>> set5 = set(list1)
>>> set5
{65, 78, 79, 80, 88, 89, 90, 59}
>>> list2 = list(set5)
>>> list2
[65, 78, 79, 80, 88, 89, 90, 59]
```

(4) 创建不可变集合

Python 中集合包含两种类型：可变集合(set)和不可变集合(frozenset)。上面介绍的是创建可变集合的方法。frozenset() 函数可以将元组、列表和字符串等类型数据转换成不可变集合，下面介绍用 frozenset() 函数创建不可变集合。

```
>>> set7 = frozenset('hello world')
>>> type(set7)
<class 'frozenset'>
>>> set7
frozenset({'l', 'w', ' ', 'e', 'd', 'r', 'h', 'o'})
>>> set8 = {1, 2, 'a', set7}
>>> set8
{1, 2, frozenset({'l', 'w', ' ', 'e', 'd', 'r', 'h', 'o'}), 'a'}
```

上面代码中利用 frozenset() 函数生成了一个不可变集合 set7，set7 可以作为可变集合 set8 中的一个元素。

5.2.2 集合的常用运算

Python 中的集合支持多种集合运算，很多运算和数学中的集合运算含义一样。

1. 专门的集合运算

表 5-1 列出了专门的集合运算，表中 A 与 B 均表示集合。

表 5-1 专门的集合运算

表达式	功 能	表达式	功 能
$A \& B$	A 与 B 的交集	$A - B$	A 与 B 的差集
$A B$	A 与 B 的并集	$A \wedge B$	A 与 B 的对称差集

【例 5-16】 利用集合运算符分别计算集合 {1,2,3,4,5} 和集合 {2,4,6,8} 的交集、并集、差集和对称差集。

参考代码如下：

```
>>> set9 = {1,2,3,4,5}
>>> set10 = {2,4,6,8}
>>> seta = set9&set10
>>> seta
{2, 4}
>>> setb = set9|set10
>>> setb
{1, 2, 3, 4, 5, 6, 8}
>>> setc = set9 - set10
>>> setc
{1, 3, 5}
>>> setd = set9^set10
>>> setd
{1, 3, 5, 6, 8}
```

2. 集合的比较运算

表 5-2 列出了集合的比较运算,表中 A 与 B 均表示集合,C 表示元素。

表 5-2 集合的比较运算

表 达 式	功 能
$A == B$	判断 A 与 B 是否相等
$A != B$	判断 A 与 B 是否不相等
$A < B$	判断 A 是否是 B 的真子集
$A <= B$	判断 A 是否是 B 的子集(包括非真子集)
$A > B$	判断 A 是否是 B 的真超集
$A >= B$	判断 A 是否是 B 的超集(包括非真超集)
$C \text{ in } A$	C 是否是 A 的成员
$C \text{ not in } A$	C 是否不是 A 的成员

(1) $A == B$

判断集合 A 和 B 是否相等。若相等返回 True,否则返回 False,下面例子也充分说明集合是无序的。

```
>>> A = {'a', 'b', 'c', 'd'}
>>> B = {'c', 'd', 'b', 'a'}
>>> A == B
True
```

(2) $A \neq B$

判断集合 A 和 B 是否不相等。如果集合 A 和 B 具有不同的元素,则返回 True,否则返回 False。

```
>>> A = {'a', 'b', 'c', 'd'}
>>> B = {'c', 'd', 'b', 'e'}
>>> A == B
False
>>> A != B
True
```

(3) $A < B$

判断集合 A 是否是 B 的真子集。如果 A 不等于 B,且 A 中的所有元素都是 B 的元素,则返回 True,否则返回 False。

```
>>> A = {'a', 'b', 'c', 'd'}
>>> B = {'c', 'd', 'b', 'a'}
>>> A < B
False
>>> B = {'c', 'd', 'b', 'a', 'e'}
>>> A < B
True
```

(4) $A \leq B$

判断集合 A 是否是 B 的子集。如果 A 中所有元素都是 B 的元素,则返回 True,否则返回 False。

```
>>> A = {'a', 'b', 'c', 'd'}
>>> B = {'c', 'd', 'b', 'a'}
>>> A <= B
True
>>> B = {'c', 'd', 'b', 'a', 'e'}
>>> A <= B
True
```

(5) $A > B$

判断集合 A 是否是 B 的真超集。如果 A 不等于 B,且 B 中所有元素都是 A 的元素,则返回 True,否则返回 False。

```
>>> A = {'a', 'b', 'c', 'd'}
>>> B = {'c', 'd', 'b', 'a'}
>>> A > B
False
>>> B = {'c', 'd', 'b'}
>>> A > B
True
```

(6) A >= B

判断集合 A 是否是 B 的超集。如果 B 中所有元素都是 A 的元素,则返回 True,否则返回 False。

```
>>> A = {'a', 'b', 'c', 'd'}
>>> B = {'c', 'd', 'b', 'a'}
>>> A >= B
True
>>> B = {'c', 'd', 'b'}
>>> A >= B
True
```

(7) C in A

判断 C 是否是集合 A 中的元素。如果 C 是集合 A 中的元素,则返回 True,否则返回 False。

```
>>> A = {'a', 'b', 'c', 'd'}
>>> 'a' in A
True
>>> 1 in A
False
```

(8) C not in A

判断 C 是否不是 A 中的元素。如果 C 不是集合 A 中的元素,则返回 True,否则返回 False。

```
>>> A = {'a', 'b', 'c', 'd'}
>>> 'a' not in A
False
>>> 1 not in A
True
```

5.2.3 集合的常用方法

Python 提供的方法分为两类：面向所有集合的方法和面向可变集合的方法。面向所有集合的方法与集合的基本操作类似。面向可变集合的方法,是会对原集合产生更新操作的一些方法。

1. 面向所有集合的方法

面向所有集合的常用方法如表 5-3 所示,表中 A 与 B 均表示集合。

表 5-3 面向所有集合的常用方法

表 达 式	功 能
len(A)	计算集合 A 中的元素个数
A.copy()	复制集合 A
A.issubset(B)	判断 A 是否是 B 的子集
A.issuperset(B)	判断 A 是否是 B 的超集
A.isdisjoint(B)	判断 A 与 B 是否没有共同元素

表 达 式	功 能
A.union(B)	返回 A 与 B 的并集
A.intersection(B)	返回 A 与 B 的交集
A.difference(B)	返回 A 与 B 的差集
A.symmetric_difference(B)	返回 A 与 B 的对称差集

【例 5-17】 现有集合{1,2,3,4}、{1,2,3}和{2,4,6,8},应用于表 5-3 中的表达式。
参考代码如下:

```
>>> A = {1, 2, 3, 4}
>>> B = {1, 2, 3}
>>> C = {2, 4, 6, 8}
>>> len(A)
4
>>> D = A.copy()
>>> D
{1, 2, 3, 4}
>>> B.issubset(A)
True
>>> B.issuperset(A)
False
>>> A.issuperset(B)
True
>>> A.isdisjoint(C)
False
>>> A.union(C)
{1, 2, 3, 4, 6, 8}
>>> A.intersection(C)
{2, 4}
>>> A.difference(C)
{1, 3}
>>> A.symmetric_difference(C)
{1, 3, 6, 8}
```

2. 面向可变集合的方法

面向可变集合的方法会修改原集合中的元素,所以不适合用于不可变集合。常见的面向可变集合的方法如表 5-4 所示,表中 A 与 B 均表示集合。

表 5-4 面向可变集合的方法

表 达 式	功 能
A.update(B)	把集合 A 修改为 A 与 B 的并集
A.intersection_update(B)	把集合 A 修改为 A 与 B 的交集
A.difference_update(B)	把集合 A 修改为 $A-B$
A.symmetric_difference_update(B)	把集合 A 修改为 $A \cup B - A \cap B$
A.add(X)	把对象 X 添加到集合 A 中

续表

表 达 式	功 能
A.discard(X)	把 A 中的对象 X 删除,若不存在,没有任何操作
A.remove(X)	把 A 中的对象 X 删除,若不存在,则产生 KeyError 异常
A.pop()	删除 A 中任意元素,返回该元素
A.clear()	清空 A

(1) update()方法

```
>>> A = {1,2,3,4}
>>> B = {2,4,6,8}
>>> A.update(B)
>>> A
{1, 2, 3, 4, 6, 8}
```

(2) intersetion_update()方法

```
>>> A = {1,2,3,4}
>>> B = {2,4,6,8}
>>> A.intersection_update(B)
>>> A
{2, 4}
```

(3) difference_update()方法

```
>>> A = {1,2,3,4}
>>> B = {2,4,6,8}
>>> A.difference_update(B)
>>> A
{1, 3}
```

(4) symmetric_difference_update()方法

```
>>> A = {1,2,3,4}
>>> B = {2,4,6,8}
>>> A.symmetric_difference_update(B)
>>> A
{1, 3, 6, 8}
```

(5) add()方法

```
>>> A = {1,2,3,4}
>>> A.add('python')
>>> A
{1, 2, 3, 4, 'python'}
```

(6) discard()方法

```
>>> A = {1, 2, 3, 4}
>>> A.discard(1)
>>> A
{2, 3, 4}
>>> A.discard('python')
>>> A
{2, 3, 4}
```

(7) remove()方法

```
>>> A = {1, 2, 3, 4}
>>> A.remove(1)
>>> A
{2, 3, 4}
>>> A.remove('python')
Traceback (most recent call last):
  File "<pyshell#186>", line 1, in <module>
    A.remove('python')
KeyError: 'python'
>>> A
{2, 3, 4}
```

(8) pop()方法

```
>>> A = {1, 2, 3, 4}
>>> A.pop()
1
>>> A
{2, 3, 4}
```

注意：该方法是删除集合中某元素。

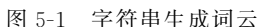
(9) clear()方法

```
>>> A = {1, 2, 3, 4}
>>> A.clear()
>>> A
set()
```

5.3 wordcloud 库

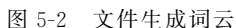
wordcloud 库是根据文本生成词云的 Python 第三方库。词云以词语为基本单位,根据其在文本中出现的频率设计大小不同的效果,从而能更加直观和艺术地展示文本。下面先看一段简单的代码:

运行代码会生成如图 5-1 所示的图片文件。



上面代码是给一个字符串生成词云，wordcloud 也可以把文件生成词云，只需把上面代码的第三行改为：`wordcloud.generate(open('A1.txt','r').read())`。其中 A1 是文本档名。完整代码如下：

生成词云图片如图 5-2 所示。



wordcloud 把词云作为一个对象,把文本中词语出现的频率作为参数绘制词云,词云的大小、颜色、形状都可以设定。设置方法是在实例化 wordcloud 对象时给定,例如想设定词云文件的背景色为白色,高度和宽度都为 300 像素,那么对应的代码为:

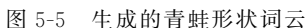


词云常用参数如表 5-5 所示。

参 数	功 能
font_path	指定字体文件的路径
width	文件宽度
height	文件高度
mask	词云形状,默认方形图
stopwords	排出词列表,即不显示在图中的词
background_color	背景色,默认为黑色

wordcloud 可以生成任意形状的词云,为获取形状,提供相应的图像,如图 5-4 所示,该图片保存在 D 盘根目录下,图片名称为“abc.jpg”,然后修改 mask 的值为 wctype。下面是生成青蛙形状的图云程序。由于需要对图像进行处理,所以需要导入 numpy 和 PIL 库。

生成的词云如图 5-5 所示。



【题目 1】 新建字典 dict1,要求字典内有 26 个键值对,每个键是一个大写英文字母,所有的值均为 None。

【题目 2】 现有字典 dict2={'姓名': '李伟', '性别': '男', '身高': 178, '体重': 75}, 编写循环程序输出: (1)dict2 中所有键。(2)dict2 中所有值。(3)dict2 中所有键值对。

【题目 3】 自定义一个非空字典,用户通过键盘输入数据,判断输入的数据是否在字典中,若在,则删除对应的键值对,输出“已删除”;否则,输出“不存在”。

【题目 4】 通过键盘输入一个由任意字符组成的字符串,利用字典编写程序统计输入的字符串中每个字母的个数。

【题目5】 已知有两个集合 setA 和 setB,分别存储了选择选修课 A 和选修课 B 的学生姓名,请自行创建这两个集合。计算并输出:

- (1) 同时选择了 setA 和 setB 两门选修课的学生姓名和人数。
- (2) 仅选择了选修课 A 而没有选择选修课 B 的学生姓名和人数。

(3) 仅选择了一门选修课的学生姓名和人数。

【题目 6】 通过键盘输入一个由任意字符组成的字符串,利用集合去重统计输入的字符串中字符种类的个数。

【题目 7】 上网下载英文文档,利用 wordcloud 生成词云图片文件。要求:

- (1) 图片背景为白色。
- (2) 图片尺寸为 400 像素×400 像素。
- (3) 图片形状为任意卡通人物形状。

习 题

【选择题】

1. 下面选项中能正确地创建字典的是()。
A. dict={}.fromkeys((1,2)) B. dict={a:1,b:2,c:3}
C. dict={{1,2}:1} D. dict=dict([1,2],[3,4])
2. 下列数据类型中,不能作为字典的键的是()。
A. 整数 B. 字符串 C. 列表 D. 元组
3. 设 A 和 B 为两个集合,下面选项中一定不是进行复制集合操作的是()。
A. A=B B. A=B.copy()
C. A.update(B) D. A=B.get()
4. 下面关于字典的描述中错误的选项是()。
A. 键值对中键必须是不可变的
B. 一个字典中所有键值对的键必须是唯一的
C. 字典中可以存储任意类型的数据
D. 一个字典中所有键值对的值不允许重复
5. 下列语句执行的结果是()。

```
1 a = {}.fromkeys("hello world!")
2 a.pop("o")
3 len(a)
```

- A. 8 B. 9 C. 10 D. 11

6. 下面程序的运行结果是()。

```
1 A1 = {}
2 A1[1] = 1
3 A1[3] = 2
4 A1[3] += 2
5 print(A1[3])
```

- A. 1 B. 3 C. 4 D. 5
7. 下列命令中可以删除字典中指定键值对的是()。
A. pop() B. popitem() C. clear() D. discard()

8. 集合 $A = \{1, 3, 5, 7, 9\}$, 集合 $B = \{1, 2, 3, 4, 5\}$, $A.symmetric_difference(B)$ 的返回值是()。

- A. $\{1, 2, 3, 4, 5, 7, 9\}$ B. $\{7, 9\}$
C. $\{1, 3, 5\}$ D. $\{2, 4, 7, 9\}$

9. 下面程序的运行结果是()。

```
1 A = set("aabbcbdeff")
2 A.discard(a)
3 len(A)
```

- A. 6 B. 5 C. 8 D. 9

10. 下列方法中可以应用于不可变集合的是()。

- A. add() B. discard() C. copy() D. remove()

【判断题】

1. 不通过键可以直接访问某一个键值对中的值。()
2. 字典中允许嵌套集合。()
3. 集合可以作为字典的键。()
4. 可以修改字典中已有的键。()
5. 一个字典中的键是不允许重复的,但值可以重复。()
6. 一个字典中可以有二个相同的键值对。()
7. 集合中不能包含集合。()
8. $\{'a', 'b', 'c', 'd'\}$ 和 $\{'c', 'a', 'd', 'b'\}$ 相等。()
9. 一个集合中元素的数据类型必须是一致的。()
10. 集合中不允许出现相同的数。()
11. 集合的 pop() 方法可以删除指定元素。()

【填空题】

1. Python 中字典以_____为界限符,字典内的每个元素的两部分之间用_____连接。
2. 集合分为_____和_____两种类型,其中可以用 set() 函数创建的是_____。
3. 已知字典 $D1 = \{'a': 1, 'b': 2, 'c': 3\}$, 把键 'c' 对应的值 3 改为 4 的命令是_____。
4. 已知字典 $D1 = \{'a': 1, 'b': 2, 'c': 3\}$, 显示出 D1 中所有键值对用的命令是_____。
5. 已知 D1 和 D2 是两个集合,用 D2 中的键值对更新 D1 的命令是_____。
6. 下面程序的运行结果是_____。

```
1 a1 = {}.fromkeys("123456789", 1)
2 sum = 0
3 for k in a1:
4     sum += a1[k]
5 print(sum)
```

7. 创建一个空集合且赋值给变量 set1 的命令是_____。
8. 有两个元组, $tup1 = ('name', 'id', 'age')$, $tup2 = ('张三', '001', 19)$, 请用一条语句将

这两个元组转换为一个字典。生成的字典为{'name': '张龙', 'id': '001', 'age': 19}, 这条语句为_____。

9. 下面程序的运行结果是_____。

```
1 list1 = {1,2,3,4}
2 list1.discard('1')
3 print(list1)
```

10. 若 $A = \{1, 3, 5, 7, 9\}$, $B = \{1, 2, 3, 4, 5\}$, 则 $A.intersection(B) =$ _____, $A.difference(B) =$ _____。

【简答题】

1. 什么是空字典和空集合? 怎么创建?
2. 集合有哪两种类型? 如何创建?
3. 试述删除字典和集合中元素的方法, 并比较相同点和不同点。
4. 试述字典与集合这两种类型的数据与列表、元组和字符串的区别。
5. 集合中数据不允许重复, 可以利用这一特性进行去重操作, 试述如何利用集合把字符串中数据进行去重。