第3章 序列数据

南通职业大学 朱亚林

内容提要

Python中的序列数据，是用一组连续的内存空间来存放多个值，几乎所有的程序设计语言都提供了类似的数据结构。主要包含列表、元组、字典和集合等形式。

1. 列表

- 列表是一种可变的、有序的数据结构，可以随时添加和删除其中的元素。
- 列表中元素的类型可以各不相同，所有元素放在一对中括号[ ]中，元素间用逗号分隔，如：[1，2，3，’abc’，4]。

- 列表的基本操作
- 列表的方法
- 列表应用

1. 列表——列表的基本操作

- 创建列表：可以用列表常量或list()、range()等函数来创建列表

```
>>> list1 = [1,2,3]
>>> list2 = list(('A','BC',3))
>>> list3 = list("中国欢迎您！ ")
>>> list4 = list(range(1,4))
```

  - 求表长：len(list1)
  - 列表更新：可以通过赋值的方式更新元素的值
  - 删除元素或列表：del list1[0]

1. 列表——列表的基本操作

- 列表合并：可以用加法运算实现两个列表合并

```
>>> list1=[1,2]
>>> list2=[3,4]
>>> list3=list1+list2
[1, 2, 3, 4]
```

- 列表乘法：可以利用乘法运算创建具有重复值的列表

```
>>> list1=[1,2]
>>> list2=list1*3
[1, 2, 1, 2, 1, 2]
```

1. 列表——列表的基本操作

- 列表分片：可以通过**指定范围索引**来对列表进行分片取值

> 分片就是从原有的列表中切分一部分下来，但是原来的列表保持不变

```
>>> list1=[1,2,['A','B'],3,4,5]
>>> list2=list1[1:3]    #将表list1的第2~3个元素取出
 [2, ['A', 'B']]
```

1. 列表——列表的方法

- 检索元素：list.index(value [,start[,end]])，其中start和end用于限定搜索范围,返回值为对应的索引。

```
>>> list1=[1,3,2,4,6,5]
>>> list1.index(4,2,6)
3
```

- 统计元素出现的次数： list.count(元素)

```
>>> list1.count(2)
1
```

1. 列表——列表的方法

- 在表尾添加新的元素：list.append()方法

```
>>> list1=[1,3,2,4,6,5]
>>> list1.append(8)
[1, 3, 2, 4, 6, 5, 8]
```

1. 列表——列表的方法

- 在表中插入新的元素： list.insert (index,obj)方法, obj元素值

```
>>> list1=[1,2,['A','B'],3,4,5]
>>> list1.insert(2,6)   #在表 list1的索引2处插入新元素6
 [1, 2, 6, ['A', 'B'], 3, 4, 5]
```

1. 列表——列表的方法

- 合并两个列表：list1.extend (list2)，新列表list2 连接到列表list1后

```
>>> list=[1,2,3]
>>> list2=[4,5,"abc"]
>>> list1.extend(list2)
 [1, 2, 3, 4, 5, 'abc']
```

1. 列表——列表的方法

- 移除并返回元素：list.pop ([index])

```
>>> list=[1,2,3]
>>> list.append(list.pop(0))  #移除列表中第一个元素，并作为新元素添加到列表表尾
>>> list
[2, 3, 1]
```

- 移除第一个被匹配元素：list.remove (obj)

```
>>> list1=[1,2,3,2,4,5]
>>> list1.remove(2)
>>> list1
[1, 3, 2, 4, 5]
```

1. 列表——列表的方法

- 列表逆置：list.reverse ()

```
>>> list1=[1,2,3,4,5]
>>> list1.reverse()
[5, 4, 3, 2, 1]
```

- 列表排序：list.sort ([reverse=True])，默认为升序，参数reserve=True时为降序。

```
>>> list=[2,1,5,3,4]
>>> list.sort()
[1, 2, 3, 4, 5]
>>> list.sort(reverse=True)
[5, 4, 3, 2, 1]
```

1. 列表——列表的方法

- 另一种排序：若采用内置函数sorted()排序，则不改变原列表的次序，可以将排序结果赋值给新表。语法格式为： sorted(list)，如：

```
>>> list1=[2,1,5,3,4]
>>> list2=sorted(list1)
>>> list2
[1, 2, 3, 4, 5]
>>> list1
[2, 1, 5, 3, 4]
```

1.列表——列表的方法

- 清空列表：可以使用clear()方法清空列表中的元素

```
>>> list.clear()
>>> list
[]
```

2. 元组

元组可看成元素不可变的列表，元组常量用圆括号表示，如（1，2，3）。

2. 元组——元组的基本操作

元组创建：用赋值常量或tuple()函数来创建元组

```
>>> a=(1,)  #创建只含1个元素的元组时，需以逗号结尾，多个元素时无需
(1,)
>>> b=tuple((1,2,3))
(1, 2, 3)
>>> print(tuple('Python'))
('P', 'y', 't', 'h', 'o', 'n')
```

2. 元组——元组的基本操作

- 元组读取：元组名[索引]。

```
>>> a=('C', 'Python','Java')
>>> a[1]
'Python'
>>> a[1][0]
'P'
```

- 元组删除：使用del命令删除整个元组对象

```
>>> del  b   #使用del命令删除整个元组对象b
```

2. 元组——元组的基本操作

- 元组切片：元组名[start:end] （不包括end）

```
>>> a=tuple('Python')
('P', 'y', 't', 'h', 'o', 'n')
>>> b=a[2:5]
('t', 'h', 'o')
```

- 求长度： len(元组)

```
>>> len((1,2,3))
3
```

2. 元组——元组的基本操作

- 元组合并：利用加法运算合并多个元组

```
>>> a=(1,2,3)
>>> b=(4,5,6)
>>> c=a+b
(1, 2, 3, 4, 5, 6)
```

- 成员判断：用in操作符判断对象是否属于元组

```
>>> a=(1,2,3)
>>> 2 in a
True
```

2. 元组——元组的方法

- 元素检索：元组. index(value,[,start[,end]])，检索首次出现的下标。

```
>>> a=(1, 2, 3, 1, 2, 3)
>>> a.index(2)
1
```

- 元素统计：元组. count(元素)

```
>>> a.count(3)
2
```

2.元组——列表与元组的区别

- 元组是一个不可变的序列，列表是可变序列；
- 在操作上有很多相似的地方，如索引、检索、切片、合并、重复、统计等，但元组没有append()、insert()、extend()、remove()、pop()等方法；
- 元组的速度比列表要快，若创建序列主要用于检索或类似用途，建议用元组；若可能涉及序列的修改，需用列表；
- 因为是不可变序列，元组可以作为字典的键，而列表不可以；
- 元组和列表可以相互转换:内置函数list()和tuple()

3.字典

- 字典是Python中唯一的映射类型，每个成员由“键：值”对的形式组成，所有成员由一对大括号“{”和“}”括起来，相邻成员用逗号间隔。定义形式为：

dict-name={key1:value1, key2:value2,..., keyn:valueN}

- 例：

student={"Zhang":20 , "Li":18 , "Wang":22 , "Zhao":19}

3. 字典——字典的基本操作

- 创建字典：通过赋值的方式创建例：

```
>>> dict1={}
>>> dict1["name"]="Tom"
>>> dict1["age"]=20
>>> dict1["address"]="maanshan"
>>> dict1
{'name': 'Tom‘ ,  'age': 20 ,  'address': 'maanshan'}
```

3. 字典——字典的基本操作

- 创建字典：通过内置函数dict()创建

```
>>> dict2=dict([(1,'a'),(2,'b'),(3,'c')])
>>> dict2
{1: 'a', 2: 'b', 3: 'c'}
>>> dict3=dict(a=1,b=2,c=3)
>>> dict3
{'a': 1, 'b': 2, 'c': 3}
```

3. 字典——字典的基本操作

- 创建字典：通过内建方法fromkeys() 创建

格式：dict.fromkeys(seq[, value])

```
>>> dict4={}.fromkeys (['name','age','addr'])  #给定了键的内容，但值的内容为空
{'name': None, 'age': None, 'addr': None}
>>> dict4['name']='John'
{'name': 'John', 'age': None, 'addr': None}
>>> dict5={}.fromkeys (['name','age','addr'],10)
{'name': 10, 'age': 10, 'addr': 10}
```

3. 字典——字典的基本操作

- 添加与修改:当以“键”为下标对字典元素赋值时，若该“键”不存在，则表示添加一个新元素；若该“键”存在，则表示修改该“键”的值。

```
>>> dict5={'语文':90, '数学':80, '英语':85}
{'语文': 90, '数学': 80, '英语': 85}
>>> dict5['政治']=78     #添加
{'语文': 90, '数学': 80, '英语': 85, '政治': 78}
>>> dict5['数学']=88     #修改
{'语文': 90, '数学': 88, '英语': 85, '政治': 78}
```

3. 字典——字典的基本操作

- 删除成员或字典:

```
>>> del dict5['作文']   #删除元素
>>> del dict5           #删除字典
```

- 字典遍历:利用循环语句和字典的items()方法。

  格式：字典名.items()。

```
for key,value in dict1.items():
    print("Key:%-10s Value:%-10s"%(key,value))
```

3. 字典

【例3-3】遍历并输出字典的所有键值对。

```
#exp3-3.py
dict1={'name': 'Tom','telephone':'13905551234','address': 'maanshan'}
for key,value in dict1.items():
        print("Key:%-10s Value:%-10s"%(key,value))
```

运行结果：

```
Key:name        Value:Tom
Key:telephone   Value:13905551234
Key:address     Value:maanshan
```

3. 字典

【例3-4】遍历并输出字典的所有键。

分析：利用字典的keys()方法：字典名.keys()。

```
#exp3-4.py
dict1={'name': 'Tom','telephone':'13905551234','address': 'maanshan'}
for key in dict1.keys():
        print("Key:%-10s "%key .title() ) # title()方法可以取出字符串的标题
```

运行结果：

```
Key:name
Key:telephone
Key:address
```

3．字典

【例3-5】输出字典的所有值。

分析：可以使用字典的内置方法values()来返回字典的所有值，不考虑重复的值。

```
#exp3-5.py
dict1={'name': 'Tom','telephone':'13905551234','address':
'maanshan'}
print("字典中所有的值为：",list(dict1.values()))
```

运行结果：

字典中所有的值为： ['Tom', '13905551234', 'maanshan']

3. 字典——字典的方法

- 获取指定键对应的值:

  字典名. get(key ,default=None)

```
>>> dict1={'name': 'Tom','telephone':'13905551234','address': 'maanshan'}
>>> dict1.get('address')
'maanshan'
>>> print(dict1.get('age'))  #键不存在时返回None
None
```

3. 字典——字典的方法

- 清空字典：使用clear()方法可以删除字典的所有元素，使其变成一个空字典。

```
>>> dict1={'A':1 ,'B':2,'C':3}
>>> dict1.clear()
>>> len(dict1)
0
```

3. 字典——字典的方法

- 字典复制：字典的copy()方法可以返回一个具有相同键值对的新字典，这种方式只拷贝父对象，而不拷贝内部的子对象，故称浅拷贝。

```
>>> dict1={'name': 'Tom','telephone':'13905551234','sex': ['男','女']}
>>> dict2=dict1                    #直接赋值就是字典的引用
>>> dict3=dict1.copy()             #浅拷贝
>>> dict1['name']='John'           #修改父类对象对dict3没有影响
>>> dict1['sex'].remove('女')      #删除子对象的值，对dict3有影响
>>> dict2,dict3
({'name': 'John', 'telephone': '13905551234', 'sex': ['男']}, {'name': 'Tom',
'telephone': '13905551234', 'sex': ['男']})
```

3. 字典——字典的方法

- 字典复制：深拷贝

  若需完全拷贝父对象及其子对象，则需引入copy模块实现深拷贝。

```
>>> import copy
>>> dict1={'a':[1,2,3]}
>>> dict2=copy.deepcopy(dict1)
>>> dict1['a'].append(4)
>>> dict1,dict2  #两个字典完全独立
({'a': [1, 2, 3, 4]}, {'a': [1, 2, 3]})
```

3．字典——字典的方法

- 字典更新:update()方法可以将一个字典的键值对更新到指定字典中,若没有相同项，则添加该项。

```
>>> dict1={'name': 'Tom','telephone':'13905551234','sex': ['男','女']}
>>> dict2={'name':'John'}
>>> dict1.update(dict2)   #更改
>>> dict1
{'name': 'John', 'telephone': '13905551234', 'sex': ['男', '女']}
>>> dict3={'age':20}
>>> dict1.update(dict3)   #添加
{'name': 'John', 'telephone': '13905551234', 'sex': ['男', '女'], 'age': 20}
```

4.集合

- 在 Python中集合是一个无序的不重复元素的序列。
- 集合有两种类型：可变集合和不可变集合。可变集合可以对集合中的元素进行添加和删除，而不可变集合则不能改变元素的值。

4. 集合

- 可变集合的创建：可通过一对花括号{ }将不同的元素括起来，元素间用逗号间隔。也可以通过内置函数set()来创建，此时函数set()的参数是一个列表。

```
>>> a_set={'Tom','Jack','Mary','John'}
>>> a_set
{'Mary', 'John', 'Tom', 'Jack'}
>>> b_set=set()  #创建空集合只能用set()函数，{}是创建空字典
set()
>>> c_set=set([1,2,3,1])  #用列表来构造
{1, 2, 3}              #有重复元素时，只接收一个
>>> d_set=set('ABCD')  #字符串作为参数，创建的是一个单字符多元素的集合
{'D', 'C', 'B', 'A'}
```

4.集合

集合的几个特点：

01

集合内的数据对象都是唯一的(不能重复多次的)

02

集合是无序的存储结构，集合中的数据没有先后顺序关系

03

集合内的元素必须是不可变对象

04

集合是可迭代对象

05

集合是相当于只有键没有值的字典(键则是集合的数据)

4. 集合

- 不可变集合的创建：只能用内置函数frozenset()实现。如：

```
>>> g_set=frozenset(['赵','钱','孙','李'])
#本集合不能增加和删除元素
>>> g_set
frozenset({'赵', '李', '钱', '孙'})
```

4.集合——集合的基本操作

- 集合的访问：不能通过索引的方式访问元素，只能通过遍历来访问所有元素

  for s in a_set：print(s)

- 可变集合的元素添加：通过add()方法实现添加一个元素，使用update()方法一次性添加多个元素。

```
>>> a_set={'Mary', 'John', 'Tom', 'Jack'}
>>> a_set.add('Alice')    #添加一个
>>> a_set
{'Tom', 'Alice', 'Jack', 'Mary', 'John'}
>>> b_set={'Jim','Rube'}
>>> a_set.update(b_set)  #添加多个
{'Tom', 'Jim', 'Alice', 'Jack', 'Mary', 'John', 'Rube'}
```

4.集合——集合的基本操作

- 可变集合的元素删除：可以通过remove()方法、pop()方法、discard()方法和clear()方法来实现元素的删除。其中pop()函数会删除集合中第一个元素并返回，clear()函数会删除集合中的所有元素；删除元素不存在时，remove()函数会抛出异常，而discard()函数不会抛出异常。

4.集合——集合的基本操作

- 可变集合的元素删除示例：

```
>>> a_set={'Tom', 'Jim', 'Alice', 'Jack', 'Mary', 'John', 'Rube'}
>>> a_set.remove('Alice')
{'Tom', 'Jim', 'Jack', 'Mary', 'John', 'Rube'}
>>> a_set.remove('Kite')  #抛出异常，
>>> a_set.discard('Kite')  #无任何提示
>>> a_set.pop()
'Tom'
>>> a_set
{'Jim', 'Jack', 'Mary', 'John', 'Rube'}
```

4.集合——集合的基本操作

- 集合的常规操作：支持并(|)、交(&)、差(-)、包含(in)等数学集合运算。

```
>>> a_set={1,2,3,5,7,8}
>>> b_set={2,4,6,9}
>>> a_set | b_set   #求两个集合的并集
{1, 2, 3, 4, 5, 6, 7, 8, 9}
>>> a_set & b_set  #求两个集合的交集
{2}
>>> a_set - b_set    #求两个集合的差集
{1, 3, 5, 7, 8}
>>> 3 in a_set     #判断指定元素是否包含于集合中
True
>>> len(a_set)     #求集合元素个数
6
```

4.集合——集合的基本操作

类型转换：通过内置函数list()和tuple()将集合分别转换成列表和元组。

```
>>> a_set={1,2,3,5,7,8}
>>> list(a_set)
[1, 2, 3, 5, 7, 8]
>>> tuple(a_set)
(1, 2, 3, 5, 7, 8)
```

小结

01

列表

介绍了列表的概念、基本操作和方法；

02

元组

介绍了元组的概念、基本操作和方法；

03

字典

介绍了字典的概念、基本操作和方法；

04

集合

介绍了集合的概念、基本操作和方法。

练习

1、下列语句,可用于创建列表的是____。

A. (1,2,3)

B. [1,2,3]

C. {1,2,3}

D. <1,2,3>

练习

2、关于列表，选项(　　)描述不正确。

A. 元素类型可以不同

B. 长度没有限制

C. 必须按顺序插入元素

D. 支持in运算符

练习

3、以下关于元组说法中，正确的是（　）。

A. 元组不支持切片操作

B. 所有元素数据类型必须相同

C. 插入的新元素放在最后

D. 支持in运算符

练习

4、元组和列表都有的方法是( )。

A. extend()

B. index()

C. append()

D. remove()

练习

5、以下代码的输出结果是（　）。

```
list1=['A','B','C']
list2=[10,20,30]
D=dict(zip(list1,list2))
print(D)
```

A．{'A': 10, 'B': 20, 'C': 30}

B．{10:'A', 20: 'B', 30:'C' }

C．抛出异常

D．不确定

练习

6、编写程序，设计一个字典，用户输入内容作为“键”，然后输出字典对应的“值”，若键不存在，则输出提示信息“键不存在”。

参考程序：

```
D={"name":"Zhangsan","sex":"M","address":"Nanjing","phone":"123456"}
Key = input("请输入一个键：")
if D.get(Key)!=None:
    print("值为：%s"%D.get(Key))
else:
    print("键不存在")
```

练习

7、已知字典

D={"name":"Zhangsan","sex":"M","address":"Nanjing","phone":"123456"}，请设计代码分别实现以下功能：

（1）输出字典D的所有键值对；

（2）输出D的phone值；

（3）修改D的"address"值为"Shanghai"；

（4）添加键值对“age”: 20；

（5）删除字典D的"sex"键值对。

参考代码如下页：

练习

参考代码如下：

```
D={"name":"Zhangsan","sex":"M","address":"Nanjing","phone":"123456"}
for key,value in D.items():
    print("Key:%-10s Value:%-10s"%(key,value))
print(D['phone'])
D['address']="Shanghai"
D['age']=20
del D['sex']
print(D)
```

练习

8．A和B是两个集合，对A&B的描述正确的是（　）。

A．A和B的并运算，包括两个集合的所有元素

B．A和B的差运算，包括集合A但不在集合B中的元素

C．A和B的交运算，包括同时在集合A和B中的元素

D．A和B的补运算，包括集合A和B中的不相同元素