Python语言程序设计

主讲人：袁宪锋

E-mail：yuanxianfeng@sdu.edu.cn
Phone：152-6312-1688
Office：知行北楼607

Python语言程序设计

第四章：序列的应用

Python语言程序设计

序列

[概念]

序列是一块用于存放多个值的连续内存空间，并且按一定顺序排列，可以通过索引取值。

Python中的序列主要有列表、元组、集合、字典和字符串，集合与字典不支持索引、切片、相加和相乘操作。

序列中常用的6种操作：

[概念]

序列每一个元素都有一个编号，也称之为索引，索引序号是从0开始递增的，也即下标为0代表第一个元素。

序列的正数索引

序列的负数索引

[概念]

切片是访问序列中元素的另一种方法，它可以**访问一定范围内的元素**，通过切片操作可以生成一个新的序列。

```
sname[start : end : step]
```

P81 说明

参数说明：

- sname：表示序列的名称。
- start：表示切片的开始位置（包括该位置），如果不指定，则默认为0。
- end：表示切片的截止位置（不包括该位置），如果不指定，则默认为序列的长度。
- step：表示切片的步长，如果省略，则默认为1，当省略该步长时，最后一个冒号也可以省略。

A[2:6]获取的是第几到第几个元素？

3-6,0开始

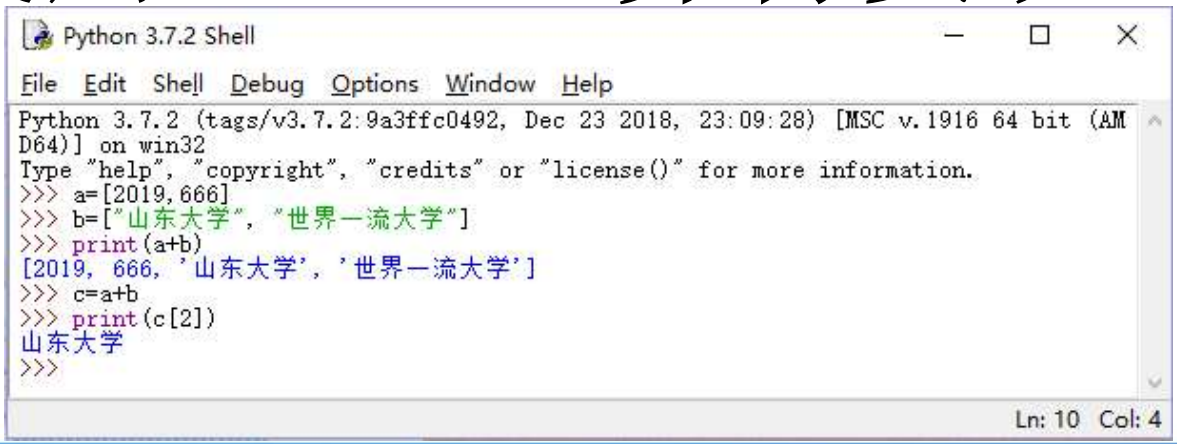
A[2:6:2]获取的是？

A[2], A[4]

[概念]

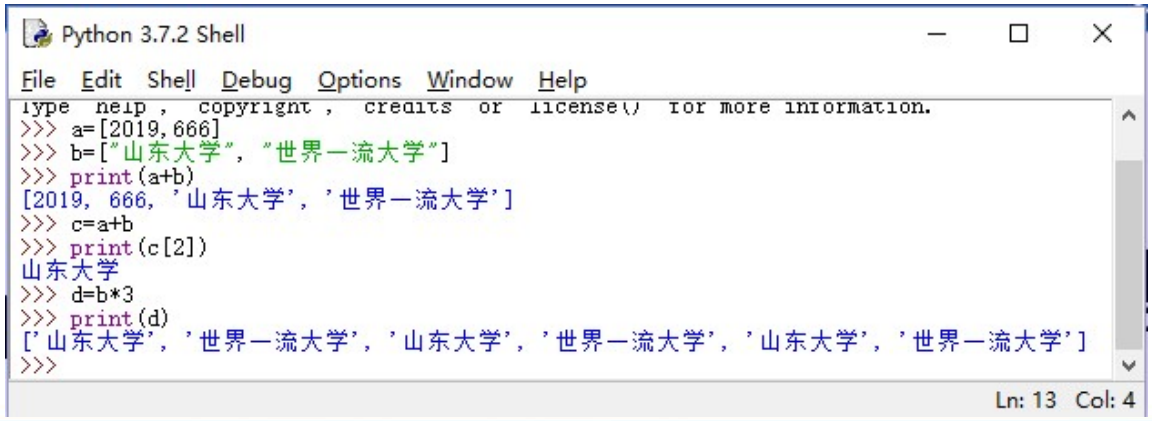
Python中支持两种**相同类型**的**序列**相加操作，即将两个序列进行连接，不会去除重复的元素，使用（+）运算符实现。

相同类型是指同为列表、元组、集合等，序列中的元素类型可以不同。例如：



[概念]

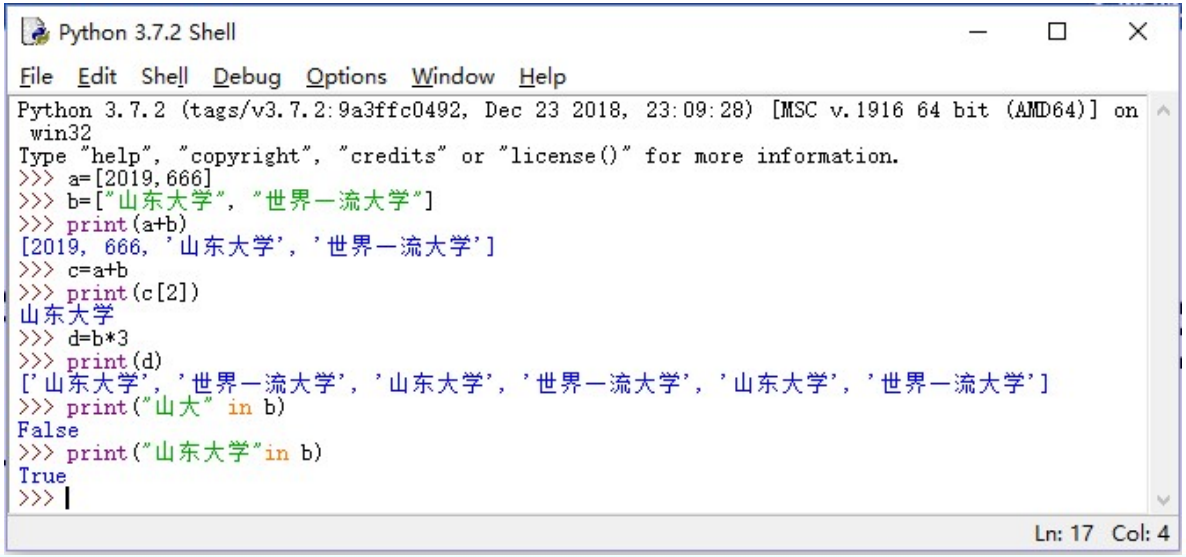
Python中使用数字n乘以一个序列会得到一个新的序列，其内容为**原序列被重复n次**的结果，例：



Python语言程序设计

[概念]

Python中使用**in**关键字查询某个元素是否为序列的成员：value **in** sequence



Python语言程序设计

Python 中使用**len()**函数计算序列长度，使用**max()**函数返回序列中最大元素，使用**min()**函数返回序列中的最小元素。Python提供的部分内置函数：

函　数	作　用
list()	将序列转换为列表
str()	将序列转换为字符串
sum()	计算元素和
sorted()	对元素进行排序
reversed()	反向序列中的元素
enumerate()	将序列组合为一个索引序列，多用在 for 循环中

P83

列表（list）

列表有序可变对象，int float string tuple 不可变

a=1　b=1　a is b ?
A=[1] b=[1] a is b?

Python语言程序设计

```
>>> x = 1
>>> id(x)
31106520
>>> y = 1
>>> id(y)
31106520
>>> x = 2
>>> id(x)
31106508
>>> y = 2
>>> id(y)
31106508
>>> z = y
>>> id(z)
31106508
```

python中的**不可变数据类型** int float string tuple，**不允许变量的值发生变化，如果改变了变量的值，相当于是新建了一个对象**，而对于**相同的值的对象，在内存中则只有一个对象(交互模式下tuple例外)**，内部会有一个引用计数来记录有多少个变量引用这个对象。

```
>>> a = [1, 2, 3]
>>> id(a)
41568816
>>> a = [1, 2, 3]
>>> id(a)
41575088
>>> a.append(4)
>>> id(a)
41575088
>>> a += [2]
>>> id(a)
41575088
>>> a
[1, 2, 3, 4, 2]
```

可变数据类型，允许变量的值发生变化，即如果对变量进行append、+=等这种操作后，**只是改变了变量的值，而不会新建一个对象，变量引用的对象的地址也不会变化**，不过对于**相同的值的不同对象，在内存中则会存在不同的对象**，即每个对象都有自己的地址，**相当于内存中对于同值的对象保存了多份**，这里不存在引用计数，是实实在在的对象。

Python中一切皆对象: 列表有序可变对象，int float string tuple 不可变

Python语言程序设计

1. 使用赋值运算符直接创建列表
2. 创建空列表
3. 创建数值列表

Python语言程序设计

Python中的列表是由一系列**按照特定顺序排列**的元素组成，是一种内置的**可变序列，形式上，所有元素都在一对[]中**，相邻元素用,隔开（**注意中英文符号**）。

内容上可以将整数、实数、字符串、列表、元组等任何类型放入列表中，**同一列表中元素类型可以不同**，这与其他语言不同。

#c语言中数组
int nums[]={1,2,3,4,5}

```
>>> list=["山东大学","世界一流大学",666,["努力奋斗",666]]
>>> print(list)
['山东大学', '世界一流大学', 666, ['努力奋斗', 666]]
>>> print(type(list))
<class 'list'>
>>> print(list[1:3])
['世界一流大学', 666]
>>> print(list[4])
Traceback (most recent call last):
  File "<pyshell#13>", line 1, in <module>
    print(list[4])
IndexError: list index out of range
>>> print(list[3])
['努力奋斗', 666]
```

List()可以将range对象、字符串、元组或其他可迭代类型的数据转换为列表。

```
>>> a=list(range(1,10,2))
>>> print(a)
[1, 3, 5, 7, 9]
>>> b="13579"
>>> c=list(b)
>>> print(c)
['1', '3', '5', '7', '9']
>>> d="山东大学"
>>> print(list(d))
['山', '东', '大', '学']
>>>
```

访问列表元素

Python语言程序设计

1. 直接使用print()函数输出
2. 索引
3. 切片

P86

```
>>> list=["山东大学","世界一流大学",666,["努力奋斗",666]]
>>> print(list)
['山东大学', '世界一流大学', 666, ['努力奋斗', 666]]
>>> print(type(list))
<class 'list'>
>>> print(list[1:3])
['世界一流大学', 666]
>>> print(list[4])
Traceback (most recent call last):
  File "<pyshell#13>", line 1, in <module>
    print(list[4])
IndexError: list index out of range
>>> print(list[3])
['努力奋斗', 666]
```

遍历列表

Python语言程序设计

1 直接使用for循环

```
for item in listname:
    #输出item
```

```
>>> a=["山东大学",985,211,"双一流"]
>>> for item in a:
        print(item)

山东大学
985
211
双一流
>>>
```

2 for循环和enumerate()函数

```
for index, item in enumerate(listname):
    #输出index, item
```

```
>>> a=["山东大学",985,211,"双一流"]
>>> for index, item in enumerate(a):
        print(index+1, item)

1 山东大学
2 985
3 211
4 双一流
>>>
```

添加、修改和删除列表元素
—更新列表

Python语言程序设计

列表添加元素：

1.**通过+连接序列（同类型）**

2.**通过列表对象的append()方法在列表尾部添加元素：** listname.append(obj)

3.**通过列表对象的extend()方法将一个列表添加至另一个列表中：**

Listname.extend(**seq**)

4.**通过insert()向列表指定位置插入元素，**

listname.insert(**index**, **obj**)

ch4.py - C:/Users/Raytine/Desktop/c...

File Edit Format Run Options Window Help

```
a=["山东大学",211]
b=[985,"双一流高校"]
print("a列表原内容是：", a)
print("b列表原内容是：", b)
print("示例1.a+b列表：", a+b)
a.append("世界一流大学")
print("示例2.a列表append函数增加元素后：", a)
a.extend(b)
print("示例3.a列表与b表合并extend函数：", a)
a.insert(1,"简称山大")
print("xf示例4.insert函数使用：", a)
```

Ln: 11 Col: 29

```
========== RESTART: C:/Users/Raytine/Desktop/ch4.py ==========
a列表原内容是： ['山东大学', 211]
b列表原内容是： [985, '双一流高校']
示例1.a+b列表： ['山东大学', 211, 985, '双一流高校']
示例2.a列表append函数增加元素后： ['山东大学', 211, '世界一流大学']
示例3.a列表与b表合并extend函数： ['山东大学', 211, '世界一流大学', 985, '双一流高校']
xf示例4.insert函数使用： ['山东大学', '简称山大', 211, '世界一流大学', 985, '双一流高校']
>>>
```

列表修改元素： **Ch4.py**
对应索引位置修改相应元素即可

Python语言程序设计

1 根据**索引**删除　　**del** listname[i]　　listname.**pop**(i)

2 根据**元素值**删除　　listname.**remove**(obj)

```
a=['山东大学', '简称山大', 211, '世界一流大学', 985, '双一流高校']
print("a列表原内容是：", a)
del a[1]
print("示例1.a列表删除第二个元素后：", a)
a.pop(-2)
print("示例2.a列表删除倒数第二个元素后：", a)
delvalue_1=211
if a.count(delvalue_1)>0:
    a.remove(delvalue_1)
print("示例3.a列表删除指定元素后：", a)
```

P92

```
================== RESTART: C:/Users/Raytine/Desktop/ch4.py ==================
a列表原内容是： ['山东大学', '简称山大', 211, '世界一流大学', 985, '双一流高校']
示例1.a列表删除第二个元素后： ['山东大学', 211, '世界一流大学', 985, '双一流高校']
示例2.a列表删除倒数第二个元素后： ['山东大学', 211, '世界一流大学', '双一流高校']
示例3.a列表删除指定元素后： ['山东大学', '世界一流大学', '双一流高校']
>>>
```

Ch4.py

Python语言程序设计

```
a = [1, 2, 3, 4]
for x in a:
    a.remove(x)
print(a)
print("=" * 20)
```

```
b = [1, 2, 3, 4]
for i in b:
    b.pop()
print(b)
```

```
c = [1, 2, 3, 4]
for i in range(len(c)):
    del c[0]
print(c)
```

投票 最多可选1项

设置

```
a = [1, 2, 3, 4]
for x in a:
    a.remove(x)
print(a)
```

Python 代码如上，选择正确的输出结果：

A [1,2]

B []

C [3,4]

D [2,4]

提交

投票 最多可选1项

设置

```
b = [1, 2, 3, 4]
for i in b:
    b.pop()
print(b)
```

Python 代码如上，选择正确的输出结果：

A []

B [1,2]

C [1,3]

D [2,4]

提交

投票 最多可选1项

设置

```
c = [1, 2, 3, 4]
for i in range(len(c)):
    del c[0]
print(c)
```

Python 代码如上，选择正确的输出结果：

A []

B [1,2]

C [1,3]

D [2,4]

Test.py

提交

对列表进行统计计算

Python语言程序设计

1.获取指定元素出现次数 listname.**count**(obj)

2.获取元素首次出现的下标 listname.**index**(obj)

3.统计数值列表元素和 **sum**(interable[,start])

```
a=list(range(1,10,2))
print(a)
print(a.count(2))
print(a.index(3))
print(sum(a))
```

```
''' 
========== RESTART: C:/Users/Raytine/Desktop/ch4.py ==========
[1, 3, 5, 7, 9]
0
1
25
>>>
```

Ch4.py

对列表进行排序

1 使用列表对象的sort()方法 Listname.sort(key=None,reverse=False)

```
grade = [98,99,97,100,100,96,94,89,95,100]          # 10名学生语文成绩列表
print("原列表：",grade)
grade.sort()                                        # 进行升序排列
print("升　序：",grade)
grade.sort(reverse=True)                            # 进行降序排列
print("降　序：",grade)
```

```
原列表：  [98, 99, 97, 100, 100, 96, 94, 89, 95, 100]
升　序：  [89, 94, 95, 96, 97, 98, 99, 100, 100, 100]
降　序：  [100, 100, 100, 99, 98, 97, 96, 95, 94, 89]
```

```
char = ['cat','Tom','Angela','pet']
char.sort()                                         # 默认区分字母大小写
print("区分字母大小写：",char)
char.sort(key=str.lower)                            # 不区分字母大小写
print("不区分字母大小写：",char)
```

```
区分字母大小写：  ['Angela', 'Tom', 'cat', 'pet']
不区分字母大小写：  ['Angela', 'cat', 'pet', 'Tom']
```

2 使用内置的sorted()函数

sorted(interable, key=None, reverse=False)

```
grade = [98,99,97,100,100,96,94,89,95,100]          # 10名学生语文成绩列表
grade_as = sorted(grade)                            # 进行升序排列
print("升序: ",grade_as)
grade_des = sorted(grade,reverse = True)            # 进行降序排列
print("降序: ",grade_des)
print("原序列: ",grade)
```

```
升序:  [89, 94, 95, 96, 97, 98, 99, 100, 100, 100]
降序:  [100, 100, 100, 99, 98, 97, 96, 95, 94, 89]
原序列:  [98, 99, 97, 100, 100, 96, 94, 89, 95, 100]
```

使用sort()**方法**时会改变原列表中元素的排列顺序，而使用sorted()**函数**时，会建立原列表的一个副本，并不改变原列表中元素顺序。

列表推导式

[概念]

列表推导式可以**快速生成**一个**列表**，或者根据某个列表**生成**满足指定需求的**列表**。

P96

Python语言程序设计

(1) **生成指定范围的数值列表：** list=[**Expression for var in range()**]

```
import random                          # 导入random标准库
randomnumber = [random.randint(10,100) for i in range(10)]
print("生成的随机数为：",randomnumber)
```

```
生成的随机数为：  [38, 12, 28, 26, 58, 67, 100, 41, 97, 15]
```

(2) **根据列表生成指定需求的列表：** list=[**Expression for var in list**]

```
price = [1200,5330,2988,6200,1998,8888]
sale = [int(x*0.5) for x in price]
print("原价格：",price)
print("打五折的价格：",sale)
```

```
原价格：  [1200, 5330, 2988, 6200, 1998, 8888]
打五折的价格：  [600, 2665, 1494, 3100, 999, 4444]
```

random.randint(10,100)
[10-100]
P96

(3) 从列表中选择符合条件的元素组成新的列表：

list=[**Expression for var in list if condition**]

```
price = [1200,5330,2988,6200,1998,8888
sale = [x for x in price if x>5000]
print("原列表：",price)
print("价格高于5000的：",sale)
```

```
原列表：  [1200, 5330, 2988, 6200, 1998, 8888]
价格高于5000的：  [5330, 6200, 8888]
```

Python语言程序设计

二维列表的使用

在Python中，二维列表是包含列表的列表，即一个列表的每一个元素又都是一个列表。

```
listname = [[元素11, 元素12, 元素13, …, 元素1n],
 [元素21, 元素22, 元素23, …, 元素2n],
 …,
 [元素n1, 元素n2, 元素n3, …, 元素nn]]
```

1. 直接定义二维列表

```
verse = [['千', '山', '鸟', '飞', '绝'], ['万', '径', '人', '踪', '灭'],
['孤', '舟', '蓑', '笠', '翁'], ['独', '钓', '寒', '江', '雪']]
```

Verse: 4行5列二维列表

2. For循环嵌套生成4x6的二维列表

```
xf=[] #创建空列表
for i in range(4):
    xf.append([]) #空列表中添加一个空列表元素
    for j in range(6):
        xf[i].append(j)
print(xf)
```

列表

列表

```
================== RESTART: C:\Users\Raytine\Desktop\ch4.py ==================
[[0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5]]
>>>
```

3. 列表推导式创建二维列表

```
xf=[[j for j in range(6)] for i in range(4)]
print(xf)
```

```
================== RESTART: C:\Users\Raytine\Desktop\ch4.py ==================
[[0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5]]
>>>
```

P99 **ch4.py**

Python语言程序设计

列表：有序可变对象 → 可变对象：内存地址不变情况下，对象值可变；不可变对象：

[xx,xx,xx] python

A B C

[A,B] C 数组

{ X=[1,2,3] list
  X=list(_) }

二、元素访问

① X=[1,2,3] (索引)
X[0]：1
X[2]：3
X[3]：X Error

② 切片
X[1:2]：[2,X]
s end end-1

三、遍历：

{ ① for item in listname
  ② for index, item in listname 元素索引值 }

四、列表元素添加

{ A=[1,2] B=[3,4]：A+B
  A.append(3) (字符)
  A.extend(B) seq A+B
  A.insert(index, obj) [-,-] }

五、删除元素

{ del
  list.pop(i) ③
  list.remove(ele) 啥 }

六、推导式

{ list=[Expression for var in range]
  list=[Expression for var in list]
  list=[exp for var in list if Cond] }

七、二维列表

A=[[1,2],[3,4,5]]

for 元组

不可变对象：数，字符，tuple

{ a=1 b=1 a is b? True
  a=[1,2] b=[1,2] a is b False }

元组（tuple）

与列表类似也是由一系列按特定顺序排列的元素组成，但是不可变序列（**数值、字符串、元组**），形式上元组中的所有元素均放在一对“**（ ）**”中，相邻的两个元素用“**,**”隔开，元素类型可以不同。**元组可用于保存程序中不可修改的内容。**

Test2 CODE

- **不可变对象：**对象存放在地址中的值不会被改变（所谓的改变是创建了一块新的地址并把新的对象的值放在新地址中原来的对象并没有发生变化）；Tupple, 数值，Str,

- **可变对象：**对象存放在地址中的值会原地改变,**对于相同的值的不同对象，在 内存中则会存在不同的对象**，即每个对象都有自己的地址，相当于内存中对于同值的对象保存了多份；

```
>>> a=1
>>> b=1
>>> a is b
```

True

```
>>> a=[1,2]
>>> b=[1,2]
>>> a is b
```

False

Python语言程序设计

```
>>> x = 1
>>> id(x)
31106520
>>> y = 1
>>> id(y)
31106520
>>> x = 2
>>> id(x)
31106508
>>> y = 2
>>> id(y)
31106508
>>> z = y
>>> id(z)
31106508
```

python中的**不可变数据类型** int float string tuple，**不允许变量的值发生变化，如果改变了变量的值，相当于是新建了一个对象**，而对于**相同的值的对象，在内存中则只有一个对象(交互模式下tuple例外)**，内部会有一个引用计数来记录有多少个变量引用这个对象。

```
>>> a = [1, 2, 3]
>>> id(a)
41568816
>>> a = [1, 2, 3]
>>> id(a)
41575088
>>> a.append(4)
>>> id(a)
41575088
>>> a += [2]
>>> id(a)
41575088
>>> a
[1, 2, 3, 4, 2]
```

可变数据类型，允许变量的值发生变化，即如果对变量进行append、+=等这种操作后，**只是改变了变量的值，而不会新建一个对象，变量引用的对象的地址也不会变化**，不过对于**相同的值的不同对象，在内存中则会存在不同的对象**，即每个对象都有自己的地址，**相当于内存中对于同值的对象保存了多份**，这里不存在引用计数，是实实在在的对象。

Python中一切皆对象: 列表有序可变对象，int float string tuple 不可变

单选题 1分

设置

此题未设置答案，请点击右侧设置按钮

```
>>> a=['JINAN','WEIHAI']
>>> b=['JINAN','WEIHAI']
>>> c=a
>>> a=a+['QINGDAO']
>>> print(a, b, c)
```

可变对象显式赋值 创建新对象

A ['JINAN', 'WEIHAI', 'QINGDAO'] ['JINAN', 'WEIHAI'] ['JINAN', 'WEIHAI', 'QINGDAO']

B ['JINAN', 'WEIHAI', 'QINGDAO'] ['JINAN', 'WEIHAI ', 'QINGDAO'] ['JINAN', 'WEIHAI', 'QINGDAO']

C ['JINAN', 'WEIHAI', 'QINGDAO'] ['JINAN', 'WEIHAI'] ['JINAN', 'WEIHAI']

提交

est2 CODE

投票 最多可选1项

设置

```
>>> a=['JINAN','WEIHAI']
>>> b=['JINAN','WEIHAI']
>>> c=a
>>> a+=['QINGDAO']
>>> print(a,b,c)
```

A ['JINAN', 'WEIHAI', 'QINGDAO'] ['JINAN', 'WEIHAI'] ['JINAN', 'WEIHAI', 'QINGDAO']

B ['JINAN', 'WEIHAI', 'QINGDAO'] ['JINAN', 'WEIHAI ', 'QINGDAO'] ['JINAN', 'WEIHAI', 'QINGDAO']

C ['JINAN', 'WEIHAI', 'QINGDAO'] ['JINAN', 'WEIHAI'] ['JINAN', 'WEIHAI']

Test2 CODE

投票 最多可选1项

设置

```
>>> a=['JINAN','WEIHAI']
>>> b=['JINAN','WEIHAI']
>>> c=a
>>> a+=['QINGDAO']
>>> print(a,b,c)
```

A ['JINAN', 'WEIHAI', 'QINGDAO'] ['JINAN', 'WEIHAI'] ['JINAN', 'WEIHAI', 'QINGDAO']

B ['JINAN', 'WEIHAI', 'QINGDAO'] ['JINAN', 'WEIHAI ', 'QINGDAO'] ['JINAN', 'WEIHAI', 'QINGDAO']

C ['JINAN', 'WEIHAI', 'QINGDAO'] ['JINAN', 'WEIHAI'] ['JINAN', 'WEIHAI']

Test2 CODE

投票 最多可选1项

设置

```
x=(1,2,3)
y=(1,2,3)
z=x

x=x+(4,5)
print(x,z)
```

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

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

Test2 CODE

提交

Python语言程序设计

Python语言程序设计

1 使用赋值运算符直接创建元组

Tuplename=(element1, element2,…,element n)

- 注意创建一个元素的元组时后面也要加，否则会出错。**例如：**

A=（“山东大学”），A实际为字符串；
A=（“山东大学”,），A为包含一个字符串元素的元组。

2 创建空元组 A=()

( )包括起来的对象 不一定非得是元组，有可能是字符串

3 创建数值元组

tuple(data)
Data是可以转换为元组的数据，其类型可以是range对象、字符串、元组或其他可迭代类型的数据。

P101-103

访问
元组元素

Python语言程序设计

1. 直接使用print()函数输出
2. 索引
3. 切片

```
#元组的访问
xf=("山东大学","自动化","测控",985,211,["双一流高校",666])
print(xf)
print(xf[0])
print(xf[3:])
```

```
================== RESTART: C:\Users\Raytine\Desktop\ch4.py ==================
('山东大学', '自动化', '测控', 985, 211, ['双一流高校', 666])
山东大学
(985, 211, ['双一流高校', 666])
>>>
```

ch4 CODE

Python语言程序设计

元组遍历：

for 循环遍历

```
xf=("清华","北京","浙江","上海交通","中山","山东")
print("好学校：", end=" ")
for univ in xf:
    print(univ+"大学", end=" ")
```

```
================== RESTART: C:\Users\Raytine\Desktop\ch4.py ==================
好学校：  清华大学 北京大学 浙江大学 上海交通大学 中山大学 山东大学
>>>
```

for循环配合emurate()函数遍历

```
xf=("清华","北京","浙江","上海交通","中山","山东")
print("好学校：", end=" ")
for index, univ in enumerate(xf):
    print("Top"+str(index+1), univ+"大学", end=" ")
```

```
>>>
================== RESTART: C:\Users\Raytine\Desktop\ch4.py ==================
好学校：  Top1 清华大学 Top2 北京大学 Top3 浙江大学 Top4 上海交通大学 Top5 中山大
学 Top6 山东大学
>>>
```

P105 Ch4.py

修改元组元素

Python语言程序设计

元组（不可变）不能单独修改某一元素，可以将两个元组连接；一个元素时注意后面加,

```
#元组元素修改
xf=("清华大学","北京大学","浙江大学","上海交通大学","中山大学")
y=("山东大学",)
xf_y=xf+y
print(xf_y)
xf_y[5]="双一流"
print(xf_y)
```

```
================== RESTART: C:\Users\Raytine\Desktop\ch4.py ==================
('清华大学', '北京大学', '浙江大学', '上海交通大学', '中山大学', '山东大学')
Traceback (most recent call last):
  File "C:\Users\Raytine\Desktop\ch4.py", line 66, in <module>
    xf_y[5]="双一流"
TypeError: 'tuple' object does not support item assignment
>>>
```

元组可以整体重新赋值实现对元素的修改

```
#元组元素修改
xf=("清华大学","北京大学","浙江大学","上海交通大学","中山大学")
y=("山东大学",)
xf_y=xf+y
print(xf_y)
#xf_y[5]="双一流"
xf_y=("清华大学","北京大学","浙江大学","上海交通大学","中山大学","双一流")
print(xf_y)
```

```
================== RESTART: C:\Users\Raytine\Desktop\ch4.py ==================
('清华大学', '北京大学', '浙江大学', '上海交通大学', '中山大学', '山东大学')
('清华大学', '北京大学', '浙江大学', '上海交通大学', '中山大学', '双一流')
>>>
```

Ch4.py

元组推导式

[概念]

元组推导式可以快速**生成**一个**元组**，或者根据某个元组**生成**满足指定需求的**元组**。

Python语言程序设计

回顾列表推导式3种形式：

（1）生成指定范围的数值列表：list=[Expression for var in range()]

```
import random                          # 导入random标准库
randomnumber = [random.randint(10,100) for i in range(10)]
print("生成的随机数为：",randomnumber)
```

```
生成的随机数为：  [38, 12, 28, 26, 58, 67, 100, 41, 97, 15]
```

（2）根据列表生成指定需求的列表：list=[Expression for var in list]

```
price = [1200,5330,2988,6200,1998,8888]
sale = [int(x*0.5) for x in price]
print("原价格：",price)
print("打五折的价格：",sale)
```

```
原价格：  [1200, 5330, 2988, 6200, 1998, 8888]
打五折的价格：  [600, 2665, 1494, 3100, 999, 4444]
```

（3）从列表中选择符合条件的元素组成新的列表：list=[Expression **for** var **in** list **if** condition]

```
price = [1200,5330,2988,6200,1998,8888
sale = [x for x in price if x>5000]
print("原列表：",price)
print("价格高于5000的：",sale)
```

```
原列表：  [1200, 5330, 2988, 6200, 1998, 8888]
价格高于5000的：  [5330, 6200, 8888]
```

元组推导式需要进行转换，使用**tuple**函数：

```
import random                                          # 导入random标准库
randomnumber = (random.randint(10,100) for i in range(10))
randomnumber = tuple(randomnumber)                     # 转换为元组
print("转换后：",randomnumber)
```

```
转换后： (76, 54, 74, 63, 61, 71, 53, 75, 61, 55)
```

```
number = (i for i in range(4))                 # 生成生成器对象
for i in number:                               # 遍历生成器对象
    print(i,end=" ")                           # 输出每个元素的值
print(tuple(number))                           # 转换为元组输出
```

```
0 1 2 3 ()
```

遍历后生成器即变空

元组与列表的区别

元组

不可变序列

（不能添加、修改和删除元素，可以整体替换）

支持切片的操作

（只能访问元组中的元素）

元组访问速度快

元组可以作为字典键

列表

可变序列

（随时添加、修改或者删除）

支持切片的操作

（可以访问、修改元组中的元素）

列表访问速度慢

不能作为字典的键

字典（dictionary）

Python中字典与列表类似，也是可变序列，不过与列表不同，字典是无序的可变序列，保存的内容是以“键Key-值Value”对的形式存放的。Key是唯一的，Value可以是多个。字典在定义包含多个命名字段的对象时，用途大。字典又称关联数组或散列表(Hash)。

特点：

（1）通过键key而不是索引来读取

（2）字典是任意对象的无序集合

（3）字典是可变的并且可以任意嵌套

（4）字典的键必须唯一

（5）字典的键key必须不可变，可以使用数字、字符串或元组但是不可是列表。

Python语言程序设计

主讲人：袁宪锋

E-mail：yuanxianfeng@sdu.edu.cn
Phone：152-6312-1688
Office：知行南楼机电学院305A

Python语言程序设计

Python语言程序设计

1. 创建字典，在键和值之间用冒号分隔，相邻两个元素用逗号分隔，所有元素放在一对{ }中，语法如下：

```
dictionary = {'key1':'value1', 'key2':'value2',  …, 'keyn':'valuen',}
```

- dictionary：表示字典名称。
- key1、key2…keyn：表示元素的键，必须是唯一的，并且不可变，例如，可以是字符串、数字或者元组。
- value1、value2…valuen：表示元素的值，可以是任何数据类型，不是必须唯一的。

创建一个字典存放控制学院和机电学院的地址

```
#字典创建

dic={'山大控制学院':'山东济南经十路17923号','山大机电与信息工程学院':'山东威海文化西路180号'}
print(dic)
```

```
>>>
 RESTART: C:\Users\xianfengyuan\OneDrive - sdu.edu.cn\10-课程\18-19-2-17自动化测控-Python\python材料\备课\ch4\ch4.py
{'山大控制学院': '山东济南经十路17923号', '山大机电与信息工程学院': '山东威海文化西路180号'}
>>>
```

2. 还可以使用dict()函数通过已有数据创建字典：

（1）通过映射函数创建字典

```
dictionary = dict(zip(list1,list2))
```

- ☑ dictionary：表示字典名称。
- ☑ zip() 函数：用于将多个列表或元组对应位置的元素组合为元组，并返回包含这些内容的 zip 对象。如果想获取元组，可以将 zip 对象使用 tuple() 函数转换这元组；如果想获取列表，则可以使用 list() 函数将其转换为列表。
- ☑ list1：一个列表，用于指定要生成字典的键。
- ☑ list2：一个列表，用于指定要生成字典的值。如果 list1 和 list2 的长度不同，则与最短的列表长度相同。

```
#字典创建

dic={'山大控制学院':'山东济南经十路17923号','山大机电与信息工程学院':'山东威海文化西路180号'}
print(dic)
name=['山东大学控制学院','山东大学机电学院']
address=['山东济南经十路17923号','山东威海文化西路180号']
dic2=dict(zip(name,address))
print(dic2)
print(list(zip(name,address)))
print(tuple(zip(name,address)))
```

```
 RESTART: C:\Users\xianfengyuan\OneDrive - sdu.edu.cn\10-课程\18-19-2-17自动化测控-Python\python材料\备课
{'山大控制学院': '山东济南经十路17923号', '山大机电与信息工程学院': '山东威海文化西路180号'}
{'山东大学控制学院': '山东济南经十路17923号', '山东大学机电学院': '山东威海文化西路180号'}
[('山东大学控制学院', '山东济南经十路17923号'), ('山东大学机电学院', '山东威海文化西路180号')]
(('山东大学控制学院', '山东济南经十路17923号'), ('山东大学机电学院', '山东威海文化西路180号'))
```

Python语言程序设计

3. 可以通过给定的“键-值”创建字典

Dic=**dict**(key1=value1,key2=value2,…,keyn=valuen)

```
#字典创建

dic={'山大控制学院':'山东济南经十路17923号','山大机电与信息工程学院':'山东威海文化西路180号'}
print(dic)
name=['山东大学控制学院','山东大学机电学院']
address=['山东济南经十路17923号','山东威海文化西路180号']
dic2=dict(zip(name,address))
print(dic2)
print(list(zip(name,address)))
print(tuple(zip(name,address)))
print()

dic=dict(是='yes',no='否', nine=9)
print(dic)
```

```
 RESTART: C:\Users\xianfengyuan\OneDrive - sdu.edu.cn\10-课程\18-19-2-17自动化测控-Python\pytho
{'山大控制学院': '山东济南经十路17923号', '山大机电与信息工程学院': '山东威海文化西路180号'}
{'山东大学控制学院': '山东济南经十路17923号', '山东大学机电学院': '山东威海文化西路180号'}
[('山东大学控制学院', '山东济南经十路17923号'), ('山东大学机电学院', '山东威海文化西路180号')]
(('山东大学控制学院', '山东济南经十路17923号'), ('山东大学机电学院', '山东威海文化西路180号'))

{'是': 'yes', 'no': '否', 'nine': 9}
```

4. 可以通过给定的元组和列表创建字典

{(a1,a2,a3):[b1,b2,b3]}

```
a=('济南','威海',985)
b=['控制学院','机电学院',211]
c={a:b}
print(c)
```

```
{('济南', '威海', 985): ['控制学院', '机电学院', 211]}
```

P112

Python语言程序设计

通过键值访问字典元素：

```
name=['山东大学控制学院','山东大学机电学院']
address=['山东济南经十路17923号','山东威海文化西路180号']
dic2=dict(zip(name,address))
print('学校的地址字典：', dic2)
print()
print('山大控制学院地址是:', dic2['山东大学控制学院'])
print()
print('山大机电学院地址是:', dic2['山东大学机电学院'] if '山东大学机电学院' in dic2 else '无此地址')
print('山大交通学院地址是：', dic2['山东大学机电学院'] if '山东大学交通学院' in dic2 else '无此地址')
```

```
 RESTART: C:\Users\xianfengyuan\OneDrive - sdu.edu.cn\10-课程\18-19-2-17自动化测控-Python\python材料\备课\ch4\ch4.py
学校的地址字典： {'山东大学控制学院': '山东济南经十路17923号', '山东大学机电学院': '山东威海文化西路180号'}

山大控制学院地址是: 山东济南经十路17923号

山大机电学院地址是: 山东威海文化西路180号
山大交通学院地址是：  无此地址
>>>
```

使用.get： dict.get(key, [ ])

```
print('使用.get获取山大机电学院地址:', dic2.get('山东大学机电学院','查无此地址'))
print('使用.get获取山大交通学院地址:',dic2.get('山大交通学院地址','sorry,查无此地'))
```

```
使用.get获取山大机电学院地址: 山东威海文化西路180号
使用.get获取山大交通学院地址: sorry,查无此地
>>>
```

通过items()方法实现对字典的遍历 dictionary.items()

```
name=['山东大学控制学院','山东大学机电学院']
address=['山东济南经十路17923号','山东威海文化西路180号']
dic2=dict(zip(name,address))

for item in dic2.items():
    print(item)
print()
for key,value in dic2.items():
    print(key,'的地址是：', value)
```

```
('山东大学控制学院', '山东济南经十路17923号')
('山东大学机电学院', '山东威海文化西路180号')

山东大学控制学院 的地址是： 山东济南经十路17923号
山东大学机电学院 的地址是： 山东威海文化西路180号
>>>
```

返回值可以是遍历（键-值对）的元组列表也可以是元组中的各元素。

P116 Ch4.py

4.4.4 添加、修改和删除字典元素

可以使用 dictionary[key]=value的形式实现对字典元素的修改和添加

```
name=['山东大学控制学院','山东大学机电学院']
address=['山东济南经十路17923号','山东威海文化西路180号']
dic2=dict(zip(name,address))

print('原字典：',dic2)
dic2['山东大学控制学院']='山东济南经十路73号'
print('修改控院地址后字典：',dic2)
dic2['山大机械学院']='济南历下经十路17923号'
print('增加机械学院地址：',dic2)
```

```
 RESTART: C:\Users\xianfengyuan\OneDrive - sdu.edu.cn\10-课程\18-19-2-17自动化测控-Python\python材料\备课\ch4\ch4.py
原字典： {'山东大学控制学院': '山东济南经十路17923号', '山东大学机电学院': '山东威海文化西路180号'}
修改控院地址后字典： {'山东大学控制学院': '山东济南经十路73号', '山东大学机电学院': '山东威海文化西路180号'}
增加机械学院地址： {'山东大学控制学院': '山东济南经十路73号', '山东大学机电学院': '山东威海文化西路180号', '山大机械学院': '济南历下经十路17923号'}
>>>
```

```
del dic2['山大机械学院']
print('删除机械学院地址：',dic2)
```

```
删除机械学院地址： {'山东大学控制学院': '山东济南经十路73号', '山东大学机电学院': '山东威海文化西路180号'}
>>>
```

Python语言程序设计

```
#字典推导式
import random
randomdic={i:random.randint(50,100) for i in range(2,6)}
print(randomdic)
```

```
 RESTART: C:\Users\xianfengyuan\OneDrive - sdu.edu.cn\                    Python\python材料\备课\ch4\ch4.py
{2: 82, 3: 52, 4: 84, 5: 97}
>>>
```

```
name=['山东大学控制学院','山东大学机电学院']
address=['山东济南经十路17923号','山东威海文化西路180号']
dic2=dict(zip(name,address))
print('dic2字典：',dic2)
dic3={i+'的地址是':j for i,j in zip(name,address)}
print('dic3字典:',dic3)
i,j=zip(name,address)
print(i,j)
```

```
 RESTART: C:\Users\xianfengyuan\OneDrive - sdu.edu.cn\10-课程\18-19-2-17自动化测控-Python\python材料\备课\ch4\ch4.py
dic2字典： {'山东大学控制学院': '山东济南经十路17923号', '山东大学机电学院': '山东威海文化西路180号'}
dic3字典: {'山东大学控制学院的地址是': '山东济南经十路17923号', '山东大学机电学院的地址是': '山东威海文化西路180号'}
('山东大学控制学院', '山东济南经十路17923号') ('山东大学机电学院', '山东威海文化西路180号')
>>>
```

https://blog.csdn.net/weixin_30462049/article/details/98004430?utm_medium=distribute.pc_relevant.none-task-blog-2~default~CTRLIST~default-1.no_search_link&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2~default~CTRLIST~default-1.no_search_link

集合（set）

[概念]

Python：用于保存不重复元素，最好的应用是去重。

形式上，集合所有元素都放在一对“{}”中，相邻两个元素使用，分隔。

```
a=[1,'1',2,3,2,3]
print(set(a))
```

```
{1, 2, 3, '1'}
>>>
```

可变集合
set

不可变集合
frozenset

Python语言程序设计

4.5.1集合的创建

1.直接使用{}创建集合：直接将集合赋值给变量实现创建集合

```
setname = {element 1,element 2,element 3,…,element n}
```

```
set1 = {'水瓶座','射手座','双鱼座','双子座'}
set2 = {3,1,4,1,5,9,2,6}
set3 = {'Python', 28, ('人生苦短', '我用Python')}
```

```
{'水瓶座', '双子座', '双鱼座', '射手座'}
{1, 2, 3, 4, 5, 6, 9}
{'Python', ('人生苦短', '我用Python'), 28}
```

Set集合是无序的，且元素无重复。

2.采用set()函数创建 setname = set(iteration)

```
set1 = set("命运给予我们的不是失望之酒，而是机会之杯。")
set2 = set([1.414,1.732,3.14159,2.236])
set3 = set(('人生苦短', '我用Python'))
```

注意：创建空集合的时候不能直接使用{}（这样创建的是空字典），推荐使用set()创建。

```
{'不', '的', '望', '是', '给', ',', '我', '。', '酒', '会', '杯', '运', '们', '予', '而', '失', '机', '命', '之'}
{1.414, 2.236, 3.14159, 1.732}
{'人生苦短', '我用Python'}
```

集合元素的添加

```
setname.add(element)
```

```
mr = set(['零基础学Java','零基础学Android','零基础学C语言','零基础学C#','零基础学PHP'])
mr.add('零基础学Python')                                    # 添加一个元素
print(mr)
```

```
{'零基础学PHP', '零基础学Android', '零基础学C#', '零基础学C语言', '零基础学Python', '零基础学Java'}
```

集合元素的删除

在 Python 中，可以使用 del 命令删除整个集合，也可以使用集合的 pop() 方法或者 remove() 方法删除一个元素，或者使用集合对象的 clear() 方法清空集合，即删除集合中的全部元素，使其变为空集合。

Python语言程序设计

```
mr = set(['零基础学Java','零基础学Android','零基础学C语言','零基础学C#','零基础学PHP','零基础学Python'])
mr.remove('零基础学Python')                    # 移除指定元素
print('使用remove()方法移除指定元素后：',mr)
mr.pop()                                       # 删除一个元素
print('使用pop()方法移除一个元素后：',mr)
mr.clear()                                     # 清空集合
print('使用clear()方法清空集合后：',mr)
```

```
使用remove()方法移除指定元素后： {'零基础学Android', '零基础学PHP', '零基础学C语言', '零基础学Java',
'零基础学C#'}
使用pop()方法移除一个元素后： {'零基础学PHP', '零基础学C语言', '零基础学Java', '零基础学C#'}
使用clear()方法清空集合后： set()
```

```
#集合的删除
a=set(['山东大学威海校区','山东大学青岛校区','山东大学千佛山校区'])
print('a原本集合为：',a )
a.remove('山东大学千佛山校区') if '山东大学千佛山校区'in a else print('不存在此项')
print('a删除最后一个元素后：',a )
a.remove('山东大学千佛山校区') if '山东大学千佛山校区'in a else print('不存在此项')
```

```
a原本集合为： {'山东大学青岛校区', '山东大学千佛山校区', '山东大学威海校区'}
a删除最后一个元素后： {'山东大学青岛校区', '山东大学威海校区'}
不存在此项
>>>
```

集合是无序的推荐 .remove()

Python语言程序设计

4.5.3 集合的交集“&”、并集“|”和差集“-”运算

```
py=set(['张三','李四','王二','小明'])
c=set(['小胡','小孙','小齐','小明','王二'])
print('选择python课的同学：',py)
print('选择c课程的同学：',c)
print('同时选择py和c的学生（交集）：',py&c)
print('全部选课学生名单（并集）：',py|c)
print('仅选择py的学生名单(差集)：',py-c)
```

```
...
 RESTART: F:\OneDrive - sdu.edu.cn\10-课程\18-19-2-17自动化测控-Python\python材
料\备课\ch4\ch4.py
选择python课的同学：  {'小明', '李四', '张三', '王二'}
选择c课程的同学：  {'小胡', '小孙', '小齐', '小明', '王二'}
同时选择py和c的学生（交集）：  {'小明', '王二'}
全部选课学生名单（并集）：  {'小胡', '小孙', '小齐', '张三', '小明', '李四', '王
二'}
仅选择py的学生名单(差集)：  {'李四', '张三'}
>>>
```

列表、元组、字典和集合的区别

Python语言程序设计

表 4.2　列表、元组、字典和集合的区别

数据结构	是否可变	是否重复	是否有序	定义符号
列表（list）	可变	可重复	有序	[]
元组（tuple）	不可变	可重复	有序	()
字典（dictionary）	可变	可重复	无序	{key:value}
集合（set）	可变	不可重复	无序	{ }

字典 键-值对，集合元素不可重复。

Python语言程序设计

列表简介

列表：用于存储任意数目、任意类型的数据集合。

列表是内置可变序列，是包含多个元素的有序连续的内存空间。列表定义的标准语法格式：

a = [10,20,30,40]其中，10,20,30,40 这些称为：列表 a 的元素。

列表中的元素可以各不相同，可以是任意类型。比如：a = [10,20,'abc',True]

方法	要点	描述
list.append(x)	增加元素	将元素 x 增加到列表 list 尾部
list.extend(aList)	增加元素	将列表 alist 所有元素加到列表 list 尾部
list.insert(index,x)	增加元素	在列表 list 指定位置 index 处插入元素 x
list.remove(x)	删除元素	在列表 list 中删除首次出现的指定元素 x
list.pop([index])	删除元素	删除并返回列表 list 指定为止 index 处的元素，默认是最后一个元素
list.clear()	删除所有元素	删除列表所有元素，并不是删除列表对象
list.index(x)	访问元素	返回第一个 x 的索引位置，若不存在 x 元素抛出异常
list.count(x)	计数	返回指定元素 x 在列表 list 中出现的次数
len(list)	列表长度	返回列表中包含元素的个数
list.reverse()	翻转列表	所有元素原地翻转
list.sort()	排序	所有元素原地排序
list.copy()	浅拷贝	返回列表对象的浅拷贝

Python语言程序设计

切片是 Python 序列及其重要的操作，适用于列表、元组、字符串等等。切片的格式如下：

切片 slice 操作可以让我们快速提取子列表或修改。标准格式为：[起始偏移量start ： 终止偏移 量 end [: 步 长 step]]

注：当步长省略时顺便可以省略第二个冒号

操作和说明	示例	结果
[:] 提取整个列表	[10,20,30][:]	[10,20,30]
[start:]从 start 索引开始到结尾	[10,20,30][1:]	[20,30]
[:end]从头开始知道 end-1	[10,20,30][:2]	[10,20]
[start:end]从 start 到 end-1	[10,20,30,40][1:3]	[20,30]
[start:end:step] 从 start 提取到 end-1，步长是 step	[10,20,30,40,50,60,70][1:6:2]	[20, 40, 60]

示例	说明	结果
[10,20,30,40,50,60,70][-3:]	倒数三个	[50,60,70]
10,20,30,40,50,60,70][-5:-3]	倒数第五个到倒数第三个(包头不包尾)	[30,40]
[10,20,30,40,50,60,70][::-1]	步长为负，从右到左反向提取	[70, 60, 50, 40, 30, 20, 10]

Python语言程序设计

元组元素不可修改，元组不可增加元素、修改元素、删除元素

1. 元组的核心特点是：不可变序列。
2. 元组的访问和处理速度比列表快。
3. 与整数和字符串一样，元组可以作为字典的键，列表则不能作为字典的键使用。

字典是“键值对”的无序可变序列，字典中的每个元素都是一个“键值对”，包含：“键对象”和“值对象”。

可以通过“键对象”实现快速获取、删除、更新对应的“值对象”。列表中我们通过“下标数字”找到对应的对象。字典中通过“键对象”找到对应的“值对象”。

“键”是任意的不可变对象，比如：整数、浮点数、字符串、元组。但是：列表、字典、集合这些可变对象，不能作为“键”。并且“键”不可重复。“值”可以是任意的数据，并且可重复。

Python语言程序设计

操作	解释
adict.keys()	返回一个包含字典所有KEY的列表；
adict.values()	返回一个包含字典所有value的列表；
adict.items()	返回一个包含所有（键，值）元祖的列表；
adict.clear()	删除字典中的所有项或元素；
adict.copy()	返回一个字典浅拷贝的副本；
adict.fromkeys(seq, val=None)	创建并返回一个新字典，以seq中的元素做该字典的键，val做该字典中所有键对应的初始值（默认为None）；
adict.get(key, default = None)	返回字典中key对应的值，若key不存在字典中，则返回default的值（default默认为None）；
adict.has_key(key)	如果key在字典中，返回True，否则返回False。 现在用 in 、 not in；
adict.iteritems() adict.iterkeys() adict.itervalues()	与它们对应的非迭代方法一样，不同的是它们返回一个迭代子，而不是一个列表；
adict.pop(key[,default])	和get方法相似。如果字典中存在key，删除并返回key对应的vuale；如果key不存在，且没有给出default的值，则引发keyerror异常；
adict.setdefault(key, default=None)	和set()方法相似，但如果字典中不存在Key键，由 adict[key] = default 为它赋值；
adict.update(bdict)	将字典bdict的键值对添加到字典adict中。

```
name=['山东大学控制学院','山东大学机电学院']
address=['山东济南经十路17923号','山东威海文化西路180号']
dic2=dict(zip(name,address))
print(dic2)
print(dic2.keys())
print(dic2.items())

for key in dic2.keys():
    print(key)
print()
for key,value in dic2.items():
    print(key,'的地址是：', value)
```

```
 RESTART: F:\OneDrive - sdu.edu.cn\10-课程\18-19-2-17自动化测控-Python\python材料\备课\ch4\ch4.py
{'山东大学控制学院': '山东济南经十路17923号', '山东大学机电学院': '山东威海文化西路180号'}
dict_keys(['山东大学控制学院', '山东大学机电学院'])
dict_items([('山东大学控制学院', '山东济南经十路17923号'), ('山东大学机电学院', '山东威海文化西路180号')])
山东大学控制学院
山东大学机电学院

山东大学控制学院 的地址是： 山东济南经十路17923号
山东大学机电学院 的地址是： 山东威海文化西路180号
>>>
```

列表推导式生成列表对象，语法如下：
[表达式 for item in 可迭代对象]或者：[表达式for item in 可迭代对象 if 条件判断]

字典的推导式生成字典对象，格式如下：
{key_expression : value_expression for 表达式 in 可迭代对象} 类似于列表推导式，字典推导也可以增加 if 条件判断、多个 for 循环。

集合推导式
集合推导式生成集合，和列表推导式的语法格式类似：
{表达式 for item in 可迭代对象 }或者：{表达式 for item in 可迭代对象 if 条件判断}

元组推导式
元组推导式注意使用tuple()转换

```
import random                                          # 导入random标准库
randomnumber = (random.randint(10,100) for i in range(10))
randomnumber = tuple(randomnumber)                     # 转换为元组
print("转换后：",randomnumber)
```

Python语言程序设计

1. 模拟某网站的注册过程：

step1: 要求输入用户名（必须为字母）和密码（必须包括数字和字母组合）

step2: 当用户输入符合规定的用户名和密码时提示用户“注册成功”，若用户输入的信息不符合规定，提示用户输入规则“用户名必须为字母，密码必须包括数字和字母”

step3: 提醒用户注册成功后，询问用户是否登录，若输入YES则进入step4,输入NO进入step5.

step4:实现用户输入用户名和密码，当用户名为 注册的名字且密码为注册密码时，显示“登陆成功”，否则显示“登陆失败，您还可以尝试x次”，失败时允许重复输入三次，每次失败时提醒剩余输入次数。

step5:退出