第3章
使用字符串、列表和元组

CS, ZJU
2018年12月

Overview

- 序列的访问及运算符
- 字符串使用
- 列表和元组使用

3.1 序列的访问及运算符

- 序列（ Sequence ）

为满足程序中复杂的数据表示，Python支持组合数据类型，可以将一批数据作为一个整体进行数据操作，这就是数据容器的概念。

容器中可包含多个数据（元素），容器中的数据（元素）有先后次序，每个元素通过用其下标（索引）来访问。序列的下标从0开始，后面下标依次为1, 2, 3, …. 。

序列是其中一大类数据容器的统称, 不是具体的数据类型。

- 常用的序列类型：列表（list）,字符串（string）,元组（tuple）
- range()函数产生的结果也是序列，各种序列操作对它也适用，用
- for语句可遍历range产生的对象
- for i in range(4):print(i,end=“”），不需要list(range(4))

通用的序列操作

所有的序列类型都可以进行的操作归纳如下表所示。

表3-1 序列的操作

操作	描述
X1+X2	联接序列X1和X2,生成新序列
X*n	序列X重复n次，生成新序列
X[i]	引用序列X中下标为i的成员
X[i: j]	引用序列X中下标为i到j-1的子序列
X[i: j: k]	引用序列X中下标为i到j-1的子序列，步长为k
len(X)	计算序列X中成员的个数
max(X)	序列X中的最大值
min(X)	序列X中的最小值
v in X	检查v是否在序列X中，返回布尔值
v not in X	检查v是否不在序列X中，返回布尔值

访问单个数据

- 用[ ]来访问序列中的一个元素。

比如访问字符串中的某个字符：

```
prompt = ‘hello’
print(prompt[0])
```

输出：

h

```
print(prompt[4])
```

输出：

o

访问单个数据（续）

- 假设序列中的元素个数是n，下标的有效范围是0到n-1，或者-1到-n。
- 如果下标的绝对值大于n-1，则会发生下标越界错误。
- 如果下标的值为负数，表示从序列的最后一个元素往前引用，比如：

```
prompt = ‘hello’
print(prompt[-1],prompt[-4])
```

输出

o e

访问一部分数据

- 如果要访问序列中的一部分元素，可以使用切片（slice）。切片通过冒号分隔两个下标来实现。比如访问列表中的一部分：

```
a = [2,3,5,7,11,13]
print(a[1:3])
```

输出

[3,5]

注：如右图所示，切片a[1:3] 表示包含从第1个下标（1）开始到第2个下标（3）前面的下标（2）为止的部分元素的子序列（列表）。

访问一部分数据（续）

- 切片使用负的下标访问

a[1:-3]

结果：[3,5]

- 切片省略第2个下标，表示从第1个下标的元素开始到最后一个元素的切片。

a[2:]

结果：[5, 7, 11, 13]

- 第1个下标为0时，可以省略。

a[:3]

结果：[2, 3, 5]

a[:-2]

结果：[2, 3, 5, 7]

访问一部分数据（续2）

- 切片使用第3个参数，该参数表示切片选择元素的步长。

a[0:5:2]

结果是: [2, 5, 11]

- 切片使用第3个参数为负数时，表示逆向取切片。

a[-1:0:-1]

结果是：[13, 11, 7, 5, 3]

a[::-1]

结果是：[13, 11, 7, 5, 3, 2]

复制一个序列

- 如果将一个序列变量赋值给另外一个变量，则这2个变量表达了同一个序列。

```
a = [2, 3, 5, 7, 11, 13]
b = a
b[0] = 1
print(a)
```

输出：

[1, 3, 5, 7, 11, 13]

- 如果希望2个变量各自拥有独立的序列，可使用切片。

```
a = [2, 3, 5, 7, 11, 13]
b = a[:]
```

注：a[:]表示从头到尾的整个序列“切”出来的序列

序列拆包运算

- 序列拆包是把一个序列的元素给多个变量赋值(引用）
- >>>info=["章武","male",30,(70,80,65,78)]
- >>>name,sex,age,grade=info
- >>>name
- “章武”
- >>>grade
- (70,80,65,78)
- 用下划线表示匿名变量，只获取需要的元素
- name,_,_,grade=info

- 字符串也是序列，也可以拆包

```
>>>s="ab"
>>> c,d=s
>>> c
'a'
>>> d
'b'
```


变量前加*，获取子序列

- >>>grade=(70,80,65,78)
- >>>a,*b=grade
- >>> a
- 70
- >>> b
- [80, 65, 78]

- >>>a,*b,c=grade
- >>> a
- 70
- >>> b
- [80, 65]
- >>>c
- 78

字符串加*，获取子序列

- 字符串也是序列，可以拆包运算
- >>> s="fstring"
- >>> c,*s1=s
- >>> c
- 'f'
- >>> s1
- ['s', 't', 'r', 'i', 'n', 'g']

序列的运算符

- 加号（+）连接2个序列，合并成一个序列

```
a = [2,3,5,7,11,13]
b = [4,6,8,9,10,12]
print(a + b)
```

输出：

[2, 3, 5, 7, 11, 13, 4, 6, 8, 9, 10, 12]

- 乘号（*）重复序列

```
[4,0,4]*3
```

结果是：[4, 0, 4, 4, 0, 4, 4, 0, 4]

判断是否是序列元素

- 检查数据是否在序列中（in）

```
a = [2,3,5,7,11,13]
print(3 in a)
```

输出结果：True

- 对于列表和字符串，in有所不同，比较下面例子：

```
[2,3] in [2,3,5,7,11,13]
```

结果是：False

```
[2,3] in [[2,3],5,7,11,13]
```

结果是： True

```
'e' in 'hello'
```

结果是： True

```
'he' in 'hello'
```

结果是： True

in可以检查某个字符串是否是另一个字符串的一部分。

len,mix.max计算序列的长度

len()函数返回序列内部元素的个数。

```
>>> len([2,3,5,7])
4
>>> len('hello world')
11
```

min()和max()函数计算序列中的最小值和最大值

```
>>> max([2,3,5,7,11,13])
13
>>> min('好好学习天天向上')
'上'
```

注：字符串的大小是按照其Unicode 编码来比较的。

获取序列元素的索引 index

- >>> lst=[1,6,78,4,5,6,2,1,13,1,45]
- >>> lst.index(6) #第一个元素的索引
- 1
- >>> lst.index(9) #无此元素返回错误
- Traceback (most recent call last):
- File "<pyshell>", line 1, in <module>
- ValueError: 9 is not in list
- >>> indexlst=[lst.index(i) for i in lst]
- >>> indexlst
- [0, 1, 2, 3, 4, 1, 6, 0, 8, 0, 10]
- 元素6（红色）的索引值与index的返回值不一致，说明它不是第
- 一次出现

3.2 字符串使用

- 字符串是一连串的字符，用英文单引号（'） 或英文双引号（ " ）括起来。

'Python is the best.'

"Programming is fun."

- 引号必须成对出现；如果字符串中包含了单引号或双引号，则要用另一种引号括起来。

"It's amazing!"

'He said, "You are so cool!'

字符串使用（续）

- 长字符串
  用3个引号（单引号或双引号）括起来的字符串可以包含多行字符串。

```
'''This is a test
for multiple lines
of text.'''
```

表示包含了2个换行符的字符串：

```
'This is a test\nfor multiple lines\nof text.'
```

如果要在程序中用多行表示一个字符串，则可以在每行的结尾用反斜杠（\）结束。

```
'hello \
world'
```

结果是：'hello world'

字符串使用（续2）

- 原始字符串
  在一个字符串字面量前加一个字符r，表示这个字符串是原始字符串，其中的\不被当作是转义字符前缀。

```
r = r'hello\nworld'
print(r)
```

输出：hello\nworld

相当于r = 'hello\\nworld'

字符串使用（续3）

- **字符串是不可修改**

字符串中的数据（字符）是不能修改的。

```
s='hello'
s[0]='k'            #会得到错误
```

可以通过用新的字符串对变量重新赋值，表示新的字符串。

```
s='hello'
s='bye'
```

这样变量s表示字符串'bye'。

字符串常用方法或函数

字符串常用方法或函数	解释
S.title()	字符串S首字母大写
S.lower()，	字符串S变小写
S.upper()	字符串S变大写
S.strip(),S.rstrip(),lstrip()	删除前后空格，删除右空格，删除左空格
S.find(sub[,start[,end]])	在字符串S中查找sub子串首次出现的位置
S.replace(old,new)	在字符串S中用new子串替换old子串
S.join(X)	将序列X合并成字符串
S.split(sep=None)	将字符串S拆分成列表
S.count(sub[,start[,end]])	计算sub子串在字符串S中出现的次数

字符串常用方法或函数（续）

- 查找子串 find()（也可以用index函数）
  - 在字符串中查找子串，返回第一次出现的位置下标（从0开始），如果找不到返回-1。

```
s = 'This is a test.'
print(s.find('is'))
print(s.index('is'))
```

输出：

2

2

```
s = 'This is a test.'
print(s.find('ok'))
```

输出：

-1

字符串常用方法或函数（续2）

```
s = ‘This is a test.’
print(s.find(‘is’,3))      #指定查找开始位置
```

输出：

5

```
s = ‘This is a test.’
print(s.find(‘is’,3,6)) #指定查找开始位置及终止位置
```

输出：

-1

字符串常用方法或函数（续3）

- 统计子串出现的次数count()

```
s = 'This is a test.'
print(s.count('is'))
```

输出：

2

字符串常用方法或函数（续4）

- 修改大小写
  - 函数title() 将字符串中每个单词的首字母变成大写字母。

```
name = ‘john johnson’
print(name.title())
```

输出：

John Johnson

  - 函数upper() 将字符串中所有字母变成大写字母。
  - 函数lower() 将字符串中所有字母变成小写字母。

字符串常用方法或函数（续5）

- 删除两端的空格
  - 函数rstrip() 去掉字符串右边的空格。

```
name = "Python "
name.rstrip()
```

结果是：

'Python'

  - 函数lstrip() 去掉字符串左边的空格。
  - 函数strip() 去掉字符串左右两边的空格。

字符串常用方法或函数（续6）

- 替换字符串中的子串 replace()

```
s = 'This is a test.'
t = s.replace('is', 'eez')
print(t)
```

输出：

Theez eez a test.

将数字转换成字符串

将数字类型的数据（整数、浮点数和复数）转换成字符串，有下列2种方法。

1.函数str()

```
age = 23
print('Happy Birthday '+ str(age) +'!')
```

输出：

Happy Birthday 23!

注:不能写成：print('Happy Birthday '+ age +'!')

将数字转换成字符串

2.format()函数

是字符串的一个函数，也是用来形成格式化的字符串。使用{}来表示占位符。

```
age = 23
'My age is {}'.format(age)
```

格式占位符　被转换的数据

结果是：

'My age is 23'

将数字转换成字符串

- format()函数支持多个占位符，可以为占位符指定的被转换数据的索引。

'my name is {} ,age {}'.format('Mary',18)
结果是：
'my name is Mary ,age 18'

'my name is {1} ,age {0}'.format(10,'Mary')
结果是：
'my name is Mary ,age 10'

将数字转换成字符串

- format()函数也可以指定填充、对齐和宽度，以及精度和进制。它的一般格式是：

{<索引>:<填充字符><对齐方式><最小宽度.精度><格式>}

例：

```
'{0:*>10}'.format(10)  ##右对齐
'{0:*<10}'.format(10)  ##左对齐
'{0:*^10}'.format(10)  ##居中对齐
'{0:6.2f}'.format(1/3)
'{0:b}'.format(10)     #二进制
'{0:o}'.format(10)     #八进制
'{0:x}'.format(10)     #16进制
'{:,}'.format(12345678901) #千分位格式化
"{:10s}".format("10") #字符串左对齐
"{:10d}".format(10)   #数字右对齐
```

```
'********10'
'10********'
'****10****'
'  0.33'
'1010'
'12'
'a'
'12,345,678,901'
'10        '
'        10'
```


用域宽显示对齐列

```
num1=135.877
num2=3672.148
num3=6.345
num4=375.872
num5=77.8
num6=577.888
print('{0:10.2f}{1:10.2f}'.format(num1,num2))
print('{0:10.2f}{1:10.2f}'.format(num3,num4))
print('{0:10.2f}{1:10.2f}'.format(num5,num6))
```

程序输出

```
135.88   3672.15
  6.34    375.87
 77.80    577.89
```


输入四个字符串，求这些字符串的最大长度

```
length=0
for i in range(4):
    a=len(input())
    if a>length:
      length=a
print(length)
```


3.3 列表和元组使用

- 列表（list）
  - 由一系列按照指定顺序排列的元素组成。列表中的元素可以是不同类型。
  - 列表的表示用方括号（[ ]）将元素括起来，元素之间用逗号（,）分隔
  - 列表是序列类型的一种，序列所有的特性和操作对于列表都是成立的，除此之外，列表还有自己的特殊操作。

列表

- 列表的创建

1. 直接使用列表的字面量。

```
a = []          # 创建一个空列表
a = [2,3,5,7,11,13]
```

2. 使用list()将其他数据类型转换成一个列表。

```
a = list('hello')
a的内容是: ['h', 'e', 'l', 'l', 'o']
list(range(1,10,2))
结果是：[1,3,5,7,9]
```

列表（续）

列表的元素类型可以是任何类型，也包括列表类型。当列表的元素是列表时，可以构成多维列表，如同一个矩阵。

```
matrix = [
    [1, 2, 3, 4, 5],
    [3, 0, 8,11,14],
    [5, 6, 9,12,16],
    [7, 0, 0, 0, 0],
    [9,11,17, 0,15]
]
```

	[0]	[1]	[2]	[3]	[4]
[0]	1	2	3	4	5
[1]	3	0	8	11	14
[2]	5	6	9	12	16
[3]	7	0	0	0	0
[4]	9	11	17	0	15

- 用matrix[0][0]访问其中第一行第一列的元素
- 矩阵的每一行都是一个列表。

基本的列表操作

- 列表元素的赋值

  和字符串不同，列表中的元素可以被修改。

```
a = [1,3,5,7,11]
a[0] = 2
print(a)
```

输出：

```
[2,3,5,7,11]
```


基本的列表操作（续）

- 删除元素

  用del语句删除列表中的元素。

```
name = ['Alice', 'Kim', 'Karl', 'John']
del name[2]
print(name)
```

结果是：

```
['Alice', 'Kim', 'John']
```


基本的列表操作（续2）

- 切片赋值

  切片表示列表的一部分，可以被赋值，接受另外一个列表，替换切片那部分元素。

```
name = list('Perl')
name[2:] = list('ar')
print(name)
```

结果是：

```
['P', 'e', 'a', 'r']
```


列表的函数或方法

表 3-4 列表的常用方法或函数

列表的常用方法或函数	描述
L.append(x)	在列表L尾部追加x
L.clear()	移除列表L的所有元素
L.count(x)	计算列表L中x出现的次数
L.copy()	列表L的备份
L.extend(x)	将列表x扩充到列表L中
L.index(value[,start[,stop]])	计算在指定范围内value的下标
L.insert(index,x)	在下标index的位置插入x
L.pop(index)	返回并删除下标为index的元素，默认是最后一个
L.remove(value)	删除值为value的第一个元素
L.reverse()	倒置列表L
L.sort()	对列表元素排序

列表的函数或方法（续）

- 追加函数append()

用函数append()在列表后面增加一个元素。

```
a = [2,3,5,7,11]
a.append(13)
print(a)
```

结果是：

```
[2, 3, 5, 7, 11, 13]
```

列表的函数或方法（续2）

- 扩展函数extend()

用函数extend()把另一个列表的内容添加到列表的后面。

```
a = [2,3,5,6,11]
a.extend([13,17])
print(a)
```

结果就是：

```
[2, 3, 5, 6, 11, 13, 17]
```


列表的函数或方法（续3）

- 插入函数insert()

函数insert()用来将一个数据插入到列表的指定位置。

```
a = [2,3,5,6,11]
a.insert(2,4)
a.insert(12,13) #指定插入的位置(12)不存在，加到最后
print(a)
```

结果是：

```
[2, 3, 4, 5, 6, 11, 13]
```


列表的复制

```
a=[1,2,3,4]
b=a
print(id(a),id(b))
b[2]=5
print(a)

c=a.copy()
print(id(a),id(c))
c[2]=6
print(a)
```

```
d=a[:]
print(id(a),id(d))
d[2]=7
print(a)
```


列表的函数或方法（续4）

- 删除函数remove()

用函数remove()删除某个数据在列表中第一次出现的元素。

```
a = [2,3,5,7,11]
a.remove(5)
print(a)
```

输出：

```
[2, 3, 7, 11]
```

注:如果要删除的数据不在列表中，则会发生错误。

列表的函数或方法（续5）

- 弹出函数pop()

用函数pop()删除并返回列表中指定下标（位置）的数据，如果下标不指定，则删除最后一项。

```
a = [2,3,5,7,11]
print(a.pop())
print(a.pop(2))
print(a)
```

输出：

```
11
5
[2, 3, 7]
```


列表的函数或方法（续6）

- 反转函数reverse()

用函数reverse()将列表反转。

```
a = [2,3,5,7,11]
a.reverse()
print(a)
```

结果是：

```
[11, 7, 5, 3, 2]
```

列表的函数或方法（续7）

- 查找函数index()

函数index()用于在列表中查找某个数据第一出现的位置（下标）。

```
a = [2,3,5,7,11]
print(a.index(3))
```

输出：

```
1
```

注:如果查找的数据在列表中不存在，则会发生错误。

字符串和列表的互相操作

- 拆分字符串函数split()

函数split()用一个字符或子串将一个字符串分隔成列表的元素。

```
date = '3/11/2018'
a = date.split('/')
print(a)
```

输出：

```
['3', '11', '2018']
```

如果split()函数不带参数，就是以空格来分割字符串。

```
name = 'John Johnson'
a = name.split()
print(a)
```

输出：

```
['John', 'Johnson']
```


字符串和列表的互相操作（续）

- 聚合字符串函数join()

函数join()用于将一个列表的各个字符串类型的元素组合成一个字符串，元素之间用指定的内容填充。

```
a = ['hello','good','boy','wii']
print(' '.join(a))
print(':'.join(a))
```

输出：

```
hello good boy wii
hello:good:boy:wii
```


字符串和列表的互相操作（续2）

例3-2 求一句英文句子的单词数。单词是字母数字串，中间没有空格。

```
sentence="This    is a  pen   "
words=sentence.split()
print(len(words))
```

程序运行：

```
4
```


字符串和列表的互相操作（续3）

- 例3-3 在一行中输入若干个整数，至少输入一个整数，整数之间用空格分割，要求将数据按从小到大排序输出。

程序代码：

```
nums=input()
numl=[int(n) for n in nums.split()]
numl.sort()
print(numl)
```

程序输入：

```
5 -76 8 345 67
```

程序输出：

```
[-76, 5, 8, 67, 345]
```

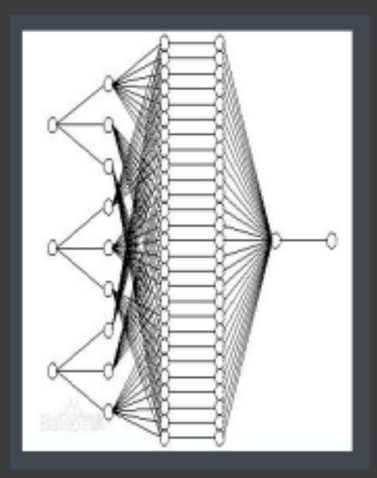

创建列表

- 用append方法：
- lst=[]
- for i in range(4):
- 　　lst.append(input())

- 用列表解析方法：
- lst=[input() for i in range(4)]

输入一行字符串，并将它转换成10进制数输出。

- 神经网络分层



- 如输入： _ahg1*B
- 新字符串：a1B
- 10进制数：2587

```
s=input()     #第一层，输入层

#第二层
lst=[t for t in s if
ord('0')<=ord(t)<=ord('9') \
or ord('a')<=ord(t)<=ord('f')        \
or ord('A')<=ord(t)<=ord('F')]

#第三层
news=''.join(lst)

#第四层 输出层
print(news)
print(int(news,16))
```


列表其他注意点

- 1．如何判断一个列表lst为空？
- len(lst)==0
- not lst ==True

- 2如何将列表lst切成相同长度n的序列？
- [lst[i:i+n] for i in range(0, len(l), n)]

- 遍历list的时候修改某些元素？
-
-
- seq=[1,2,7,3,4,3,2,7,4,5,6,5,4,3]
-
- seq[:]=[x for x in seq if seq.count(x)<2]
- print(seq)

元组

- 元组（tuple）

元组是不可修改的任何类型的数据序列。元组像列表一样可以表达任何类型、任意数量的数据的有序序列。

元组的字面量用圆括号()而不是方括号[]。

(1, 3.2, 5, 7.0, 9)

('not', 'and', 'or')

元组（续）

- 创建元组
  - 用元组的字面量

```
d = (100,20)
print(d)
```

输出：

(100, 20)

  - 用tuple()方法，把其他序列类型转换成元组。

```
a = tuple([2,3,5,7,11])
print(a)
```

输出：

(2, 3, 5, 7, 11)

元组（续2）

- 元组不可修改
  - 元组是不可修改的，即不能对元组中的元素进行增加，删除，修改或排序。
  - 列表中的修改函数append()、insert()、remove()以及del语句都不能用于元组。
  - 元组常用方法和函数

元组常用方法和函数	描述
T.count(x)	计算x元素出现的次数
T.index(x)	计算X元素的下标

输入字符串，排序后按从小到大输出每个字符及该字符在原字符串中的索引。

```
s=input()
lst=[(s[index],index) for index in range(len(s))]
lst.sort()
print(lst)
```

- (s[index],index)是一个元组，保存输入的字符和它的位置。
- 程序输入
- hello python
-
- 程序输出
- [(' ', 5), ('e', 1), ('h', 0), ('h', 9), ('l', 2), ('l', 3), ('n', 11), ('o', 4), ('o', 10), ('p', 6), ('t', 8), ('y', 7)]

列表加元组表示二维表

```
students=[(3180102988,"褚好"),
          (3170102465,"王凯亮"),
          (3160104456,"李永"),
          (3171104169,"陈鑫"),
          (318400429,"徐杭诚")]

for row in students: #按行存取
    print(row[0],row[1])
print()

for id,name in students: #按行拆包存取
    print(id,name)
print()

for index in range(len(students)):  #按索引存取
    print(students[index][0],students[index][1])
```

学号	姓名
3180102988	褚好
3170102465	王凯亮
3160104456	李永
3171104169	陈鑫
318400429	徐杭诚

3.4 随机函数库（random库）

- 计算机的随机函数生成的随机数，是按照一定的算法模拟产生的，其结果是确定的，是伪随机数。
- Python中的random模块用于生成伪随机数。

随机函数库（续）

表 3-5 random库的常用函数

函数名	含义	示例
random.random()	返回一个介于左闭右开[0.0, 1.0)区间的浮点数	random.random()
random.uniform(a, b)	返回一个介于【a，b】的浮点数。	random.uniform(1,10)
random.randint(a,b)	返回【a,b】的一个随机整型。	random.randint(15,30)
random.randrange([start], stop[, step])	从指定范围内，获取一个随机数	random.randrange(10, 30, 2)
random.choice(sequence)	从序列中获取一个随机元素	random.choice([3,78,43,7])
random.shuffle(x)	用于将一个列表中的元素打乱,即将列表内的元素随机排列	random.shuffle(l)，l是序列
random.sample(sequence, k)	从指定序列中随机获取长度为k的序列并随机排列	random.sample([1,4,5,89,7],3)
random.seed(n)	对随机数生成器进行初始化的函数。n代表随机种子。参数为空时，随机种子为系统时间	random.seed(2)

随机函数库（续2）

要使用random库，先要用“import random”语句引入random库。

```
>>> import random
>>> random.random()
0.11529299890219902

>>> random.uniform(1,10)
4.6467045646433975

 >>> random.randint(20,30)
20

>>> random.randrange(10, 30, 2)
24
```


随机函数库（续3）

```
>>> random.choice([3,78,43,7])
3

>>> l=['A',1,78,'b']
>>> random.shuffle(l)
>>> l
[1, 'b', 78, 'A']

>>> random.sample([1,4,5,89,7],3)
 [7, 5, 4]

>>> random.sample("This is a sample",5)
['s', 'h', ' ', 'a', 'a']
```


随机函数库（续4）

```
>>> random.seed(2)
>>> random.random()
0.9560342718892494

>>> random.randint(1,10)
1

 >>> random.seed(2)    #重复上面产生的随机数
>>> random.random()
0.9560342718892494

>>> random.randint(1,10)
1
```

随机函数库（续5）

例3-4 掷硬币，正面向上的概率是多少？

程序代码：

```
#掷10000次硬币，正面向上用1表示，反面向上用0表示。
import random
test=[random.randint(0,1) for i in range(10000)] #产生10000个随机数，值为0或1
print(sum(test) / len(test))
```

程序输出：
0.5006

随机函数库（续6）

例3-5 随机产生8位密码，密码由数字和字母组成。

程序代码：

```
import random

digits=[chr(i) for i in range(48,58)]
ascii_letters=[chr(i) for i in range(65,91)]+[chr(i) for i  in range(97,123)]
# 数字的个数随机产生
num_of_numeric = random.randint(1,7)
# 剩下的都是字母
num_of_letter = 8 - num_of_numeric
# 随机生成数字
numerics = [random.choice(digits) for i in range(num_of_numeric)]
```


随机函数库（续7）

```
# 随机生成字母
letters = [random.choice(ascii_letters) for i in range(num_of_letter)]
# 结合两者
all_chars = numerics + letters
# 重新排列
random.shuffle(all_chars)
# 生成最终字符串
result = ''.join([i for i in all_chars])
print(result)
```

程序输出：
GqG5B429

本章小结

- Python的序列类型数据，包括字符串、列表和元组。作为序列类型，它们有一些共同的操作和函数。
- 字符串是一连串的字符，字符串可以做计算，也可以将其他类型的数据组合进字符串形成格式化的内容来产生程序的输出。
- 列表用来保存任意类型、任意数量的数据。列表中的数据是动态的，随时可以修改，可以增加和删除。而元组则是不可修改的序列类型。
- Python的随机数函数。