

目录

第 1 章 入门准备

01 开篇词：你为什么要学 Python ?

02 我会怎样带你学 Python ?

03 让 Python 在你的电脑上安家落户

04 如何运行 Python 代码？

第 2 章 通用语言特性

05 数据的名字和种类—变量和类型

06 一串数据怎么存—列表和字符串

07 不只有一条路—分支和循环

08 将代码放进盒子—函数

09 知错能改—错误处理、异常机制

10 定制一个模子—类

11 更大的代码盒子—模块和包

12 练习—密码生成器

第 3 章 Python 进阶语言特性

13 这么多的数据结构（一）：列表、元祖、字符串最近阅读

14 这么多的数据结构（二）：字典、集合

15 Python大法初体验：内置函数

16 深入理解下迭代器和生成器

17 生成器表达式和列表生成式

18 把盒子升级为豪宅：函数进阶

19 让你的模子更好用：类进阶

20 从小独栋升级为别墅区：函数式编

13 这么多的数据结构（一）：列表、元祖、字符串

更新时间：2019-09-11 10:02:15



“

人不可有傲气，但不可无傲骨。

——徐悲鸿

”

我们之前学习了字符串和列表，除此之外 Python 中还内置有元组、字典、集合等常用的数据类型，它们都被用于存放批量数据。

存放批量数据用列表不就行了吗，为什么还要这么多数据类型？这是因为在不同的场景下，对数据的读取和修改效率以及内存使用情况的要求是不一样的，为了满足不同的场景需求，需要以不同的组织形式来存放数据。上面所述的那些数据类型，本质上就是不同的数据组织形式，Python 直接为我们提供了它们的现成的实现，我们拿来即可使用，轻而易举地获取各种不同的存放、访问和修改数据的能力。

列表、元祖、字典、集合、字符串这些数据类型中所存放的一个个单独的数据，叫作项（Item）或元素（Element）。这些数据类型除了可以存放元素以外，还能通过调用对象方法来操作管理其中的元素。

我们来详细学习下这五种内置数据类型。

一、列表

列表是 Python 中非常常用的数据类型。之前的章节中我们学习过列表的一些基础知识，这个小节将会更深入地介绍列表的各种功能。

列表是用于存放若干元素的有序序列。列表使用方括号（`[]`）来表示，其中的元素写入方括号中，多个元素时用逗号分隔，如 `[1, 'go', [0.1, 0.2]]`。它的元素可以是任意数据类型，甚至也可以是个列表。

目录	列表被创建之后，我们可以对它做很多操作，包括添加元素，删除元素，修改元素，查找元素等。
第 1 章 入门准备	创建列表
01 开篇词：你为什么要学 Python ？	1. 创建空的列表：
02 我会怎样带你学 Python ？	<div>列表 = []</div> <div>>>> items = [] >>> items []</div>
03 让 Python 在你的电脑上安家落户	2. 创建包含元素的列表：
04 如何运行 Python 代码 ？	<div>列表 = [元素1, 元素2, ..., 元素N]</div> <div>>>> items = [1, 2, 3] >>> items [1, 2, 3]</div>
第 2 章 通用语言特性	列表元素的获取
05 数据的名字和种类—变量和类型	1. 通过索引获取元素
06 一串数据怎么存—列表和字符串	<div>元素 = 列表[索引]</div> <div>>>> letters = ['a' , 'b' , 'c'] >>> letters[2] ' c '</div>
07 不只有一条路—分支和循环	2. 通过元素获取索引
08 将代码放进盒子—函数	这种方式 and 上面相反，首先在列表中寻找元素，然后返回元素对应的索引。
09 知错能改—错误处理、异常机制	<div>索引 = 列表.index(元素)</div> <div>>>> letters = ['a' , 'b' , 'c'] >>> letters.index('c') 2</div>
10 定制一个模子—类	
11 更大的代码盒子—模块和包	
12 练习—密码生成器	
第 3 章 Python 进阶语言特性	
13 这么多的数据结构（一）：列表、元祖、字符串 最近阅读	
14 这么多的数据结构（二）：字典、集合	
15 Python大法初体验：内置函数	
16 深入理解下迭代器和生成器	
17 生成器表达式和列表生成式	
18 把盒子升级为豪宅：函数进阶	
19 让你的模子更好用：类进阶	
20 从小独栋升级为别墅区：函数式编程	

目录

第 1 章 入门准备

- 01 开篇词：你为什么要学 Python ？
- 02 我会怎样带你学 Python ？
- 03 让 Python 在你的电脑上安家落户
- 04 如何运行 Python 代码 ？

第 2 章 通用语言特性

- 05 数据的名字和种类—变量和类型
- 06 一串数据怎么存—列表和字符串
- 07 不只有一条路—分支和循环
- 08 将代码放进盒子—函数
- 09 知错能改—错误处理、异常机制
- 10 定制一个模子—类
- 11 更大的代码盒子—模块和包
- 12 练习—密码生成器

第 3 章 Python 进阶语言特性

- 13 这么多的数据结构（一）：列表、元祖、字符串 最近阅读
- 14 这么多的数据结构（二）：字典、集合
- 15 Python大法初体验：内置函数
- 16 深入理解下迭代器和生成器
- 17 生成器表达式和列表生成式
- 18 把盒子升级为豪宅：函数进阶
- 19 让你的模子更好用：类进阶
- 20 从小独栋升级为别墅区：函数式编程

要想查看元素是否存在于列表中，需要借助 Python 的关键字 `in`，使用如下：

布尔值 = 元素 `in` 列表

```
>>> letters = [ 'a' , 'b' , 'c' ]
>>> 'a' in letters
True
>>> 'z' in letters
False
```

4. 统计元素在列表中的个数

统计元素在列表中的个数，或者说是元素在列表中出现的次数。

个数 = 列表.count(元素)

```
>>> numbers = [1, 2, 2, 3, 4, 5, 5, 7]
>>> numbers.count(5)
2
```

列表元素的添加

我们可以很灵活地向列表添加元素，如以追加的形式向列表末尾添加一个元素；以插入的形式向列表的任意位置插入元素；或者将一个列表中的所有元素批量的添加到另一个列表中。

1. 向列表末尾追加元素

列表.append(元素)

```
>>> letters = [ 'a' , 'b' ]
>>> letters.append( 'c' )
>>> letters
[ 'a' , 'b' , 'c' ]
```

2. 向列表的任意位置插入元素

列表.insert(索引, 元素)

```
>>> letters = [ 'a' , 'b' ]
```

<div>← 慕课专栏</div>	<div>≡ 你的第一本Python基础入门书 / 13 这么多的数据结构（一）：列表、元祖、字符串</div>
<div>目录</div>	<div>['c' , 'a' , 'b']</div>
<div>第 1 章 入门准备</div>	<div>>>> letters.insert(2, 'd') >>> letters</div>
<div>01 开篇词：你为什么要学 Python ？</div>	<div>['c' , 'a' , 'd' , 'b']</div>
<div>02 我会怎样带你学 Python ？</div>	<div>3. 列表末尾追加另一个列表的所有元素</div>
<div>03 让 Python 在你的电脑上安家落户</div>	<div>列表.extend(另一列表)</div>
<div>04 如何运行 Python 代码 ？</div>	<div>>>> letters = ['a' , 'b'] >>> letters.extend(['c' , 'd' , 'e']) >>> letters</div>
<div>第 2 章 通用语言特性</div>	<div>['a' , 'b' , 'c' , 'd' , 'e']</div>
<div>05 数据的名字和种类—变量和类型</div>	<div>列表元素的删除</div>
<div>06 一串数据怎么存—列表和字符串</div>	<div>删除元素的方式同样很灵活。</div>
<div>07 不只有一条路—分支和循环</div>	<div>1. 按索引删除元素</div>
<div>08 将代码放进盒子—函数</div>	<div>元素 = 列表.pop(索引)</div>
<div>09 知错能改—错误处理、异常机制</div>	<div>pop(索引) 会将索引对应的元素从列表中删除，同时返回这个元素。</div>
<div>10 定制一个模子—类</div>	<div>>>> letters = ['a' , 'b' , 'c'] >>> letters.pop(0) 'a' >>> letters</div>
<div>11 更大的代码盒子—模块和包</div>	<div>['b' , 'c']</div>
<div>12 练习—密码生成器</div>	<div>也可以不传递索引，这样的话默认删除并返回最后一个元素。</div>
<div>第 3 章 Python 进阶语言特性</div>	<div>>>> letters = ['a' , 'b' , 'c'] >>> letters.pop() 'c' >>> letters</div>
<div>13 这么多的数据结构（一）：列表、元祖、字符串</div>	<div>['a' , 'b']</div>
<div>14 这么多的数据结构（二）：字典、集合</div>	<div>2. 按索引删除元素（ del 方法）</div>
<div>15 Python大法初体验：内置函数</div>	<div>删除一个列表元素也可以使用 Python 中的 del 关键字，如下：</div>
<div>16 深入理解下迭代器和生成器</div>	<div></div>
<div>17 生成器表达式和列表生成式</div>	<div></div>
<div>18 把盒子升级为豪宅：函数进阶</div>	<div></div>
<div>19 让你的模子更好用：类进阶</div>	<div></div>
<div>20 从小独栋升级为别墅区：函数式编程</div>	<div></div>

目录	
第 1 章 入门准备	
01 开篇词：你为什么要学 Python ？	
02 我会怎样带你学 Python ？	
03 让 Python 在你的电脑上安家落户	
04 如何运行 Python 代码？	
第 2 章 通用语言特性	
05 数据的名字和种类—变量和类型	
06 一串数据怎么存—列表和字符串	
07 不只有一条路—分支和循环	
08 将代码放进盒子—函数	
09 知错能改—错误处理、异常机制	
10 定制一个模子—类	
11 更大的代码盒子—模块和包	
12 练习—密码生成器	
第 3 章 Python 进阶语言特性	
13 这么多的数据结构（一）：列表、元祖、字符串	最近阅读
14 这么多的数据结构（二）：字典、集合	
15 Python大法初体验：内置函数	
16 深入理解下迭代器和生成器	
17 生成器表达式和列表生成式	
18 把盒子升级为豪宅：函数进阶	
19 让你的模子更好用：类进阶	
20 从小独栋升级为别墅区：函数式编程	

```
>>> letters = [ 'a' , 'b' , 'c' ]
>>> del letters[0]
>>> letters
[ 'b' , 'c' ]
```

3. 直接删除元素

直接删除元素时，Python 会先在列表中遍历该元素，然后将匹配到的第一个元素删除。

```
列表.remove(元素)
```

```
>>> letters = [ 'a' , 'b' , 'c' ]
>>> letters.remove( 'b' )
>>> letters
[ 'a' , 'c' ]
```

4. 清空所有元素

清空所有元素即是把列表元素全部删除，最后仅为列表仅为 `[]`。

```
列表.clear()
```

```
>>> letters = [ 'a' , 'b' , 'c' ]
>>> letters.clear()
>>> letters
[]
```

列表元素的修改

1. 通过赋值修改列表元素

```
列表[索引] = 新元素
```

```
>>> letters = [ 'a' , 'b' , 'c' ]
>>> letters[2] = 'd'
>>> letters
[ 'a' , 'b' , 'd' ]
```

2. 反转整个列表

<div>← 慕课专栏</div>	<div>≡ 你的第一本Python基础入门书 / 13 这么多的数据结构（一）：列表、元祖、字符串</div>
<div>目录</div>	<div>列表.reverse()</div>
<div>第 1 章 入门准备</div>	
<div>01 开篇词：你为什么要学 Python ？</div>	
<div>02 我会怎样带你学 Python ？</div>	<div>>>> letters = ['a' , 'b' , 'c'] >>> letters.reverse() >>> letters ['c' , 'b' , 'a']</div>
<div>03 让 Python 在你的电脑上安家落户</div>	
<div>04 如何运行 Python 代码 ？</div>	
<div>第 2 章 通用语言特性</div>	<div>列表.sort()</div>
<div>05 数据的名字和种类—变量和类型</div>	
<div>06 一串数据怎么存—列表和字符串</div>	<div>>>> numbers = [2, 4, 5, 2, 1, 5, 7, 3] >>> numbers.sort() >>> numbers [1, 2, 2, 3, 4, 5, 5, 7]</div>
<div>07 不只有一条路—分支和循环</div>	
<div>08 将代码放进盒子—函数</div>	<div>3. 列表元素排序</div>
<div>09 知错能改—错误处理、异常机制</div>	<div>也可以通过指定 sort 方法的 reverse 参数来倒序排列。</div>
<div>10 定制一个模子—类</div>	<div>列表.sort(reverse=True)</div>
<div>11 更大的代码盒子—模块和包</div>	
<div>12 练习—密码生成器</div>	<div>>>> numbers = [2, 4, 5, 2, 1, 5, 7, 3] >>> numbers.sort(reverse=True) >>> numbers [7, 5, 5, 4, 3, 2, 2, 1]</div>
<div>第 3 章 Python 进阶语言特性</div>	
<div>13 这么多的数据结构（一）：列表、元祖、字符串</div>	<div>二、元组</div>
<div>14 这么多的数据结构（二）：字典、集合</div>	<div>元组和列表非常相似，也是用于存放元素的有序序列。它用的圆括号（ () ）表示，元素写入圆括号中，多个元素时用逗号分隔，如 (1, 2, 3)。</div>
<div>15 Python大法初体验：内置函数</div>	<div>元组同样具有索引，索引使用方式与列表一致。其元素同样可以是任意类型。</div>
<div>16 深入理解下迭代器和生成器</div>	<div>看起来元组就是披着圆括号外衣的列表嘛！有什么区别？</div>
<div>17 生成器表达式和列表生成式</div>	<div>元组创建完成后，便不能向其中添加元素，也不能修改和删除其中的任何一个元素。所以它与列表相比，只能查找元素，也就是说只具备读的功能，不具备写的功能。元组的这一特性叫作不可变（性）（Immutable），而列表是可变的（Mutable）。</div>
<div>18 把盒子升级为豪宅：函数进阶</div>	<div>创建元组</div>
<div>19 让你的模子更好用：类进阶</div>	<div>1. 创建空的元组：</div>
<div>20 从小独栋升级为别墅区：函数式编程</div>	<div>元组 = ()</div>

<div>← 慕课专栏</div>	<div>☰ 你的第一本Python基础入门书 / 13 这么多的数据结构（一）：列表、元祖、字符串</div>
<div>目录</div>	<div>>>> items = () >>> items ()</div>
<div>第 1 章 入门准备</div>	
<div>01 开篇词：你为什么要学 Python ？</div>	
<div>02 我会怎样带你学 Python ？</div>	<div>2. 创建包含多个元素的元组：</div>
<div>03 让 Python 在你的电脑上安家落户</div>	<div>元组 = (元素1, 元素2, ..., 元素N)</div>
<div>04 如何运行 Python 代码？</div>	<div>>>> items = (1, 2, 3) >>> items (1, 2, 3)</div>
<div>第 2 章 通用语言特性</div>	
<div>05 数据的名字和种类—变量和类型</div>	
<div>06 一串数据怎么存—列表和字符串</div>	
<div>07 不只有一条路—分支和循环</div>	<div>3. 创建只包含一个元素的元组</div>
<div>08 将代码放进盒子—函数</div>	<div>只包含一个元素的情况需要单独说明一下，因为它的形式与直觉不相符。</div>
<div>09 知错能改—错误处理、异常机制</div>	<div>创建只包含一个元素的元组，需要在唯一的那个元素后面加上逗号，如：</div>
<div>10 定制一个模子—类</div>	<div>元组 = (元素,)</div>
<div>11 更大的代码盒子—模块和包</div>	<div>>>> items = (1,) >>> items (1,)</div>
<div>12 练习—密码生成器</div>	
<div>第 3 章 Python 进阶语言特性</div>	
<div>13 这么多的数据结构（一）：列表、元祖、字符串</div>	<div>这是因为，如果括号中只有一个元素，那么 Python 会将这个括号当作优先级符号进行处理（像数学中的那样），而不是当作元组。可以试一下：</div>
<div>14 这么多的数据结构（二）：字典、集合</div>	<div>>>> items = (1) >>> items 1 >>> type(items) <class 'int' ></div>
<div>15 Python大法初体验：内置函数</div>	
<div>16 深入理解下迭代器和生成器</div>	
<div>17 生成器表达式和列表生成式</div>	<div>元组元素的获取</div>
<div>18 把盒子升级为豪宅：函数进阶</div>	<div>1. 通过索引获取元素</div>
<div>19 让你的模子更好用：类进阶</div>	<div>元素 = 元组[索引]</div>
<div>20 从小独栋升级为别墅区：函数式编程</div>	

<div>← 慕课专栏</div> <div>你的第一本Python基础入门书 / 13 这么多的数据结构（一）：列表、元组、字符串</div>	
目录	>>> letters[2] ' c '
第 1 章 入门准备	
01 开篇词：你为什么要学 Python ？	2. 通过元素获取索引
02 我会怎样带你学 Python ？	索引 = 元组.index(元素)
03 让 Python 在你的电脑上安家落户	>>> letters = (' a ' , ' b ' , ' c ') >>> letters.index(' c ') 2
04 如何运行 Python 代码 ？	3. 查看元素是否存在于元组中
第 2 章 通用语言特性	布尔值 = 元素 in 元组
05 数据的名字和种类—变量和类型	>>> letters = (' a ' , ' b ' , ' c ') >>> ' a ' in letters True >>> ' z ' in letters False
06 一串数据怎么存—列表和字符串	4. 统计元素在元组中出现的个数
07 不只有一条路—分支和循环	个数 = 元组.count(元素)
08 将代码放进盒子—函数	>>> numbers = (1, 2, 2, 3, 4, 5, 5, 7) >>> numbers.count(5) 2
09 知错能改—错误处理、异常机制	
10 定制一个模子—类	
11 更大的代码盒子—模块和包	
12 练习—密码生成器	
第 3 章 Python 进阶语言特性	
13 这么多的数据结构（一）：列表、元组、字符串 最近阅读	
14 这么多的数据结构（二）：字典、集合	
15 Python大法初体验：内置函数	元组和列表的差别
16 深入理解下迭代器和生成器	我们可以看到，元组所具有的操作在使用方式上与和列表非常相似，甚至在一定程度上可以将元组看作是列表的精简版，但它们之间也有明显的差别。
17 生成器表达式和列表生成式	<ul style="list-style-type: none">元组是不可变的（Immutable），列表是可变的（Mutable），元组在被创建之后，就不能添加、删除和修改元素，而列表可以一般情况下元组的性能在略高于列表
18 把盒子升级为豪宅：函数进阶	我们在什么时候用列表，什么时候用元组？
19 让你的模子更好用：类进阶	
20 从小独栋升级为别墅区：函数式编程	

<div><div>← 慕课专栏</div><div>三 你的第一本Python基础入门书 / 13 这么多的数据结构（一）：列表、元祖、字符串</div></div> <div><div>目录</div><div><div>第 1 章 入门准备</div><div>01 开篇词：你为什么要学 Python？</div><div>02 我会怎样带你学 Python？</div><div>03 让 Python 在你的电脑上安家落户</div><div>04 如何运行 Python 代码？</div><div>第 2 章 通用语言特性</div><div>05 数据的名字和种类—变量和类型</div><div>06 一串数据怎么存—列表和字符串</div><div>07 不只有一条路—分支和循环</div><div>08 将代码放进盒子—函数</div><div>09 知错能改—错误处理、异常机制</div><div>10 定制一个模子—类</div><div>11 更大的代码盒子—模块和包</div><div>12 练习—密码生成器</div><div>第 3 章 Python 进阶语言特性</div><div>13 这么多的数据结构（一）：列表、元祖、字符串最近阅读</div><div>14 这么多的数据结构（二）：字典、集合</div><div>15 Python大法初体验：内置函数</div><div>16 深入理解下迭代器和生成器</div><div>17 生成器表达式和列表生成式</div><div>18 把盒子升级为豪宅：函数进阶</div><div>19 让你的模子更好用：类进阶</div><div>20 从小独栋升级为别墅区：函数式编程</div></div></div>	<div><p>列表；当我们希望所有元素在创建之后便不再改变，可使用元组。</p><h3>三、字符串</h3><p>字符串也是 Python 中非常常用的内置数据类型。我们之前学习过字符串的一些内容，现在来深入的了解下。</p><p>字符串是 Python 中用来存放字符序列的数据类型，其中的元素只能是字符。字符串使用单引号或双引号来表示，如 <code>'pick'</code>，<code>"cherry"</code>，通常我们首先使用单引号。</p><p>字符串是有序序列，可以使用索引来获取其中某个位置的元素。它是不可变的，被创建之后其中的元素（也就是字符）不能被修改和删除。</p><h4>创建字符串</h4><div><div>1. 创建空字符串（即不包含字符的字符串）：</div><div><pre>字符串 = ""</pre><pre>>>> string = "" >>> string ' '</pre></div><div><div>2. 创建包含元素的字符串：</div><div><pre>字符串 = '若干字符'</pre><pre>>>> string = 'happy' >>> string ' happy'</pre></div></div><h4>字符的获取</h4><div><div>1. 通过索引获取字符</div><div><pre>字符 = 字符串[索引]</pre><pre>>>> string = 'happy' >>> string[2] 'p'</pre></div><div><div>2. 通过子串获取索引</div></div></div></div></div>
--	--

<div>← 慕课专栏</div>	<div>≡ 你的第一本Python基础入门书 / 13 这么多的数据结构（一）：列表、元祖、字符串</div>
<div>目录</div>	<div>的是子串的第一个字符的索引。</div>
<div>第 1 章 入门准备</div>	<div>索引 = 字符串.index(字符)</div>
<div>01 开篇词：你为什么要学 Python ？</div>	
<div>02 我会怎样带你学 Python ？</div>	<div>>>> string = 'happy' >>> string.index('p') 2</div>
<div>03 让 Python 在你的电脑上安家落户</div>	
<div>04 如何运行 Python 代码？</div>	<div>>>> string = 'happy' >>> string.index('app') 1</div>
<div>第 2 章 通用语言特性</div>	
<div>05 数据的名字和种类—变量和类型</div>	
<div>06 一串数据怎么存—列表和字符串</div>	<div>当字符或子串不存在时， index 方法将抛出 ValueError 错误。</div>
<div>07 不只有一条路—分支和循环</div>	<div>也可采用字符串的 find 方法来查找子串，使用方式与 index 一致，不同点在于 find 方法未找到子串时返回数字 -1，而不抛异常。</div>
<div>08 将代码放进盒子—函数</div>	<div>>>> string = 'happy' >>> string.find('app') 1 >>> string.find('z') -1</div>
<div>09 知错能改—错误处理、异常机制</div>	
<div>10 定制一个模子—类</div>	
<div>11 更大的代码盒子—模块和包</div>	
<div>12 练习—密码生成器</div>	
<div>第 3 章 Python 进阶语言特性</div>	<div>3. 查看字符是否存在于字符串中</div>
<div>13 这么多的数据结构（一）：列表、元祖、字符串</div>	<div>查看字符是否存在于字符串中，需要借助 Python 的关键字 in，如下：</div>
<div><div>最近阅读</div></div>	<div>布尔值 = 字符 in 字符串</div>
<div>14 这么多的数据结构（二）：字典、集合</div>	<div>>>> string = 'happy' >>> 'a' in string True >>> 'z' in string False</div>
<div>15 Python大法初体验：内置函数</div>	
<div>16 深入理解下迭代器和生成器</div>	
<div>17 生成器表达式和列表生成式</div>	
<div>18 把盒子升级为豪宅：函数进阶</div>	<div>4. 统计字符在字符串中的个数</div>
<div>19 让你的模子更好用：类进阶</div>	<div>个数 = 字符串.count(字符)</div>
<div>20 从小独栋升级为别墅区：函数式编程</div>	

<div>← 慕课专栏</div> <div>三 你的第一本Python基础入门书 / 13 这么多的数据结构（一）：列表、元祖、字符串</div>	
目录	<pre>>>> string.count('p') 2</pre>
第 1 章 入门准备	
01 开篇词：你为什么要学 Python ？	字符串的处理
02 我会怎样带你学 Python ？	字符串自带的方法非常多，除了上面介绍的几个之外还有四十多个，这是因为字符处理是编程时的一项高频工作。Python 将这些字符处理相关的功能以方法的形式集成在字符串里。
03 让 Python 在你的电脑上安家落户	这里列举几个常见的方法：
04 如何运行 Python 代码？	<ul style="list-style-type: none">• startswith：判断字符串是否以某个子串开头，返回布尔值
第 2 章 通用语言特性	<pre>>>> string = 'happy' >>> string.startswith('ha') True</pre>
05 数据的名字和种类—变量和类型	<ul style="list-style-type: none">• endswith：判断字符串是否以某个子串结尾，返回布尔值
06 一串数据怎么存—列表和字符串	<pre>>>> string = 'happy' >>> string.endswith('y') True</pre>
07 不只有一条路—分支和循环	<ul style="list-style-type: none">• replace：将字符串的子串用一个另一个字符串替换，返回一个新的字符串
08 将代码放进盒子—函数	<pre>>>> string = 'happy' >>> string.replace('y', 'iness') 'happiness'</pre>
09 知错能改—错误处理、异常机制	<ul style="list-style-type: none">• strip：去除字符串前后的空白符号，如空格、换行符、制表符，返回一个新的字符串
10 定制一个模子—类	<pre>>>> string = ' \t happy \n' >>> string.strip() ' happy '</pre>
11 更大的代码盒子—模块和包	<ul style="list-style-type: none">• split：将字符串用某个子串分隔开，分隔后的各个部分放入列表中，并返回这个列表
12 练习—密码生成器	<pre>>>> string = 'I am happy' >>> string.split(' ') ['I', 'am', 'happy']</pre>
第 3 章 Python 进阶语言特性	
13 这么多的数据结构（一）：列表、元祖、字符串 <small>最近阅读</small>	
14 这么多的数据结构（二）：字典、集合	
15 Python大法初体验：内置函数	
16 深入理解下迭代器和生成器	
17 生成器表达式和列表生成式	
18 把盒子升级为豪宅：函数进阶	
19 让你的模子更好用：类进阶	
20 从小独栋升级为别墅区：函数式编程	

<div>← 慕课专栏</div>	<div>≡ 你的第一本Python基础入门书 / 13 这么多的数据结构（一）：列表、元祖、字符串</div>
<div data-bbox="97 192 148 221">目录</div> <div data-bbox="97 266 250 293">第 1 章 入门准备</div> <div data-bbox="97 338 426 365">01 开篇词：你为什么要学 Python？</div> <div data-bbox="97 423 359 450">02 我会怎样带你学 Python？</div> <div data-bbox="97 506 429 533">03 让 Python 在你的电脑上安家落户</div> <div data-bbox="97 591 346 618">04 如何运行 Python 代码？</div> <div data-bbox="97 669 290 696">第 2 章 通用语言特性</div> <div data-bbox="97 741 411 768">05 数据的名字和种类—变量和类型</div> <div data-bbox="97 824 411 851">06 一串数据怎么存—列表和字符串</div> <div data-bbox="97 909 371 936">07 不只有一条路—分支和循环</div> <div data-bbox="97 992 331 1019">08 将代码放进盒子—函数</div> <div data-bbox="97 1077 411 1104">09 知错能改—错误处理、异常机制</div> <div data-bbox="97 1160 290 1187">10 定制一个模子—类</div> <div data-bbox="97 1245 371 1272">11 更大的代码盒子—模块和包</div> <div data-bbox="97 1328 290 1355">12 练习—密码生成器</div> <div data-bbox="97 1408 368 1435">第 3 章 Python 进阶语言特性</div> <div data-bbox="97 1480 456 1538">13 这么多的数据结构（一）：列表、元祖、字符串<div>最近阅读</div></div> <div data-bbox="97 1597 419 1655">14 这么多的数据结构（二）：字典、集合</div> <div data-bbox="97 1713 397 1740">15 Python大法初体验：内置函数</div> <div data-bbox="97 1796 370 1823">16 深入理解下迭代器和生成器</div> <div data-bbox="97 1881 370 1908">17 生成器表达式和列表生成式</div> <div data-bbox="97 1964 389 1991">18 把盒子升级为豪宅：函数进阶</div> <div data-bbox="97 2049 370 2076">19 让你的模子更好用：类进阶</div> <div data-bbox="97 2132 430 2159">20 从小独栋升级为别墅区：函数式编程</div>	<div data-bbox="612 188 657 215">字符串</div> <div data-bbox="646 311 1056 421"><pre>>>> words = ['I' , 'am' , 'happy'] >>> ' '.join(words) 'I am happy'</pre></div> <div data-bbox="593 512 1187 539"><ul style="list-style-type: none">• upper：将字符串转化为大写字母形式，返回一个新的字符串</div> <div data-bbox="646 633 874 743"><pre>>>> string = 'happy' >>> string.upper() 'HAPPY'</pre></div> <div data-bbox="593 837 1184 864"><ul style="list-style-type: none">• lower：将字符串转化为小写字母形式，返回一个新的字符串</div> <div data-bbox="646 958 885 1070"><pre>>>> string = 'HAPPY' >>> string.lower() 'happy'</pre></div> <div data-bbox="560 1164 1461 1234"><p>注意上面的这些字符处理功能，对字符串作处理后都是返回一个新的字符串，而不会直接修改原有的字符串。为什么呢？字符串不可变呀！</p></div> <div data-bbox="560 1283 646 1310"><p>字符转义</p></div> <div data-bbox="560 1357 1461 1426"><p>我们在创建字符串时，有一些字符是没法直接在引号中表示的，如单引号或双引号，因为这和表示字符串本身的符号冲突了，如：</p></div> <div data-bbox="593 1520 877 1720"><pre>>>> string = 'I' m happy' File "<code></code>" , line 1 string = 'I' m happy' ^ SyntaxError: invalid syntax</pre></div> <div data-bbox="560 1812 1461 1881"><p>抛出 <code>SyntaxError</code> 语法错误异常，因为 Python 将 <code>string = 'I' m happy'</code> 看作字符串的赋值，而后面的 <code>m happy'</code> 就无法解析了，因为不符合任何语法。</p></div> <div data-bbox="560 1928 1461 1998"><p>这时就需要使用字符转义了，我们在这类无法直接在字符串中表示的字符前加上 <code>\</code> 符号，形如 <code>\'</code>，这样 Python 在解析时就能理解这是嵌入在字符串中的单引号 <code>'</code>：</p></div>

<div>← 慕课专栏</div> <div>你的第一本Python基础入门书 / 13 这么多的数据结构（一）：列表、元祖、字符串</div>															
目录	<pre>>>> string "I' m happy"</pre>														
第 1 章 入门准备															
01 开篇词：你为什么要学 Python ？	像 <code>\</code> 这样在前面加了反斜杠（ <code>\</code> ），为了能在字符串中正常表示的字符，叫作转义字符。而这个转化的行为叫作字符转义。														
02 我会怎样带你学 Python ？	与单引号的用法相同，双引号用 <code>\</code> 来转义。														
03 让 Python 在你的电脑上安家落户	字符串中的 <code>\</code> 用来做字符转义了，那怎么在字符串中表示斜杆 <code>\</code> 这个字符呢？使用 <code>\\</code> ，将斜杆 <code>\</code> 转义一下。除此之外，有一些空白符号也需要用转义字符来表示，因为我们没办法直接在字符串中表示它们，如常见的换行（ <code>\n</code> ），制表符（ <code>\t</code> ），回车（ <code>\r</code> ）。														
04 如何运行 Python 代码？	<table><tr><th>常用的转义字符</th><th>含义</th></tr><tr><td><code>\'</code></td><td>单引号</td></tr><tr><td><code>\"</code></td><td>双引号</td></tr><tr><td><code>\\</code></td><td>反斜杠</td></tr><tr><td><code>\n</code></td><td>换行符</td></tr><tr><td><code>\t</code></td><td>制表符（Tab）</td></tr><tr><td><code>\r</code></td><td>回车</td></tr></table>	常用的转义字符	含义	<code>\'</code>	单引号	<code>\"</code>	双引号	<code>\\</code>	反斜杠	<code>\n</code>	换行符	<code>\t</code>	制表符（Tab）	<code>\r</code>	回车
常用的转义字符	含义														
<code>\'</code>	单引号														
<code>\"</code>	双引号														
<code>\\</code>	反斜杠														
<code>\n</code>	换行符														
<code>\t</code>	制表符（Tab）														
<code>\r</code>	回车														
第 2 章 通用语言特性	举个例子，如果在字符串使用了 <code>\n</code> ，那么在用 <code>print()</code> 输出字符串的时候，这个字符串会被换行输出。如：														
05 数据的名字和种类—变量和类型	<pre>>>> print('第一行\n第二行') 第一行 第二行</pre>														
06 一串数据怎么存—列表和字符串	使用 <code>\n</code> 换行符使得我们能够在一行的字符串来表示多行的内容。														
07 不只有一条路—分支和循环	说明：转义字符虽然在书写时使用了两个字符，但是在程序中它只是一个字符。可以自己来试验下：														
08 将代码放进盒子—函数	<pre>>>> len(' \n') 1 >>> len(' \' ') 1</pre>														
09 知错能改—错误处理、异常机制	如果我们就想在字符串中表示 <code>\n</code> 这两个字符，而不是让它表示换行，该怎么办？有两种方式：														
10 定制一个模子—类	1. 使用 <code>\\n</code> ，将 <code>\n</code> 前面的反斜杠转义														
11 更大的代码盒子—模块和包															
12 练习—密码生成器															
第 3 章 Python 进阶语言特性															
13 这么多的数据结构（一）：列表、元祖、字符串 最近阅读															
14 这么多的数据结构（二）：字典、集合															
15 Python大法初体验：内置函数															
16 深入理解下迭代器和生成器															
17 生成器表达式和列表生成式															
18 把盒子升级为豪宅：函数进阶															
19 让你的模子更好用：类进阶															
20 从小独栋升级为别墅区：函数式编程															

<div>← 慕课专栏</div> <div>三 你的第一本Python基础入门书 / 13 这么多的数据结构（一）：列表、元祖、字符串</div>	
目录	
第 1 章 入门准备	2. 使用原始字符串
01 开篇词：你为什么要学 Python？	原始字符串
02 我会怎样带你学 Python？	原始字符串就是在字符串的起始引号前加上一个 r 或 R 字母，这样字符串中的内容将不会被转义，将按照原样输出。使用方法：
03 让 Python 在你的电脑上安家落户	<code>r'字符串内容'</code>
04 如何运行 Python 代码？	<pre>>>> print(r' 第一行\n第二行') 第一行\n第二行</pre>
第 2 章 通用语言特性	多行字符串
05 数据的名字和种类—变量和类型	我们之前所使用的字符串都被书写成一行，要想让字符串可以跨行书写，写成多行的形式，有两种方法：
06 一串数据怎么存—列表和字符串	1. 字符串的每行末尾使用 \ 续行
07 不只有一条路—分支和循环	以多行的形式书写字符串，每行的末尾使用 \ 续行。需要注意输出内容为一行。
08 将代码放进盒子—函数	<pre>string = '第一行\ 第二行\ 第三行'</pre>
09 知错能改—错误处理、异常机制	<pre>>>> string = '第一行\ … 第二行\ … 第三行' >>> print(string) ‘第一行第二行第三行’</pre>
10 定制一个模子—类	可以看到这种方式可以让字符串以多行的方式来书写，但是输出内容还是被当作一行。如果想要输出内容为多行，需要在字符串中显式地使用 \n 进行换行。
11 更大的代码盒子—模块和包	<pre>>>> string = '第一行\n\ … 第二行\n\ … 第三行' >>> print(string) 第一行 第二行 第三行</pre>
12 练习—密码生成器	
第 3 章 Python 进阶语言特性	
13 这么多的数据结构（一）：列表、元祖、字符串 最近阅读	
14 这么多的数据结构（二）：字典、集合	
15 Python大法初体验：内置函数	
16 深入理解下迭代器和生成器	
17 生成器表达式和列表生成式	
18 把盒子升级为豪宅：函数进阶	
19 让你的模子更好用：类进阶	
20 从小独栋升级为别墅区：函数式编程	

← 慕课专栏	三 你的第一本Python基础入门书 / 13 这么多的数据结构（一）：列表、元组、字符串
目录	在 Python 中字符串也可以使用三个单引号或三个双引号来表示字符串，这样子字符串中的内容就可以多行书写，并且被多行输出。
第 1 章 入门准备	<div><pre>string = '''第一行 第二行 第三行'''</pre></div> <div><pre>>>> string = ' ' '第一行 ... 第二行 ... 第三行' ' ' >>> print(string) 第一行 第二行 第三行</pre></div> <p>使用三引号的方式，字符串可被多行书写，且被多行输出，其中不需要显式地指明 <code>\n</code> 换行。</p>
01 开篇词：你为什么要学 Python？	
02 我会怎样带你学 Python？	
03 让 Python 在你的电脑上安家落户	
04 如何运行 Python 代码？	
第 2 章 通用语言特性	列表、元组、字符串的通用操作
05 数据的名字和种类—变量和类型	我们把列表、元组、字符串统称为序列。
06 一串数据怎么存—列表和字符串	1. 使用 <code>len()</code> 函数获取序列长度
07 不只有一条路—分支和循环	<div><pre>>>> letters = ('a' , 'b') >>> len(letters) 2</pre></div> <div><pre>>>> letters = 'abcd' >>> len(letters) 4</pre></div>
08 将代码放进盒子—函数	2. 获取序列中的一个子序列
09 知错能改—错误处理、异常机制	获取序列中的子序列可以使用切片，以 <code>[起始索引:结束索引]</code> 表示。切片其实代表一个索引区间，这个区间是一个左开右闭区间，该区间内的所有元素作为子序列被返回。如：
10 定制一个模子—类	<div><pre>>>> numbers = (1, 2, 3, 4, 5) >>> numbers[0:2] (1, 2)</pre></div>
11 更大的代码盒子—模块和包	
12 练习—密码生成器	
第 3 章 Python 进阶语言特性	
13 这么多的数据结构（一）：列表、元组、字符串 最近阅读	
14 这么多的数据结构（二）：字典、集合	
15 Python大法初体验：内置函数	
16 深入理解下迭代器和生成器	
17 生成器表达式和列表生成式	
18 把盒子升级为豪宅：函数进阶	
19 让你的模子更好用：类进阶	
20 从小独栋升级为别墅区：函数式编程	

<div><div>← 慕课专栏</div><div>你的第一本Python基础入门书 / 13 这么多的数据结构（一）：列表、元组、字符串</div></div>	
目录	
第 1 章 入门准备	
01 开篇词：你为什么要学 Python ？	<pre>>>> letters = 'abcd' >>> letters[1:5] ' bcd'</pre>
02 我会怎样带你学 Python ？	
03 让 Python 在你的电脑上安家落户	3. 使用 + 符号来拼接两个序列
04 如何运行 Python 代码 ？	<pre>>>> letters_1 = ('a' , 'b') >>> letters_2 = ('c' , 'd' , 'e') >>> letters_1 + letters_2 ('a' , 'b' , 'c' , 'd' , 'e')</pre>
第 2 章 通用语言特性	
05 数据的名字和种类—变量和类型	
06 一串数据怎么存—列表和字符串	
07 不只有一条路—分支和循环	<pre>>>> letters_1 = 'ab' >>> letters_2 = 'cde' >>> letters_1 + letters_2 ' abcde'</pre>
08 将代码放进盒子—函数	
09 知错能改—错误处理、异常机制	
10 定制一个模子—类	4. 使用 * 符号来重复序列中的元素
11 更大的代码盒子—模块和包	<pre>>>> letters = ('a' , 'b') >>> letters * 3 ('a' , 'b' , 'a' , 'b' , 'a' , 'b')</pre>
12 练习—密码生成器	
第 3 章 Python 进阶语言特性	
13 这么多的数据结构（一）：列表、元组、字符串 <div>最近阅读</div>	<pre>>>> letters = 'abcd' >>> letters * 2 ' abcdabcd'</pre>
14 这么多的数据结构（二）：字典、集合	注意上面的操作结果都是返回一个新的序列，不会对修改序列的内部元素。
15 Python大法初体验：内置函数	总结
16 深入理解下迭代器和生成器	列表、元组、字符串都是有序序列，都可以使用索引。
17 生成器表达式和列表生成式	列表和元组中可以存放任意数据类型的元素，而字符串中只能存放字符。
18 把盒子升级为豪宅：函数进阶	列表是可变的，而元组和字符串是不可变的。
19 让你的模子更好用：类进阶	
20 从小独栋升级为别墅区：函数式编程	

<div><div>← 慕课专栏</div><div>☰ 你的第一本Python基础入门书 / 13 这么多的数据结构（一）：列表、元祖、字符串</div></div>	
目录	精选留言 1
第 1 章 入门准备	
01 开篇词：你为什么要学 Python？	欢迎在这里发表留言，作者筛选后可公开显示
02 我会怎样带你学 Python？	
03 让 Python 在你的电脑上安家落户	Leo梁 列表.sort(reverse=True)，这个如何理解？不懂
04 如何运行 Python 代码？	👍 0 回复 2019-11-29
第 2 章 通用语言特性	
05 数据的名字和种类—变量和类型	千学不如一看，千看不如一练
06 一串数据怎么存—列表和字符串	
07 不只有一条路—分支和循环	
08 将代码放进盒子—函数	
09 知错能改—错误处理、异常机制	
10 定制一个模子—类	
11 更大的代码盒子—模块和包	
12 练习—密码生成器	
第 3 章 Python 进阶语言特性	
13 这么多的数据结构（一）：列表、元祖、字符串	最近阅读
14 这么多的数据结构（二）：字典、集合	
15 Python大法初体验：内置函数	
16 深入理解下迭代器和生成器	
17 生成器表达式和列表生成式	
18 把盒子升级为豪宅：函数进阶	
19 让你的模子更好用：类进阶	
20 从小独栋升级为别墅区：函数式编程	