第三章：数据类型与对象

Table of Contents

[1. 序言 3](#_Toc50315954)

[1.1 章节简介 3](#_Toc50315955)

[2. 常量与变量 3](#_Toc50315956)

[2.1 变量是什么 3](#_Toc50315957)

[2.2 常量是什么 5](#_Toc50315958)

[3. 基础类型-数值类型 5](#_Toc50315959)

[3.1 布尔型 5](#_Toc50315960)

[3.2 整型 6](#_Toc50315961)

[3.3 浮点型 6](#_Toc50315962)

[3.4 空值类型 6](#_Toc50315963)

[4. 基础类型-字符串类型 7](#_Toc50315964)

[4.1 下标和切片 7](#_Toc50315965)

[4.1.1 下标索引 7](#_Toc50315966)

[4.1.2 切片 7](#_Toc50315967)

[4.2 字符串的常见操作 8](#_Toc50315968)

[4.2.1 find 8](#_Toc50315969)

[4.2.2 index 8](#_Toc50315970)

[4.2.3 count 8](#_Toc50315971)

[4.2.4 replace 9](#_Toc50315972)

[4.2.5 split 9](#_Toc50315973)

[4.2.6 startwith 9](#_Toc50315974)

[4.2.7 endswith 9](#_Toc50315975)

[4.2.8 upper 9](#_Toc50315976)

[4.2.9 lstrip 10](#_Toc50315977)

[5. 容器类型-列表list 10](#_Toc50315978)

[5.1 列表简介 10](#_Toc50315979)

[5.2 列表的常见操作 10](#_Toc50315980)

[5.2.1 列表的长度 10](#_Toc50315981)

[5.2.2 列表的访问 10](#_Toc50315982)

[5.2.3 列表的切片 11](#_Toc50315983)

[5.2.4 给列表添加元素 11](#_Toc50315984)

[5.2.5 修改元素 12](#_Toc50315985)

[5.2.6 查找元素 12](#_Toc50315986)

[5.2.7 删除元素 13](#_Toc50315987)

[5.2.8 排序 13](#_Toc50315988)

[5.2.9 列表遍历 14](#_Toc50315989)

[6. 容器类型-元组tuple 14](#_Toc50315990)

[7. 容器类型-字典dict 15](#_Toc50315991)

[7.1 字典简介 15](#_Toc50315992)

[7.2 字典的常见操作 16](#_Toc50315993)

[7.2.1 修改元素 16](#_Toc50315994)

[7.2.2 添加元素 16](#_Toc50315995)

[7.2.3 删除元素 17](#_Toc50315996)

[7.2.4 查看字典中有多少个元素 17](#_Toc50315997)

[7.2.5 返回字典中的所有key 17](#_Toc50315998)

[7.2.6 返回字典中的所有value 18](#_Toc50315999)

[7.2.7 返回一个包含所有元素(键值对)的列表 18](#_Toc50316000)

[8. 容器类型-集合(set) 18](#_Toc50316001)

[9. 总结—TBD 19](#_Toc50316002)

# 序言

## 章节简介

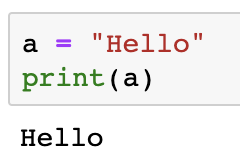
Python程序中保存的所有数据都是围绕对象这个概念来构建的。对象包括一些基本的数据类型，如数字、字符串、列表和字典。同时也可以通过类的形式创建用户定义的对象。本章介绍Python对象模型的内部原理，并介绍主要的内置数据类型。本章主要内容有：

1. 常量和变量
2. 基础类型：数值类型
3. 基础类型：字符串类型
4. 容器类型：列表list
5. 容器类型：元组tuple
6. 容器类型：字典dict
7. 容器类型：集合set
8. 变量类型转换

# 常量与变量

## 2.1 变量是什么

在前面的学习中，我们实际已经使用到了变量。下面的代码中，我们声明了一个变量a并且赋值为字符串”Hello”，在代码的第二行，我们打印出a的值。



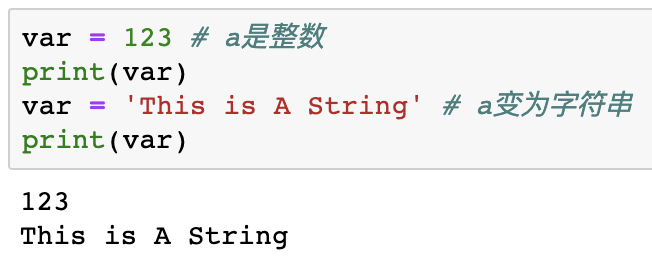
变量的概念基本上和初中代数方程里的变量是一致的，只是在计算机程序中，变量不仅可以是数字，还可以是任意类型。我们在程序中用一个变量名来代表一个变量，变量名必须是大小写英文、数字和\_的组合，且不能用数字开头，关于详细的命名规范我们在上一小节有过详细的介绍。下面的代码声明了变量a，它是一个整数：



而下面我们又声明了变量str,它的值是一个字符串：

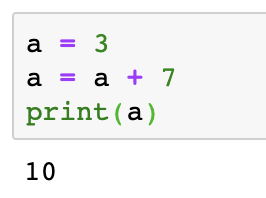


在Python中，等号=是赋值语句，可以把任意数据类型赋值给变量，同一个变量可以反复赋值，而且可以是不同类型的变量，例如下面我们声明了一个变量var,它的值开始是一个整数123，代码的第三行我们又把字符串”This is A String”赋值给var，现在它的值变成了一个字符串。



跟其他静态类型的高级语言(比如Java)相比，这种变量类型随时变化的特性，我们又把Python叫做动态语言。

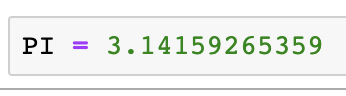
我们再来解释下赋值运算符=，它在Python中表示的意义是将等号右边的值赋值给左边的变量，而不是在数学中我们熟悉的“等于“的意思。在下面的代码中，实际上Python先计算等号右边，将变量a的值加7，再将计算得到的值重新赋值给a：



关于变量的最后，我们再来多了解一下当我们给一个变量赋值的时候，Python都在计算机内存中做了哪些事情：当我们编写一段程序a=888时，Python首先在内存中创建了一个888的数字，然后在内存中创建了一个变量a，并将a指向888。

## 2.2 常量是什么

所谓常量就是不能变的变量，比如常用的数学常数π就是一个常量。在Python中，通常用全部大写的变量名表示常量：



但事实上PI仍然是一个变量，Python根本没有任何机制保证PI不会被改变。所以，用全部大写的变量名表示常量只是一个习惯上的用法，如果你一定要改变变量PI的值，也没人能拦住你。

# 基础类型-数值类型

本小节介绍可用于表示程序中会使用的大多数数据类型。Python使用5中数字类型：布尔型、整数、长整数、浮点数以及负数。我们会介绍布尔型、整数、浮点数以及空值类型。

## 布尔型

布尔值包括True和False两个值，分别映射为数值1和0。一个布尔值只有True、False两种值。在Python中，我们使用True和False表示布尔值，比如像下面：

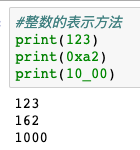


布尔类型的变量可以用来进行与或非的逻辑运算：



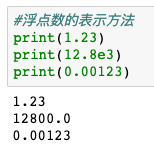
## 整型

Python3中可以处理任意大小的整数，在程序里表示一个整数的方式和我们平时一模一样，比如1，100，-123(负数)。计算机里因为是二进制保存数据，所以有时也会看到一十六进制表示的整数(十六进制里用0x表示前缀，值用0-9和a-f表示)，比如0xa2表示的整数就是162(2\*1+10\*16)。有时为了增加代码的可读性，我们也在数字中加入分隔符\_方便我们阅读，比如代码中写成10\_000\_000\_000和10000000000是完全一样的。



## 浮点型

浮点数也就是小数，之所以称为浮点数，是因为按照科学记数法表示时，一个浮点数的小数点位置是可变的。比如，1.23X10^9(1.23乘以10的九次方)和12.3X10^8是完全相等的。浮点数可以用数学写法，如1.23，3.14，-9.01，等等。但是对于很大或很小的浮点数，就必须用科学计数法表示，把10用e替代，1.23x109就是1.23e9，或者12.3e8，0.000012可以写成1.2e-5，等等。整数和浮点数在计算机内部存储的方式是不同的，整数运算永远是精确的（除法难道也是精确的？是的！），而浮点数运算则可能会有四舍五入的误差。



## 空值类型

None类型表示一个null对象(没有值的对象)。Python提供了一个null对象，在程序中表示为None。如果一个函数没有显式地返回值，则返回该空值对象。None经常用作可选参数的默认值，以便让函数检测调用者(关于函数和参数在后续章节详细介绍)是否为该参数实际传递了值。None没有任何属性，在布尔表达式中求值时为False。

注意None不能理解为0，0是有意义的，而None表示什么也没有，是一个特殊的空值。

# 基础类型-字符串类型

我们见到过单个字符’a’、’@’。由这些单个字符组成的变量类型就是字符串。在新的Python3中，字符串是以Unicode编码的(关于Unicode编码方式这里不做过多介绍)，也就是说Python的字符串支持多语言。因为字符串是Python语言中特别重要的概念(不仅仅是Python，在很多编程语言中字符串也非常重要)，我们详细的讲解一下字符串的用法。我们已经知道，单引号，双引号，包括三引号包围的字符组，就是字符串，例如：

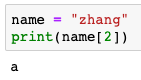


## 下标和切片

### 下标索引

下标在Python中的概念就是编号的意思，字符串、元组、列表都会经常到下标的概念，我们可以根据下标找到它们所对应的元素。就好像生活中你要准备去看电影，电影票上的座位号找到对应的位置．

我们现在考虑这样的一个问题，例如我们创建了一个字符串name=zhang，我现在想取到名为name字符串里面的a字符，如何去取呢？Python给我们提供了更为简便的方法，我们就可以下标来取出a字符：



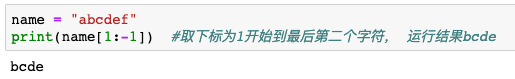
### 切片

我们可以利用下表索引取到字符串里面对应的一个元素，但如果想要截取一段元素就要用到切片。**切片**是指对操作的对象截取其中一部分的操作。字符串、列表、元组都支持切片操作。  
 切片的语法：[起始:结束:步长]

如果取出一部分，则可以在中括号[]中使用：



支持负数：



## 字符串的常见操作

假设有字符串mystr = “hello world python”,以下是常见的操作：

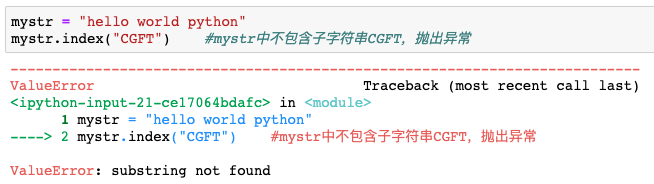
### find

检测str是否包含在字符串中，如果包含则返回开始的索引值，否则返回-1。例如：



### index

跟find()方法一样，只不过如果str不在mystr中会报一个异常(异常的概念会在后续章节介绍)。例如：



### count

返回str在mystr里面呢出现的次数，例如：



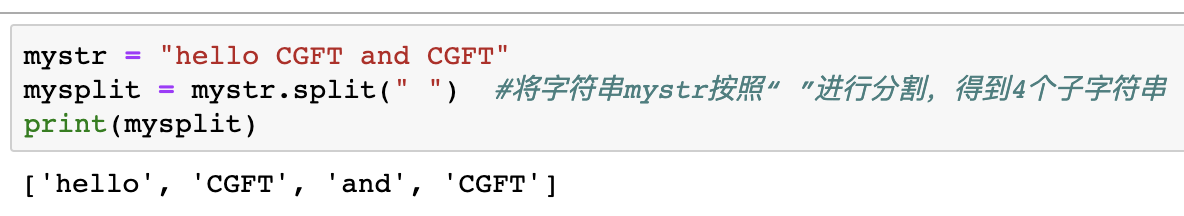
### replace

把mystr中的str1字符串替换为其他字符串，例如：



### split

将字符串按照指定字符或字符串进行分裂，例如：



### startwith

检查字符串是否以某字符或字符串开头，是则返回True,否则返回False。例如：



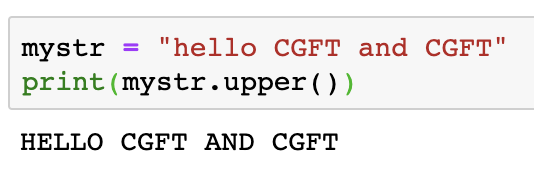
### endswith

检查字符串是否以某字符串结束，如果是返回True，否则返回False。



### upper

转换字符串中的字符为大写，例如：



### lstrip

删除字符串左边的空白字符，例如：



# 容器类型-列表list

## 列表简介

Python内置的一种数据类型是列表list。List是一种有序的集合，可以随时添加和删除其中的元素，写在方括号之间、用逗号分隔开的数值列表。列表内的项目不必全是相同的类型。例如：



## 列表的常见操作

### 列表的长度

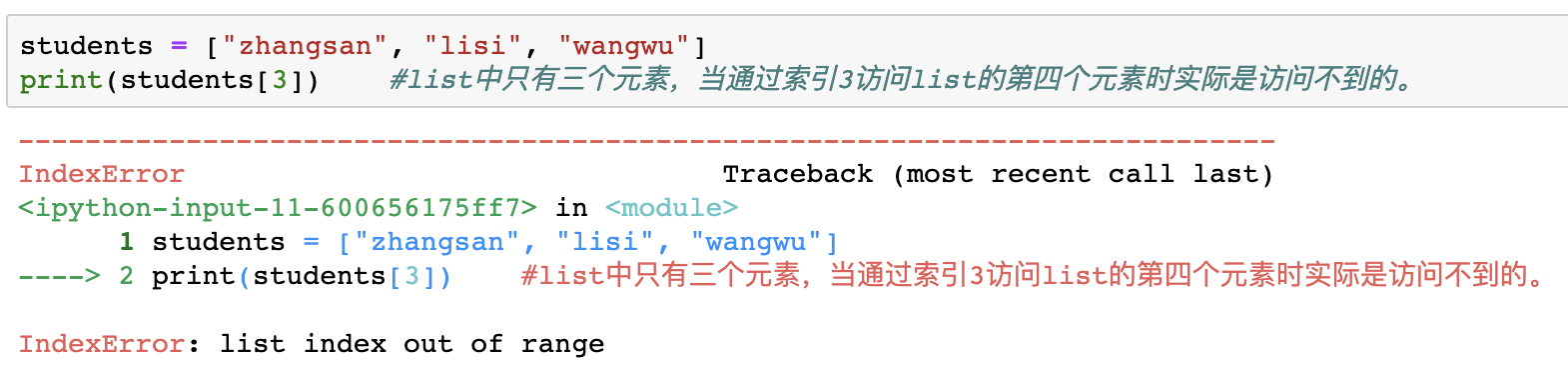


### 列表的访问

我们可以用索引来访问list中每一个位置的元素，注意索引是从0开始的：



注意：当索引超出了范围时，Python会报一个IndexError的错误。在编程时要确保通过索引访问列表元素时不要越界。记得最后一个元素的索引是len(students) – 1.



如果想获取list的最后一个元素，除了计算索引位置外，还可以使用-1直接获取最后一个元素：



### 列表的切片

切片：根据下标的范围获取一部分数据，像列表和之前介绍的字符串都可以使用切片。切片的格式为list[起始下标:结束下标:步长]。默认起始下标0，结束下标不包含，步长默认是1.



### 给列表添加元素

我们可以通过方法append, extend, insert为列表添加元素，例如：

通过append可以向列表添加元素：



通过extend方法可以将另一个集合中的元素全部添加到列表中：

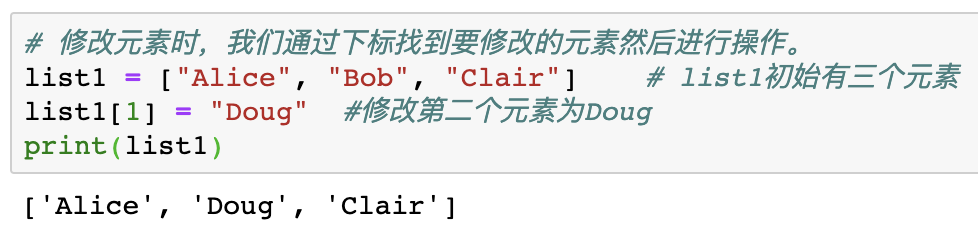


同样可以通过insert方法在指定位置插入新的元素：



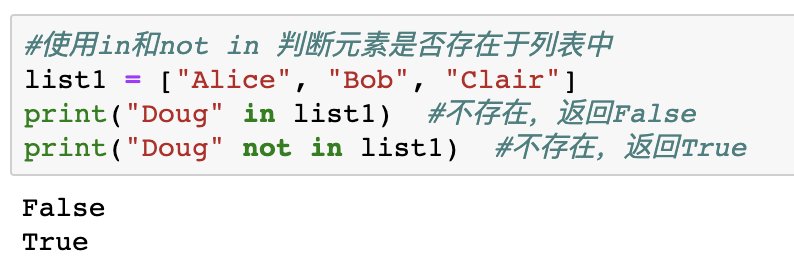
### 修改元素

修改元素时，我们通过下标找到要修改的元素然后进行操作。



### 查找元素

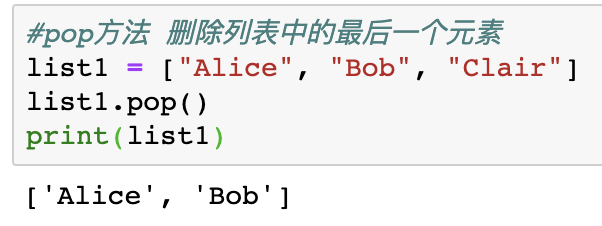
列表里的查找，就是看某个元素是否在列表中。Python的查找方式是运算符in，如果存在那么该运算返回True否则返回False；not in运算符用来判断元素是否不存在，如果不存在返回True否则返回False。



### 删除元素

删除列表中的元素通常由del,pop,remove三种方法，我们来分别介绍。

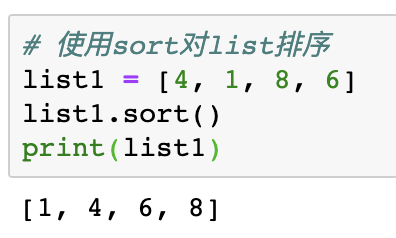






### 排序

sort方法可以将list按照特定顺序重新排列，默认为有小到大，参数reverse=True可以改为倒序。



### 列表遍历

我们也可以使用for循环来遍历列表里的所有元素。关于for循环会在后续章节详细介绍。



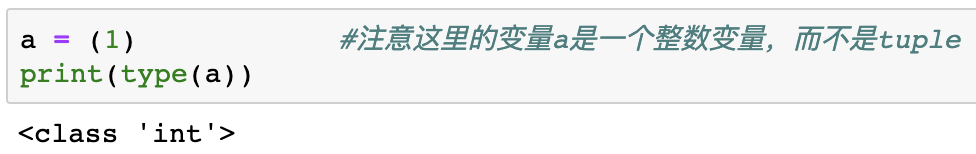
# 容器类型-元组tuple

另一种有序列表叫元组(tuple)。tuple和list非常类似，但是tuple一旦初始化就不能更改，比如同样是列出同学的名字。

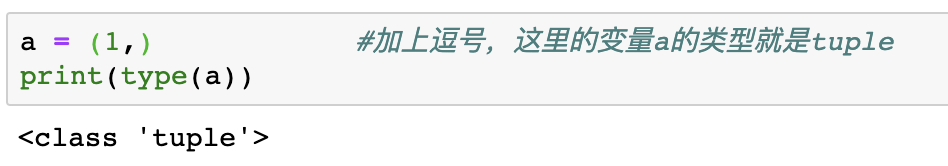


现在，tuple1这个tuple不能变了，它也没有append(), insert()这样的方法。其他获取元素的方法和list是一样的，我们可以正常的使用tuple1[0], tuple[-1]，但不能赋值成另外的元素。同学们可能会问，那不可变的集合有什么意义？因为tuple不可变，所以代码更安全。如果可能，程序中能用tuple或list的地方尽量使用tuple。

如何定义一个tuple？我们用括号表示一个tuple，括号内的所有元素用逗号分隔。注意，如果定义的是只有一个元素的tuple，如果我们这样定义：



定义的不是tuple，而是一个整形变量a！这是因为括号既可以表示tuple，又可以表示数学公式中的小括号(Python对被括号包含的表达式会作为一个整体处理)，这就产生了歧义。因此，Python规定，这种情况下，定义的是整形变量1。所以，只有一个元素的tuple定义时必须加一个逗号，来消除歧义：



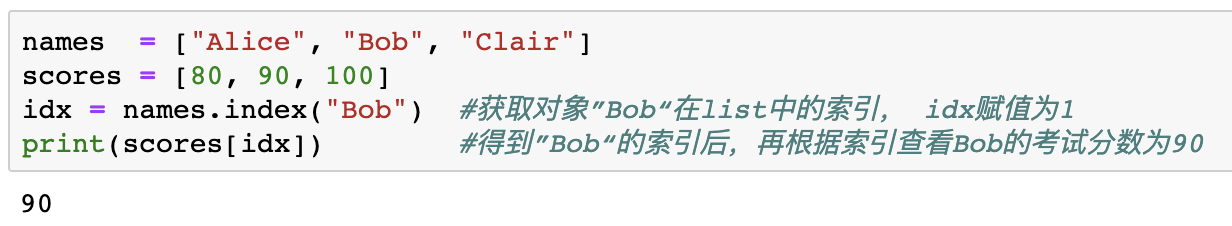
Python在显示只有一个元素的tuple时，也会加一个逗号，以免你误解为数学计算意义上的括号。

# 容器类型-字典dict

## 字典简介

字典是另一种可变容器模型，且可存储任意类型对象。

字典的每个键值对(key=>value)用冒号(:)分隔，每个对之间用逗号(,)分隔，整个字典包含在花括号{}中。因为我们使用key搜索查找键值对，所以字典里的key不能重复。举个例子，假设要根据同学的名字查找对应的成绩，如果用list实现，需要两个list:

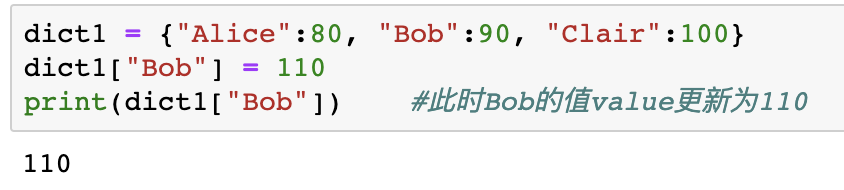


给定一个名字，要查找对应的成绩，就先要在names中找到对应的位置，再从scores取出对应的成绩，list越长，耗时越长。

如果用dict实现，只需要一个**”名字”=》”分数”**的对照表，直接根据名字查找成绩，无论这个表多大，查找名字并获取对应的分数都在这个集合里完成，查询速度相比list提高很多。用Python定义一个dict如下：



因为一个key只能对应一个value，所以，多次对一个key放入value，后面的值会把前面的值覆盖掉：



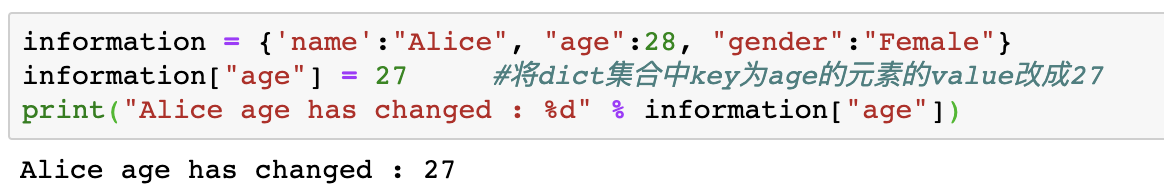
如果key不存在，dict就会报错：



## 字典的常见操作

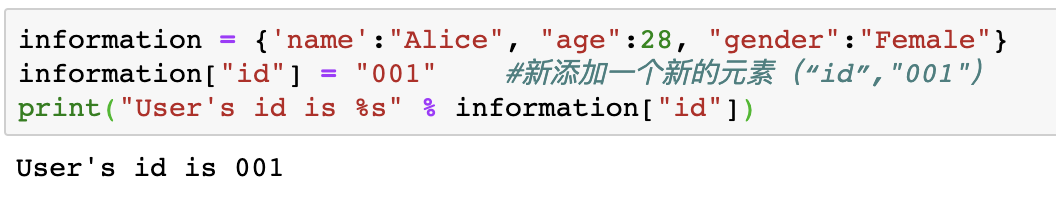
### 修改元素

字典中的元素的数据是可以修改的，只要通过key找到对应元素就可以对value进行修改。



### 添加元素

如果在使用变量名**[key] = value**时，这个键(key)在字典中不存在，那么就会在dict集合里新添加一个新的元素。



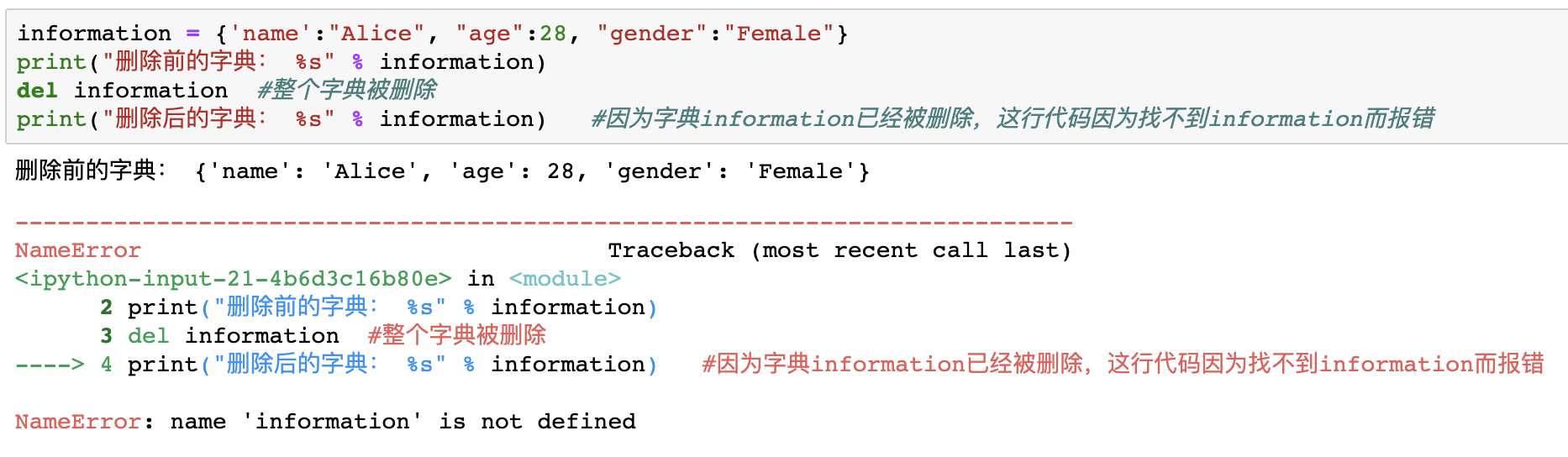
### 删除元素

对字典进行删除操作，有以下两种：**del()**和**clear()。**

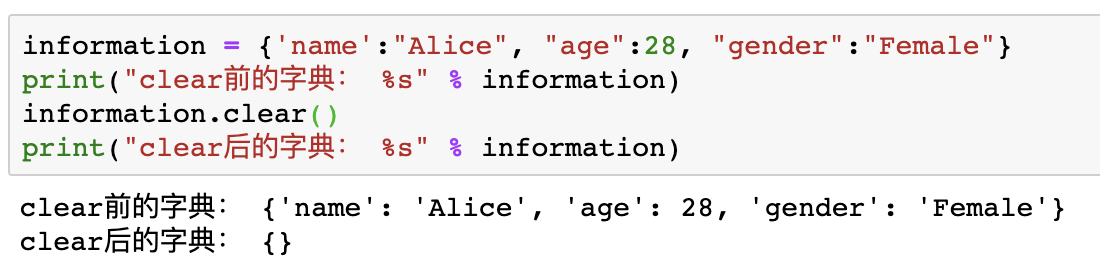
**del** 删除指定的元素：



**del** 删除整个字典：



**clear()**清空整个字典：



### 查看字典中有多少个元素

查看字典元素个数我们用方法len():



### 返回字典中的所有key

方法keys()返回一个包含所有key的列表：



### 返回字典中的所有value

方法values()返回字典的所有values：



### 返回一个包含所有元素(键值对)的列表



# 容器类型-集合(set)

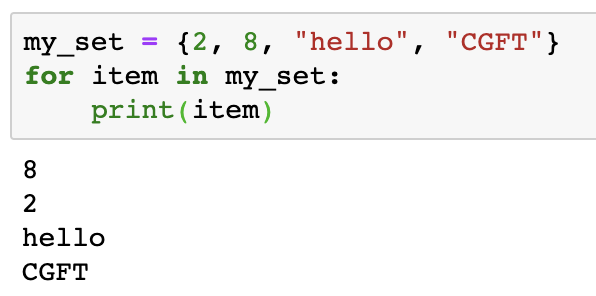
集合(set)是一个无序的不重复元素序列。

可以使用大括号{}或set()函数创建集合，注意：创建一个空集合必须用set()而不是{}，因为{}是用来创建空字典的。

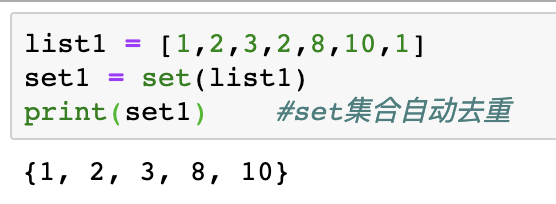
没有重复元素，set集合不支持通过索引返回指定位置元素：



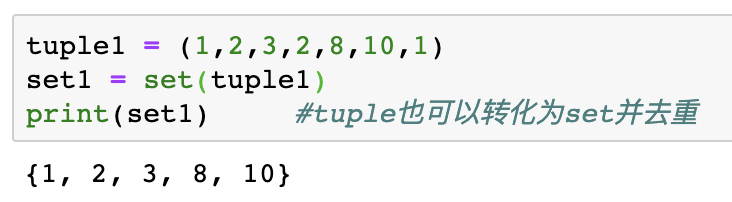
我们通过for循环遍历获取数据：



集合会对重复数据去重，下面的代码首先定义了一个list集合(包含重复数据)，当我们把列表(list)转换成集合set时，数据会被去重：



元组tuple也可以转换成set。list, tuple, set三者可以互相转化：



# 总结—TBD