**1、python中常用的数据结构有哪些？请简要介绍一下。**

数字：int、float、bool、complex

字符串：str

列表：list，用[]

元组：tuple，用()，就相当于不可修改的list

集合：set，用{}

字典：dict，用{key: value}

其中，列表，元组和字符串可以统一归为**序列类**，即这三种数据结构中的元素是有序的。比如，他们都有索引（下标）操作，还有切片、相加和长度(len)，最大值(max)，最小值(min)操作。这是他们的共同点。

补充：python中常见的数据结构可以统称为容器（container）。

序列（如列表和元组）、映射（如字典）以及集合（set）是三类主要的容器。

另外，关于这个问题，面试官很容易引出另一个问题：

**1-1、python中的哪些数据类型是可变的，哪些是不可变的？**

首先，可变/不可变是针对该对象所指向的内存中的值是否可变来判断的。

如可变类型的数据类型有列表和字典，还有集合。

不可变类型的数据类型有字符串，元组，数字。

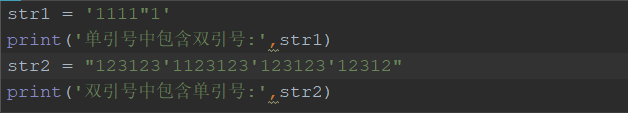
就举个最简单的数字的例子，python中有小整数池的概念，即[-5,256]范围内的整数，python解释器对他们做了特殊处理，都放在内存中的固定位置，不会因为你的操作而发生变化。

现在：a = 1 ，然后我们又重新对a赋值，a = 2，在重新赋值的过程中，整数1所对应的内存地址没有和数字的大小都没有发生变化，还在内存中的固定位置。整数2也是如此。变化的是a的指针（这里引用C中的概念）从指向数字1变成数字2。a对象指向的内存中的值没有发生变化，因此数字是不可变类型的数据类型。字符串，元组也是同理。

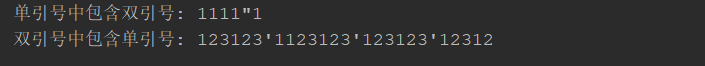
##### 2、简要描述python中单引号、双引号、三引号的区别。

* 首先，单引号和双引号在使用时基本上没有什么区别，唯一需要注意的是：

**当字符串中有单引号时，最好在外面使用双引号；当有双引号时，最好在外面使用单引号。**



**输出**



* 三引号一般不常用，除了用来做注释之外，还可以用来打印多行字符串。特殊用途，是可以打印多行字符串。



**输出**



* 如果单引号和双引号想实现以下的换行效果，需要加上换行符



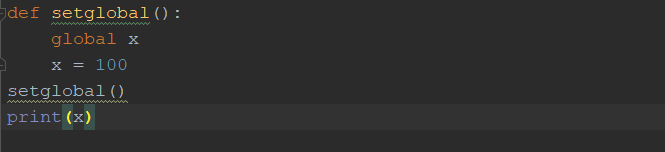
**输出**



**3、**[**如何在一个function里面设置一个全局变量**](https://www.cnblogs.com/apollo1616/articles/9785060.html)

如果要给全局变量在一个函数里赋值，必须使用global语句。global VarName的表达式会告诉Python，VarName是一个全局变量，这样Python就不会在局部命名空间里寻找这个变量了。

下面这个代码只有调用了函数,在打印才有效,否则没结果



输出

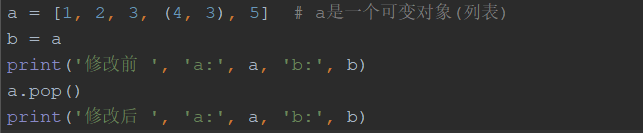


##### 4、python里面如何拷贝一个对象？（赋值、浅拷贝、深拷贝的区别）

在python中如何拷贝一个对象是需要根据具体的需求来定的。

1. 赋值：其实就是对象的引用。相当于C中的指针，修改了其中一个对象，另一个跟着改变。注意对于不可变对象而言，如果修改了其中一个对象，就相当于修改它的指针指向，另一个对象是不会跟着变化的。

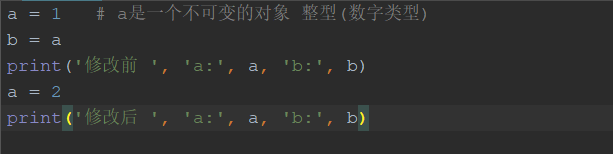
* 修改可变对象



输出



* 修改不可变对象

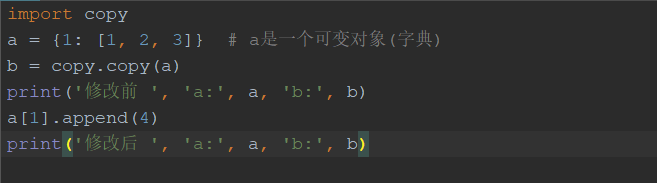


输出



1. 浅拷贝：拷贝父对象，但是不会拷贝父对象的子对象。（具体的方法有：b = copy.copy(a)，切片如b = a[1:4]）

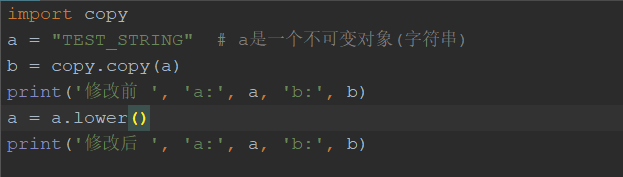
* 修改可变对象



输出



* 修改不可变对象

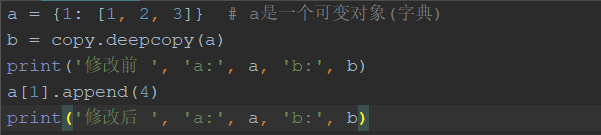


输出

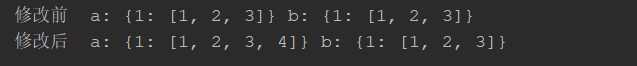


1. 深拷贝：完全拷贝了父对象和子对象（具体的方法有：b = copy.deepcopy(a)）

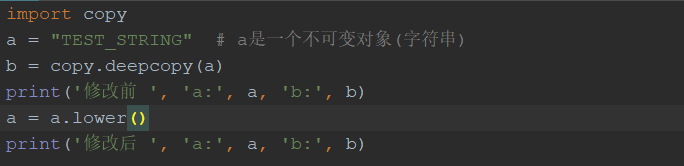
* 修改可变对象



输出



* 修改不可变对象

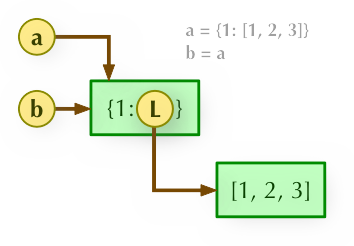


输出

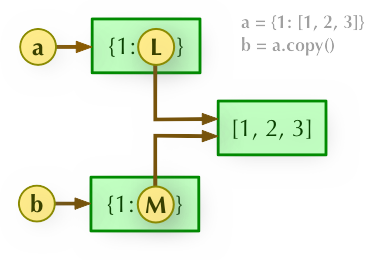


下面是图解：

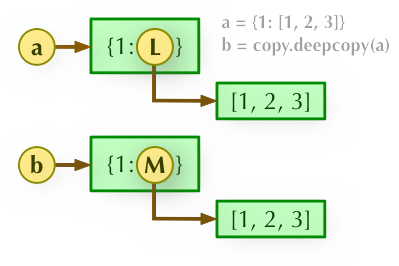
1、b = a: 赋值引用，a 和 b 都指向同一个对象。



2、b = a.copy(): 浅拷贝, a 和 b 是一个独立的对象，但他们的子对象还是指向统一对象（是引用）。



3、b = copy.deepcopy(a): 深度拷贝, a 和 b 完全拷贝了父对象及其子对象，两者是完全独立的。



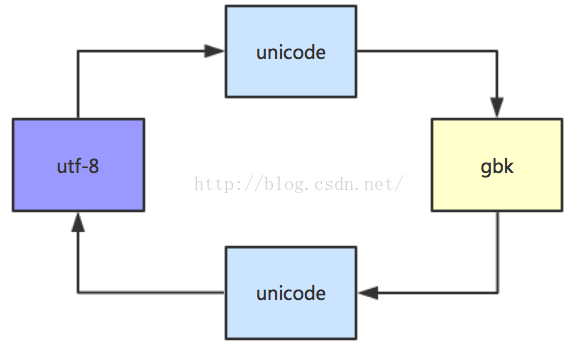
总结：

（1）当对象为不可变类型时，不论是赋值，浅拷贝还是深拷贝，那么改变其中一个值时，另一个都是不会跟着变化的。

（2）当对象为可变对象时，如果是赋值和浅拷贝，那么改变其中任意一个值，那么另一个会跟着发生变化的；如果是深拷贝，是不会跟着发生改变的。

##### 5、如果custname字符串的编码格式为uft-8,如何将custname的内容转化为gb18030的字符串？

**注意：**unicode编码是一种二进制编码，是转换编码的中间桥梁。比如需要将utf-8转换为gbk，那么就需要先转换为unicode（decode），再转为gbk（encode）。

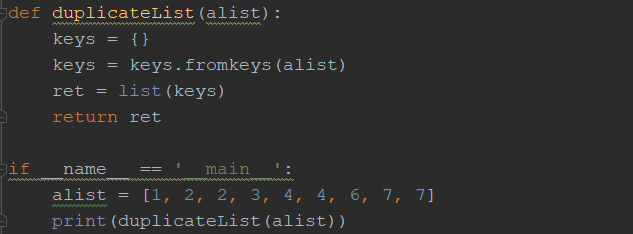


##### 6、请写出一段python代码实现删除list中的重复元素。

两种方法：

1. 利用字典的fromkeys来自动过滤重复值
2. 利用集合set的特性，元素是非重复的
3. 利用遍历去重的传统方式

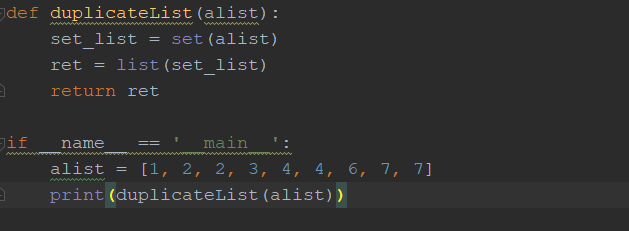
* 方法一：



输出



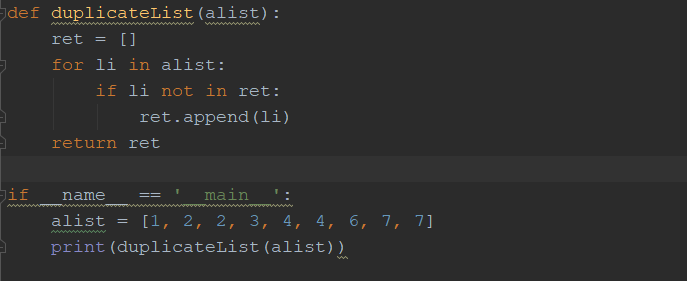
* 方法二：



输出



* 方法三：



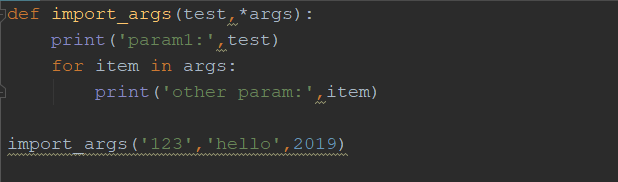
输出



##### 7、这两个参数是什么意思？*args和*kwargs。

首先，我想说的是\*args和\*\*kwargs并不是必须这样写，只有前面的\*和\*\*才是必须的。你可以写成\*var和\*\*vars。而写成\*args和\*\*kwargs只是约定俗称的一个命名规定。

\*args和\*\*kwargs主要用于函数定义，你可以将不定量的参数传递给一个函数。其中，**\*args** 是用来发送一个**非键值对**的可变数量的参数列表给一个函数；**\*\*kwargs** 允许你将不定长度的**键值对**, 作为参数传递给一个函数。 如果你想要在一个函数里处理**带名字的参数**, 你应该使用\*\*kwargs。

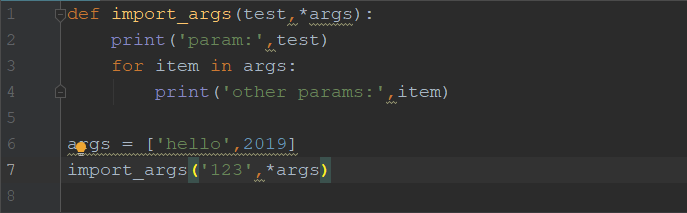


输出



这里传递了3个参数，按位置传参，'123'为test传参，'hello'和2019为\*args传参，这里传了2个参数。

注意，看下面的\*args的另一种用法：用来解压数据。



输出



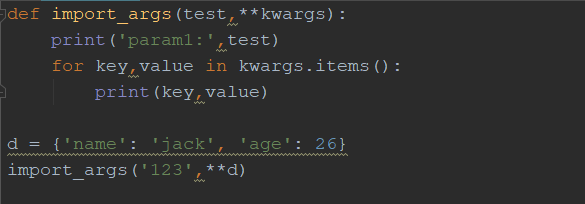
这段代码和上面的效果是一样的，但是这里第7行的\*args和第1行的\*args可是不一样的。

**第1行是表示函数可以接受不定数量的非键值对的参数，用来传参使用的。**

**第7行是用来解压列表['hello', '2019']的每一项数据的，用来解压参数的。**

这是**\*args的两种用法**，也可说是**\*的两种用法**，因为args是可变的。

接下来说说\*\*kwargs



输出



\*\*kwargs用来传递带键值对的参数，而\*\*也是用来解压字典容器内的参数。

总结：**\*args和\*\*kwargs都是用于函数中传递参数的，\*args传递的是非键值对的参数，\*\*kwargs传递的是带键值对的参数，如果还有普通参数需要传递，那么应该先传递普通的参数。**

##### 8、（1）统计如下list单词及其出现的次数。

a=['apple', 'banana', 'apple', 'tomato', 'orange', 'apple', 'banana','watermeton']

方法一