**一：变量、内存理解**

**变量**：用来标识(identify)一块内存区域。为了方便表示内存，我们操作变量实质上是在操作变量指向的那块内存单元。编译器负责分配。我们可以使用Python内建函数id()来获取变量的地址

**变量名**：是一个标识符(dientify)，用来代之一块内存空间，使用这个变量名，我们可以很方便的操作这块内存区域。

**内存：**内存是我们电脑硬件，用来存放数据，形象的理解就是内存有一个一个的小格子组成，每个格子的大小是一个字节，每个格子可以存放一个字节大小的数据。我们如何才能知道，数据存放在哪些格子中，那就得靠地址，地址类似于楼房的门牌号，是内存的标识符。

[回到顶部](https://www.cnblogs.com/lilz/p/9410319.html#_labelTop)

**二：id（）**

id(object)函数是返回对象object在其生命周期内位于内存中的地址，id函数的参数类型是一个对象。

[回到顶部](https://www.cnblogs.com/lilz/p/9410319.html#_labelTop)

**三：is 和 ==是什么**

在 Python 中一切都是对象，毫无例外整数也是对象，对象之间比较是否相等可以用==，也可以用is。==和is操作的区别是：

|  |  |
| --- | --- |
| **Is** | **比较的是两个对象的id值是否相等，也就是比较俩对象是否为同一个实例对象，是否指向同一个内存地址。** |
| **==** | **比较的是两个对象的内容是否相等，默认会调用对象的\_\_eq\_\_()方法。** |

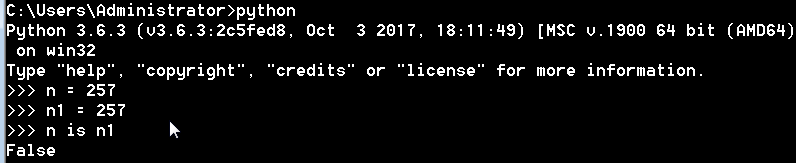
[回到顶部](https://www.cnblogs.com/lilz/p/9410319.html#_labelTop)

**四：数据池**

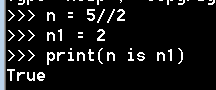
--->is比较id是否是同一个内存

**1：int**

----->范围[-5, 256]







对于257，a is b返回的竟然是False，结果可能在你的意料之中，也有可能出乎你的意料，但不管怎么，我们还是要刨根问底，找出问题的真相。

解惑一

出于对性能的考虑，Python内部做了很多的优化工作，对于整数对象，Python把一些频繁使用的整数对象缓存起来，保存到一个叫small\_ints的链表中，在Python的整个生命周期内，任何需要引用这些整数对象的地方，都不再重新创建新的对象，而是直接引用缓存中的对象。Python把这些可能频繁使用的整数对象规定在范围[-5, 256]之间的小对象放在small\_ints中，但凡是需要用些小整数时，就从这里面取，不再去临时创建新的对象。因为257不再小整数范围内，因此尽管a和b的值是一样，但是他们在Python内部却是以两个独立的对象存在的，各自为政，互不干涉。

弄明白第一个问题后，我们继续在Python交互式命令行中写一个函数，再来看下面这段代码：

片段一：

>>> c = 257

>>> def foo():

...     a = 257

...     b = 257

...     print a is b

...     print a is c

...

>>> foo()

True

False

呃，什么情况，是的，你没看错，片段一中的这段代码 a、b 值都是257的情况下，出现了a is b返回True，而a is c 返回的False，a、b、c的值都为257，为什么会出现不同的结果呢？这对于刚刚好不容易建立起来的认知就被彻底否决了吗，那这段代码中究竟发生了什么？难道解惑一中的结论是错误的吗？

解惑二

为了弄清楚这个问题，我们有必要先理解程序代码块的概念。Python程序由代码块构成，代码块作为程序的一个最小基本单位来执行。**一个模块文件、一个函数体、一个类、交互式命令中的单行代码都叫做一个代码块**。在上面这段代码中，由两个代码块构成，c = 257作为一个代码块，函数foo作为另外一个代码块。Python内部为了将性能进一步的提高，凡是在一个代码块中创建的整数对象，如果存在一个值与其相同的对象于该代码块中了，那么就直接引用，否则创建一个新的对象出来。**Python出于对性能的考虑，但凡是不可变对象，在同一个代码块中的对象，只有是值相同的对象，就不会重复创建，而是直接引用已经存在的对象。**因此，不仅是整数对象，还有字符串对象也遵循同样的原则。所以 a is b就理所当然的返回True了，而c和a不在同一个代码块中，因此在Python内部创建了两个值都是257的对象。

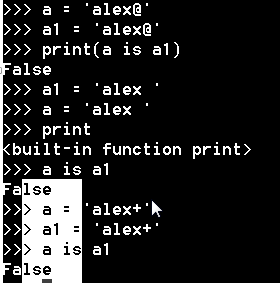
一番长篇大论之后，得出两点结论：

**1、小整数对象[-5,256]是全局解释器范围内被重复使用，永远不会被GC回收。**

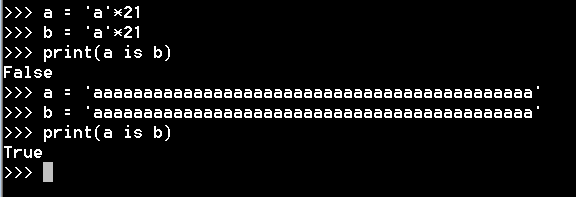
**2、同一个代码块中的不可变对象，只要值是相等的就不会重复创建新的对象。**

**2：str:**

不能有特殊字符串，全部由字母组成的字符串



字符串中单个\*20以内他们的内存地址一样，单个\*21以上，内存地址不一样



**3:py文件**

在一个py文件中，只要内容一样，内存就一样

或许你没有发现，你认真学习的样子总是最美的