链滴

Python3 查缺补漏：3、Python 中的“特权种族”是什么？

作者：zhaolixiang

原文链接：https://ld246.com/article/1568471036968

来源网站：链滴

前几天，某个学习群里有小伙伴问了一个关于id()的问题。事后，猫猫想起Python中一些常用对象的存地址是共用的，但是具体是哪些却忘了。于是，猫猫意识到这是我知识薄弱之处，有提升空间，便行了一番学习。

今天，猫猫把学习到的部分内容总结出来，分享给大家。阅读本文，大家可以学到如下内容：

1、对象的Id是什么？

2、内置id()函数是什么？

3、共用Id的内存分配策略？特权种族？

学习群里的一道问题

首先，看看小伙伴贴出的代码：

```
In [1]: a[1,2,3]
In [2]: id(a)
Out[2]: 2399283020744
In [3]: id(a.append(4))
Out[3]: 1417427824
In [4]: a.append(4)
In [5]: id(a)
Out[5]: 2399283020744
```

他的问题是：为何第二个id值（1417427824）不等于其它两个的id值（2399283020744）？还有这id值（1417427824）到底是谁的id？

现在公布答案：第二个id值（1417427824）是None的id，只要打印id(None)就能看出来；至于为是None的id，因为列表的append()方法返回值是None，而id(a.append(4))等价于取这个append作的返回值的id，也就是说id(a.append(4))等价于是id(None)。

对象Id与id()函数

python的对象有三要素：Id（identity，身份标识）、Type（类型标识）和Value（对象的值）。

其中，Value通常是一个对象能被直接“看到”的部分，而Id及Type则是相对底层的维度，无法直接看到”。举个例子（“>>>”表示输出结果）：

```
Object1=2018
Object2="2018"
# Object1的value是2018(数字)
# Object2的value是“2018”(字符串)
id(Object1) >>>2399282764784
id(Object2) >>>2399281922600
type(Object1) >>>int
type(Object2) >>>str
```

如上所述，我们建立了对象三要素与三个内置函数的联系：Id&id()、Type&type()、Value&str()。天，猫猫先跟大家一起来学习id()函数，今后再继续学习其它两个。

1、id()函数释义

id()是python内置的函数，它专门用于获取对象的内存地址，内存地址是一个整型数值，在该对象的命周期内是唯一且恒定的。语法：id([object])。

2、比较Id的两种方式

通常有两种比较对象Id的方式（is、id()比较），请看例子：

```
l1 = [1, 2, 3
]l2 = [1, 2, 3]
In [43]: l1 is l2
Out[43]: False
In [46]: id(l1)==id(l2)
Out[46]: False
# 两者Id不相等，因为：
In [44]: id(l1)
Out[44]: 2399279725576
In [45]: id(l2)
Out[45]: 2399282938056
```

is判断语句是判断两个对象的内存地址，也就等价于先取id()，再做数值比较。

3、不要与Value的比较方式混淆

Value的比较符号用双等号“==”，上例中比较l1和l2的Value要写成“l1 == l2”，明显两者的Valu是相等的。按照约定俗成的习惯，我们把Value值相等的两个对象称为“相等”，而把Id值相等的两对象称为“相同”。

所以，准确地说，上例的l1与l2相等，但是他们不相同，l1==l2，但l1 is not l2。

特权种族：共用内存的对象

每个对象被创建出来的时候，就会确定其Id标识，也就是给它分配内存地址。通常来说，新对象的内地址也是新的，会从未分配的可用地址中取。

但是，为了提高内存利用效率，对于一些常用的对象，如一些数值较小的数字对象、布尔值对象、Noe对象、较短的字符串对象等等，python采取共用对象内存的分配策略。

```
# 新分配内存地址的例子
ww=[1,2]
ee=[1,2]
id(ww)==id(ee)
>>>False

a=2018
b=2018
id(a)==id(b)
 >>>False

# 共用内存地址的例子
a=100
b=100
id(a)==id(b)
 >>>True
```

```
f1=True
f2=Trueid
(f1)==id(f2)
 >>>True

n1=None
n2=None
id(n1)==id(n2)
>>>True

s="python_cat"
t="python_cat"
id(s)==id(t)
>>>True
```

这就意味着，python中出现了“特权种族”，运行环境早早就为它们分配好了内存地址，一旦要创新的对象时，先去特权种族中查找，有Type和Value相等的对象，则新对象不分配新的内存空间，而指向已有对象。

“特权种族”的存在，使得我们不需要频繁创建这些对象，既能提高已分配内存的使用率，又减少了建对象、分配新内存的损耗。

对于共用内存地址的数字对象的取值范围，根据这篇文章《Python中神秘的-5到256》(链接见文末)python源码的分析，文中有如下结论：

Python中，对于整数对象，如果其值处于[-5,256]的闭区间内，则值相同的对象是同一个对象。

对于共用内存地址的字符串对象的取值范围，学习了几篇对python源码分析的文章后（链接见文末，猫猫总结出大致有以下结论：

Python中，字符串使用Intern机制实现内存地址共用，长度不超过20，且仅包括下划线、数字、字的字符串才会被intern；涉及字符串拼接时，编译期优化结果会与运行期计算结果不同。

```
# 编译对字符串拼接的影响
s1 = "hell"
s2 = "hello"
"hell" + "o" is s2
>>>True
s1 + "o" is s2
>>>False

# "hell" + "o"在编译时变成了"hello"，
# 而s1+"o"因为s1是一个变量，在运行时才拼接，所以没有被intern
```

参考阅读：

《Python中神秘的-5到256》

(https://zhuanlan.zhihu.com/p/33907983)

《Python中字符串的intern机制》

(https://www.cnblogs.com/greatfish/p/6045088.html)

《Python中字符串的intern机制》

(https://mrcuriosity.org/python-string-intern.html)

原文：

https://mp.weixin.qq.com/s?__biz=MzUyOTk2MTcwNg==&mid=2247483714&idx=1&sn=a36be77a5742106cbf9cbbef5ca7334&scene=21#wechat_redirect