Python 赋值、浅拷贝、深拷贝的区别？

2015年06月23日 12:20:26 阅读数：5235

http://songlee24.github.io/2014/08/15/python-FAQ-02/

在写Python过程中，经常会遇到对象的拷贝，如果不理解浅拷贝和深拷贝的概念，你的代码就可能出现一些问题。所以，在这里按个人的理解谈谈它们之间的区别。

一、赋值（assignment）

在《Python FAQ1》一文中，对赋值已经讲的很清楚了，关键要理解变量与对象的关系。

```
>>> a = [1, 2, 3]
>>> b = a
>>> print(id(a), id(b), sep='\n')
139701469405552
139701469405552
```

在Python中，用一个变量给另一个变量赋值，其实就是给当前内存中的对象增加一个“标签”而已。

如上例，通过使用内置函数 id() ，可以看出 a 和 b 指向内存中同一个对象。 `a is b` 会返回 True 。

二、浅拷贝（shallow copy）

注意：浅拷贝和深拷贝的不同仅仅是对组合对象来说，所谓的组合对象就是包含了其它对象的对象，如列表，类实例。而对于数字、字符串以及其它“原子”类型，没有拷贝一说，产生的都是原对象的引用。

所谓“浅拷贝”，是指创建一个新的对象，其内容是原对象中元素的引用。（拷贝组合对象，不拷贝子对象）

常见的浅拷贝有：切片操作、工厂函数、对象的copy()方法、copy模块中的copy函数。

```
>>> a = [1, 2, 3]
>>> b = list(a)
>>> print(id(a), id(b))           # a和b身份不同
140601785066200 140601784764968
>>> for x, y in zip(a, b):       # 但它们包含的子对象身份相同
...     print(id(x), id(y))
...
140601911441984 140601911441984
```

```
140601911442016 140601911442016
140601911442048 140601911442048
```

从上面可以明显的看出来，a 浅拷贝得到 b，a 和 b 指向内存中不同的 **list** 对象，但它们的元素却指向相同的 **int** 对象。这就是浅拷贝！

三、深拷贝（deep copy）

所谓“深拷贝”，是指创建一个新的对象，然后递归的拷贝原对象所包含的子对象。深拷贝出来的对象与原对象没有任何关联。

深拷贝只有一种方式：copy模块中的deepcopy函数。

```
>>> import copy
>>> a = [1, 2, 3]
>>> b = copy.deepcopy(a)
>>> print(id(a), id(b))
140601785065840 140601785066200
>>> for x, y in zip(a, b):
...     print(id(x), id(y))
...
140601911441984 140601911441984
140601911442016 140601911442016
140601911442048 140601911442048
```

看了上面的例子，有人可能会疑惑：

为什么使用了深拷贝，**a**和**b**中元素的**id**还是一样呢？

答：这是因为对于不可变对象，当需要一个新的对象时，python可能会返回已经存在的某个类型和值都一致的对象的引用。而且这种机制并不会影响 a 和 b 的相互独立性，因为当两个元素指向同一个不可变对象时，对其中一个赋值不会影响另外一个。

我们可以用一个包含可变对象的列表来确切地展示“浅拷贝”与“深拷贝”的区别：

```
>>> import copy
>>> a = [[1, 2],[5, 6], [8, 9]]
>>> b = copy.copy(a)               # 浅拷贝得到b
>>> c = copy.deepcopy(a)           # 深拷贝得到c
>>> print(id(a), id(b))            # a 和 b 不同
139832578518984 139832578335520
>>> for x, y in zip(a, b):         # a 和 b 的子对象相同
...     print(id(x), id(y))
...
139832578622816 139832578622816
139832578622672 139832578622672
```

```
139832578623104 139832578623104
>>> print(id(a), id(c))           # a 和 c 不同
139832578518984 139832578622456
>>> for x, y in zip(a, c):        # a 和 c 的子对象也不同
...     print(id(x), id(y))
...
139832578622816 139832578621520
139832578622672 139832578518912
139832578623104 139832578623392
```

从这个例子中可以清晰地看出浅拷贝与深拷贝地区别。

总结：

1、赋值：简单地拷贝对象的引用，两个对象的id相同。

2、浅拷贝：创建一个新的组合对象，这个新对象与原对象共享内存中的子对象。

3、深拷贝：创建一个新的组合对象，同时递归地拷贝所有子对象，新的组合对象与原对象没有任何关联。虽然实际上会共享不可变的子对象，但不影响它们的相互独立性。

浅拷贝和深拷贝的不同仅仅是对组合对象来说，所谓的组合对象就是包含了其它对象的对象，如列表，类实例。而对于数字、字符串以及其它“原子”类型，没有拷贝一说，产生的都是原对象的引用。