链滴

python 测试开发面试之深浅拷贝

作者：zyjImmortal
原文链接：https://ld246.com/article/1583206215258
来源网站：链滴

先来道题热热身

```
a = ('a', 'b','c')
c = copy.copy(a)
d = copy.deepcopy(a)

if c == d:
    print("c和d的值相等")
if id(c) == id(d):
   print("c和d的地址相等")
```

想想最后打印的是什么?

什么是深拷贝和浅拷贝

深拷贝，就是在对某个对象进行拷贝的时候，如果这个对象持有其他对象的引用，在拷贝的时候会将拷贝的对象以及引用的对象，一起拷贝。

而浅拷贝只拷贝当前对象和持有的索引，不拷贝索引指向的对象。举个例子说明一下，比如当前有个表a = [1,2,3], b = [3,4,a],[3,4,a]对象持有了[1,2,3]对象的引用，在对b进行深拷贝的时候，会将a对一起拷贝一份，而浅拷贝的时候则不会。

```
a = [1,2,3]

b = [4,5,6,a]
```

对b进行浅拷贝

```
c = copy.copy(b)
```

这个时候对a对象进行修改，会影响c

```
a.append(8)
 c
[4, 5, 6, [1, 2, 3, 8]]
```

对b进行深拷贝之后，再对a进行修改，则不会影响到d

```
d = copy.deepcopy(b)

d
[4, 5, 6, [1, 2, 3, 8]]

a.append(66)

 d
 [4, 5, 6, [1, 2, 3, 8]
```

当深浅拷贝遇到可变与不可变对象时会发生什么

上面用列表这种可变数据结构举例，再来看一下元组这种不可变结构，在进行深浅拷贝时定的现象。

```
a = (1,2,3)
b = copy.copy(a)
c = copy.deepcopy(a)
print(id(a))
print(id(b))
print(id(c))

输出:
4502776896
4502776896
4502776896
```

从结果中发现，a、b、c的内存地址大都是一样，所以在对不可变对象进行拷贝的时候，无论是浅拷还是深拷贝，都没有重新在内存中开辟新的地址，都只是对原对象增加了一个引用。

那如果不可变对象汇总包含有对可变对象的引用又会是怎么样呢?

```
a1 = [1,2,3]
a = (1,2,3, a1)
b = copy.copy(a)
c = copy.deepcopy(a)
print(id(a))
print(id(b))
print(id(c))

输出:
4502730288
4502730288
4503232240
```

b是浅拷贝生成的对象和原对象a的内存地址是一样对的，c是深拷贝生成的对象，发现内存地址和原象a的地址是不一样的。

也就是说如果不可变对象中持有了可变对象的引用，在进行深拷贝的时候会在内存中开辟新的地址存

对象。

回到文章开头提出的问题，是对不可变对象进行拷贝，而且不可变对象中并没有持有可变对象的引用所以两个print语句都会执行。