## [面试题-python 浅拷贝和深拷贝（copy模块）](https://www.cnblogs.com/yoyoketang/p/14449962.html)

**前言**

面试的时候经常会问到深拷贝和浅拷贝，那么python的深拷贝和浅拷贝有什么区别呢？

**思考题**

先来看 2 个简单的案例, 对元素 a/aa 重新赋值一个新的变量 b/bb 后，改变原来 a/aa 的值，看会不会影响新的变量 b/bb 的值

# 1.str

a = "hello"

b = a

a = "world"

print('a: {}'.format(a))

print('b: {}'.format(b))

# 2.list

aa = [1, 2, 3]

bb = aa

aa.append(4)

print('aa: {}'.format(aa))

print('bb: {}'.format(bb))

运行结果

a: world

b: hello

aa: [1, 2, 3, 4]

bb: [1, 2, 3, 4]

这是个很有趣的事情，字符串重新赋值给b后，改变原来a的值，b不会跟着变。  
但是list重新赋值给bb后，改变aa的值，bb的值也跟着变了。  
这里有个知识点：在python中，都是将“对象的引用(内存地址)”赋值给变量的。其次，在python中有6个标准数据类型，他们分为可变和不可变两类。

**可变和不可变对象**

在python中有6个标准数据类型，他们分为可变和不可变两类。

* 不可变类型：Number（数字）String（字符串）Tuple（元组）
* 可变类型：List（列表）Dictionary（字典）Set（集合）

可变对象和不可变对象的内存地址可以通过id函数获取

* 可变对象：可变对象可以在其 id() 保持固定的情况下改变其取值；
* 不可变对象：具有固定值的对象。不可变对象包括数字、字符串和元组。这样的对象不能被改变。如果必须存储一个不同的值，则必须创建新的对象。
* id(object)： 函数用于获取对象的内存地址，函数返回对象的唯一标识符，标识符是一个整数。

字符串和数字都是不可变类型，不同变量赋值一样，通过id获取的内存地址是一样的

# 作者-上海悠悠 QQ交流群:717225969

# blog地址 https://www.cnblogs.com/yoyoketang/

a = "abc"

b = "abc"

print(id(a))

print(id(b))

print(a is b)

c = 100

d = 100

print(id(c))

print(id(d))

print(c is d)

运行结果

1557212603592

1557212603592

True

1561032832

1561032832

True

list、dict 和 set集合是可变类型，虽然值一样，但是id获取的内存地址不一样

# 作者-上海悠悠 QQ交流群:717225969

# blog地址 https://www.cnblogs.com/yoyoketang/

a = {"key": "123"}

b = {"key": "123"}

print(id(a))

print(id(b))

print(a is b)

print(a == b)

c = [1, 2, 3]

d = [1, 2, 3]

print(id(c))

print(id(d))

print(c is d)

print(c == d)

运行结果

1638920310144

1638920310216

False

True

1638921292360

1638921292680

False

True

现在知道了id函数获取内存地址，我们说的深拷贝和浅拷贝是针对可变对象:list、dict 和 set集合

**copy模块**

python 中的深拷贝和浅拷贝使用 copy 模块

浅拷贝 A shallow copy constructs a new compound object and then (to the extent possible) inserts references into it to the objects found in the original.

上面这段话是官方文档上的描述，有2个含义：

* 1.浅拷贝会创建一个新的容器对象(compound object)
* 2.对于对象中的元素，浅拷贝就只会使用原始元素的引用（内存地址）

常见的浅拷贝操作有：

* 使用切片操作[:]
* 使用工厂函数（如list/dict/set）
* copy模块的copy()方法

深拷贝 A deep copy constructs a new compound object and then, recursively, inserts copies into it of the objects found in the original.

上面这段话是官方文档上的描述，也是有2个含义：

* 1.深拷贝和浅拷贝一样，都会创建一个新的容器对象(compound object)
* 2.和浅拷贝的不同点在于，深拷贝对于对象中的元素，深拷贝都会重新生成一个新的对象

**浅拷贝**

浅拷贝使用 copy 模块的 copy 方法

# 作者-上海悠悠 QQ交流群:717225969

# blog地址 https://www.cnblogs.com/yoyoketang/

import copy

a = [1, "hello", [2, 3], {"key": "123"}]

b = copy.copy(a)

print(id(a)) # 外面容器拷贝了，所以a和b的id不一样

print(id(b))

# a和b容器里面的元素对象id

print(id(a[2]))

print(id(b[2]))

运行结果

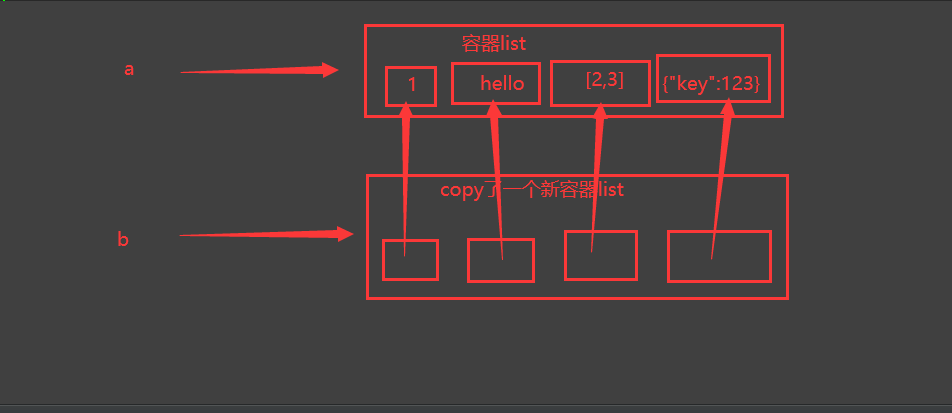
1340977220424

1340977221576

1340977220168

1340977220168

浅拷贝是拷贝了list外面一层的， 创建一个新的容器对象(compound object)，所以a和b的id是不一样的  
对于容器里面的元素对象，浅拷贝就只会使用原始元素的引用（内存地址），所以可以看到子元素的内存地址还是一样的



如果改变a里面的不可变对象数字和字符串，此时a和b的值就不一样了，但是b的后面没改变的元素还是指向a

# 改变a的 数字和字符串对象

a[0] = 2

# a 和b 的值不一样了

print(a)

print(b)

# 但是后面的元素还是指的a

print(id(a[2]))

print(id(b[2]))

运行结果

[2, 'hello', [2, 3], {'key': '123'}]

[1, 'hello', [2, 3], {'key': '123'}]

2488134044232

2488134044232

如果改变a里面的可变对象， 把[2, 3]里面的3改成 [2, 4]

# 改变a的 可变对象 [2, 4]

a[2][1] = 4

print(a)

print(b)

print(id(a[2]))

print(id(b[2]))

运行结果

[1, 'hello', [2, 4], {'key': '123'}]

[1, 'hello', [2, 4], {'key': '123'}]

2385125673544

2385125673544

此时b会随着a的改变而改变，这就是浅拷贝了

**深拷贝**

浅拷贝使用 copy 模块的 deepcopy 方法

import copy

# 作者-上海悠悠 QQ交流群:717225969

# blog地址 https://www.cnblogs.com/yoyoketang/

a = [1, "hello", [2, 3], {"key": "123"}]

b = copy.deepcopy(a)

print(id(a)) # 外面容器拷贝了，所以a和b的id不一样

print(id(b))

# a和b容器里面的元素对象id

print(id(a[2]))

print(id(b[2]))

# 改变a的 可变对象 [2, 4]

a[2][1] = 4

print(a)

print(b)

print(id(a[2]))

print(id(b[2]))

深拷贝和浅拷贝的不同点在于，深拷贝对于对象中的元素，深拷贝都会重新生成一个新的对象。  
所以不管a怎么变，都不会影响b的值

**赋值**

赋值跟浅拷贝 深拷贝是有区别的，可以看下面的示例

# 作者-上海悠悠 QQ交流群:717225969

# blog地址 https://www.cnblogs.com/yoyoketang/

a = [1, "hello", [2, 3], {"key": "123"}]

b = a

print(id(a))

print(id(b))

# a和b容器里面的元素对象id

print(id(a[2]))

print(id(b[2]))

a[0] = 2

print(a)

print(b)

运行结果

1992198687560

1992198687560

1992198687304

1992198687304

[2, 'hello', [2, 3], {'key': '123'}]

[2, 'hello', [2, 3], {'key': '123'}]

赋值语句并没有生成新的容器，跟浅拷贝的区别在于外面的容器也是指向的a的内存地址，并没有生成新的容器

参考博客资料<https://www.nowcoder.com/discuss/203654?type=2&order=0&pos=1232&page=0>  
参考博客资料<https://copyfuture.com/blogs-details/2020031720252559878eggumgw4iaj7c>