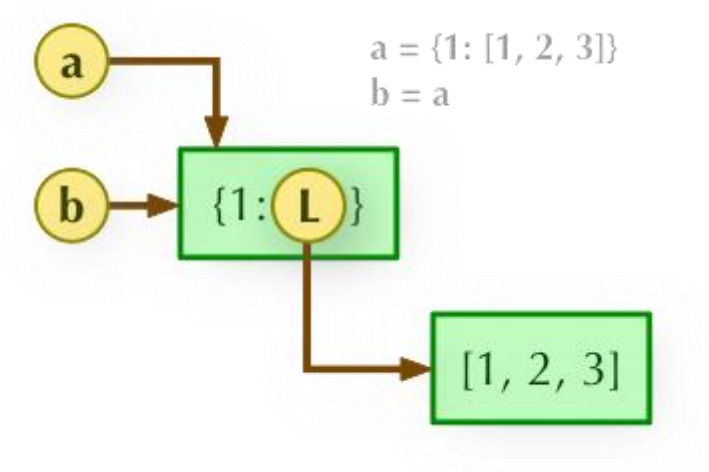
- **直接**赋值：其实就是对象的引用（别名）。
- **浅拷**贝(copy)：**拷**贝父对象，不会拷贝对象的内部的子对象。
- **深拷**贝(deepcopy)： copy **模**块的 deepcopy **方法**， **完全拷**贝了父对象及其子对象。

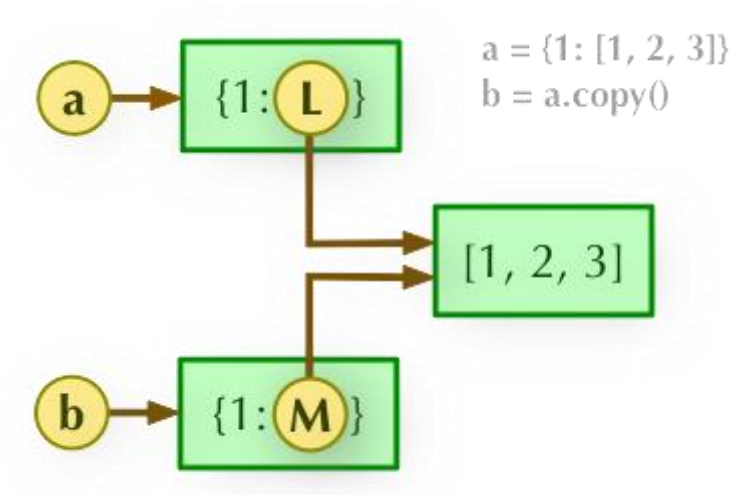
实例解析

```
a = {1: [1,2,3]}
```

1. b = a: 赋值引用，a **和** b **都指向同一个**对象，如下图：

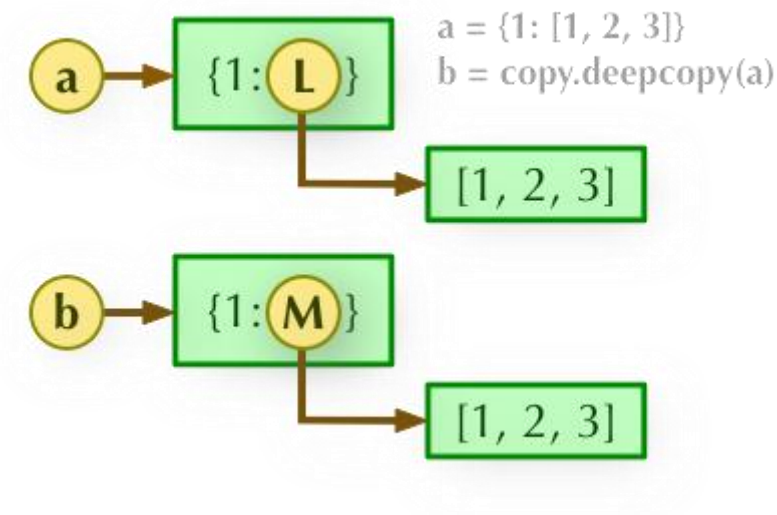


2. b = a.copy(): 浅拷贝, a **和** b **是一个独立的**对象，但他们的子对象还是指向统一对象（是引用），如下图：



3. b = copy.deepcopy(a): 需要导入copy**模**块，深度拷贝, a **和** b **完全拷**贝了父对象及

其子对象，两者是完全独立的，如下图：



举例：

```
>>>a = {1: [1,2,3]}
>>> b = a.copy()
>>> a, b
({1: [1, 2, 3]}, {1: [1, 2, 3]})
>>> a[1].append(4)
>>> a, b
({1: [1, 2, 3, 4]}, {1: [1, 2, 3, 4]})
```

```
>>>import copy
>>> c = copy.deepcopy(a)
>>> a, c
({1: [1, 2, 3, 4]}, {1: [1, 2, 3, 4]})
>>> a[1].append(5)
>>> a, c
({1: [1, 2, 3, 4, 5]}, {1: [1, 2, 3, 4]})
```

```
6
7
```

```
import copy

a = [1, 2, 3, 4, ['a', 'b']] #原始对象

b = a                        #赋值，传对象的引用
c = copy.copy(a)             #对象拷贝，浅拷贝
d = copy.deepcopy(a)         #对象拷贝，深拷贝

a.append(5)                  #修改对象a
a[4].append('c')             #修改对象a中的['a', 'b']数组对象

print( 'a = ', a )
print( 'b = ', b )
print( 'c = ', c )
print( 'd = ', d )

###输出###
a =  [1, 2, 3, 4, ['a', 'b', 'c'], 5]
b =  [1, 2, 3, 4, ['a', 'b', 'c'], 5]
c =  [1, 2, 3, 4, ['a', 'b', 'c']]
d =  [1, 2, 3, 4, ['a', 'b']]
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

```
19
20
```

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。