# 列表的复制

查看以下代码，说出打印的结果。

```
a = 12
b = a
b = 13
print(b)
print(a)

nums1 = [1, 5, 8, 9, 10, 12]
nums2 = nums1
nums2[0] = 100
print(nums2)
print(nums1)
```

思考：

1. 为什么修改了 nums2里的数据，nums1的数据也会改变?

> Python中的赋值运算都是引用(即内存地址)的传递。对于可变类型来说，修改原数据的值，会改变赋值对象的值。

1. 怎样nums1和nums2变成两个相互独立不受影响的列表?

> 使用列表的 copy 方法，或者 copy 模块就可以赋值一个列表。

## 列表的copy方法

使用列表的copy方法，可以直接将原来的列表进行复制，变成一个新的列表，这种复制方式是浅复制。

```
nums1 = [1, 5, 8, 9, 10, 12]
nums2 = nums1.copy()  # 调用列表的copy方法，可以复制出一个新的列表

nums2[0] = 100

# 修改新列表里的数据，不会影响到原有列表里的数据
print(nums2)
print(nums1)
```

## copy模块的使用

除了使用列表的copy方法以外，Python还提供了copy模块来复制一个对象。copy模块提供了浅复制和深复制

两种方式，它们的使用方式相同，但是执行的效果有一定的差异。

### 浅拷贝

浅拷贝是对于一个对象的顶层拷贝，通俗的理解是：拷贝了引用，并没有拷贝内容。

```
import copy

words1 = ['hello', 'good', ['yes', 'ok'], 'bad']

# 浅拷贝只会拷贝最外层的对象，里面的数据不会拷贝，而是直接指向
words2 = copy.copy(words1)

words2[0] = '你好'
words2[2][0] = 'no'

print(words1)  # ['hello', 'good', ['no', 'ok'], 'bad']
# wrods2 里的 yes 被修改成了 no
print(words2)  # ['你好', 'good', ['no', 'ok'], 'bad']
```

### 深拷贝

深拷贝是对于一个对象所有层次的递归拷贝。

```
import copy

words1 = ['hello', 'good', ['yes', 'ok'], 'bad']

# 深拷贝会将对象里的所有数据都进行拷贝
words2 = copy.deepcopy(words1)

words2[0] = '你好'
words2[2][0] = 'no'

print(words1)  # ['hello', 'good', ['yes', 'ok'], 'bad']
print(words2)  # ['你好', 'good', ['no', 'ok'], 'bad']
```

### 切片

列表和字符串一样，也支持切片，切片其实就是一种浅拷贝。

```
words1 = ['hello', 'good', ['yes', 'ok'], 'bad']
words2 = words1[:]
words2[0] = '你好'
words2[2][0] = 'no'
print(words1)  # ['hello', 'good', ['no', 'ok'], 'bad']
print(words2) # ['你好', 'good', ['no', 'ok'], 'bad']
```