**深度解读Python深拷贝与浅拷贝问题**

涉及到数据的传递，在数据传递使用过程中，可能会发生数据被修改的问题。为了防止数据被修改，就需要在传递一个副本，即使副本被修改，也不会影响原数据的使用。为了生成这个副本，就产生了拷贝。接下来说一下Python中的深拷贝与浅拷贝的问题。

概念普及：对象、可变类型、引用

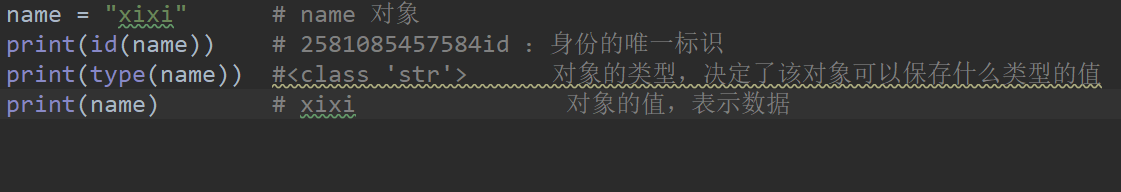
数据拷贝会涉及到Python中**对象、可变类型、引用**这3个概念，先来看看这几个概念，只有明白了他们才能更好的理解深拷贝与浅拷贝到底是怎么一回事。

**Python对象**

在Python中，对对象有一种很通俗的说法，万物皆对象。说的就是构造的任何数据类型都是一个对象，无论是数字，字符串，还是函数，甚至是模块，Python都对当做对象处理。

所有Python对象都拥有三个属性：身份、类型、值。

看一个简单的例子：



**可变与不可变对象**

在Python中，按更新对象的方式，可以将对象分为2大类：可变对象与不可变对象。

**可变对象：  列表（目前我们只学习了列表）、字典、集合**

所谓可变是指可变对象的值可变，身份（id）是不变的。

**不可变对象：数字、字符串、元组（马上要学到了）**

不可变对象就是对象的身份和值都不可变。新创建的对象被关联到原来的变量名，旧对象被丢弃，垃圾回收器会在适当的时机回收这些对象。



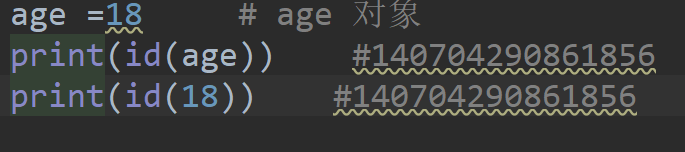
**引用**

在 Python 程序中，每个对象都会在内存中申请开辟一块空间来保存该对象，该对象在内存中所在位置的地址被称为引用。在开发程序时，所定义的变量名实际就对象的地址引用。

引用实际就是内存中的一个数字地址编号，在使用对象时，只要知道这个对象的地址，就可以操作这个对象，但是因为这个数字地址不方便在开发时使用和记忆，所以使用变量名的形式来代替对象的数字地址。 在 Python 中，变量就是地址的一种表示形式，并不开辟开辟存储空间。

就像 IP 地址，在访问网站时，实际都是通过 IP 地址来确定主机，而 IP 地址不方便记忆，所以使用域名来代替 IP 地址，在使用域名访问网站时，域名被解析成 IP 地址来使用。

通过一个例子来说明变量和变量指向的引用就是一个东西



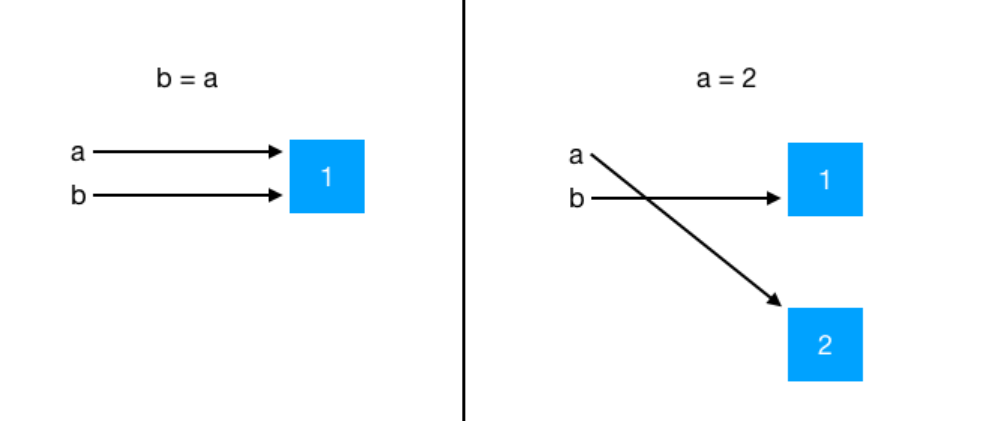
逐步深入：引用赋值

上边已经明白，引用就是对象在内存中的数字地址编号，变量就是方便对引用的表示而出现的，变量指向的就是此引用。赋值的本质就是让多个变量同时引用同一个对象的地址。  那么在对数据修改时会发生什么问题呢？

**不可变对象的引用赋值**

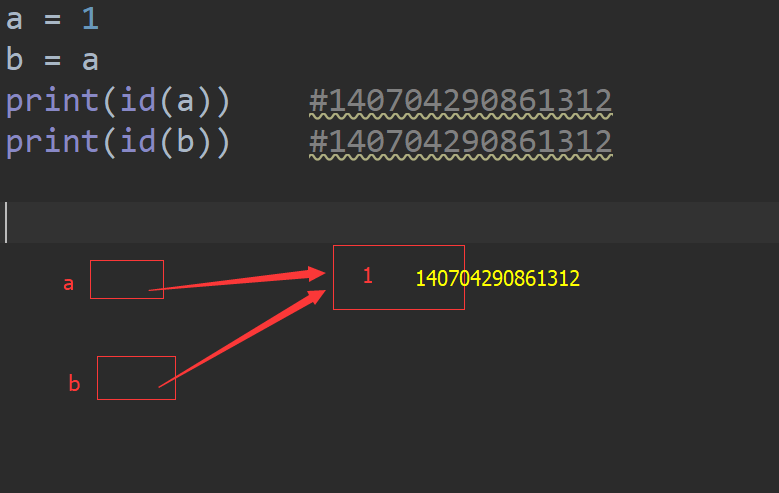
对不可变对象赋值，实际就是在内存中开辟一片空间指向新的对象，原不可变对象不会被修改。

**原理图如下：**



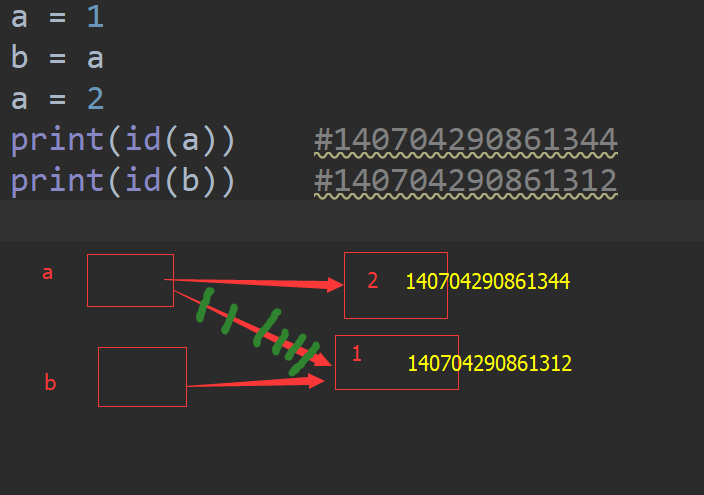
下面通过案例来理解一下：

a与b在内存中都是指向1的引用，所以a、b的引用是相同的



现在再给a重新赋值，看看会发生什么变化？

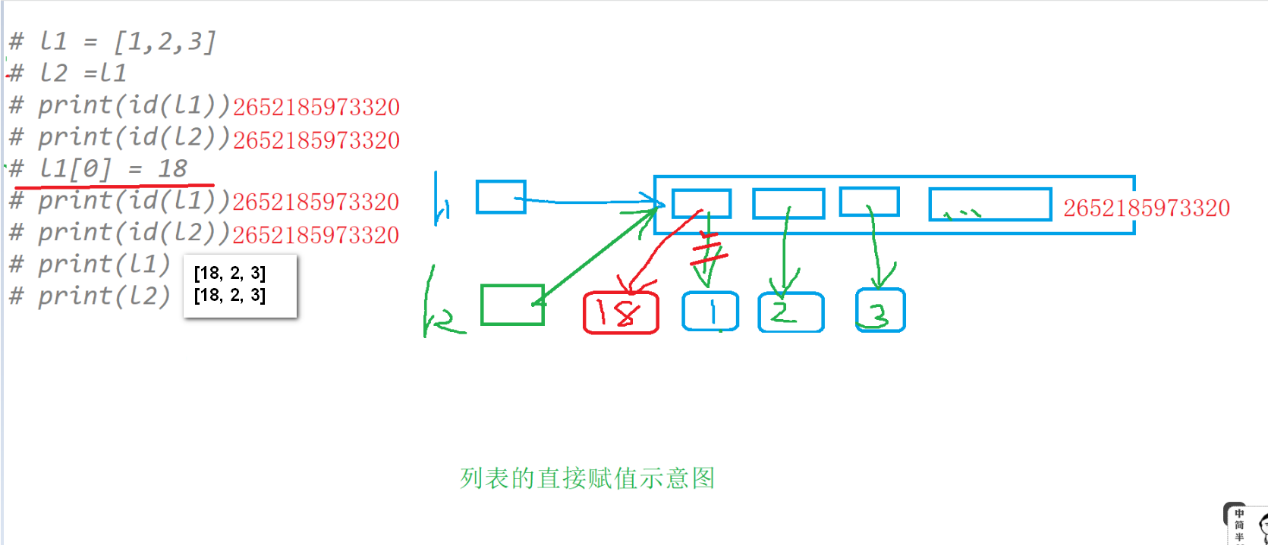
从下面不难看出：当给a 赋新的对象时，将指向现在的引用，不在指向旧的对象引用。



**可变对象的引用赋值**

可变对象保存的并不是真正的对象数据，而是对象的引用。当对可变对象进行赋值时，只是将可变对象中保存的引用指向了新的对象。

**原理图如下：**

当改变l1时，整个列表的引用会指新的对象，但是l1与l2都是指向保存的同一个列表的引用，所以引用地址不会变。

主旨详解：浅拷贝、深拷贝

经过前2部分的解读，大家对对象的引用赋值应该有了一个清晰的认识了。

**下面大家思考一个这样的问题**：Python中如何解决原始数据在函数传递之后不受影响了？

使用对象的拷贝或者深拷贝就可以愉快的解决了。

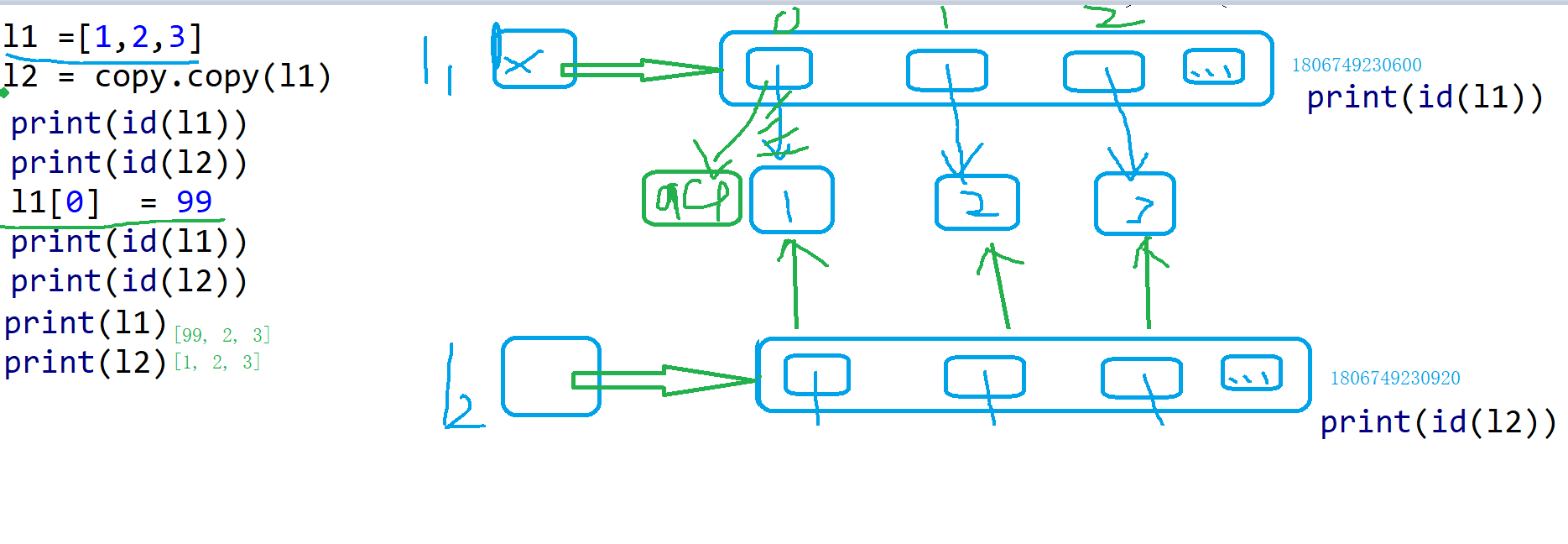
下面具体来看看Python中的浅拷贝与深拷贝是如何实现的。

**浅拷贝：**

为了解决函数传递后被修改的问题，就需要拷贝一份副本，将副本传递给函数使用，就算是副本被修改，也不会影响原始数据 。

**可变对象的拷贝**

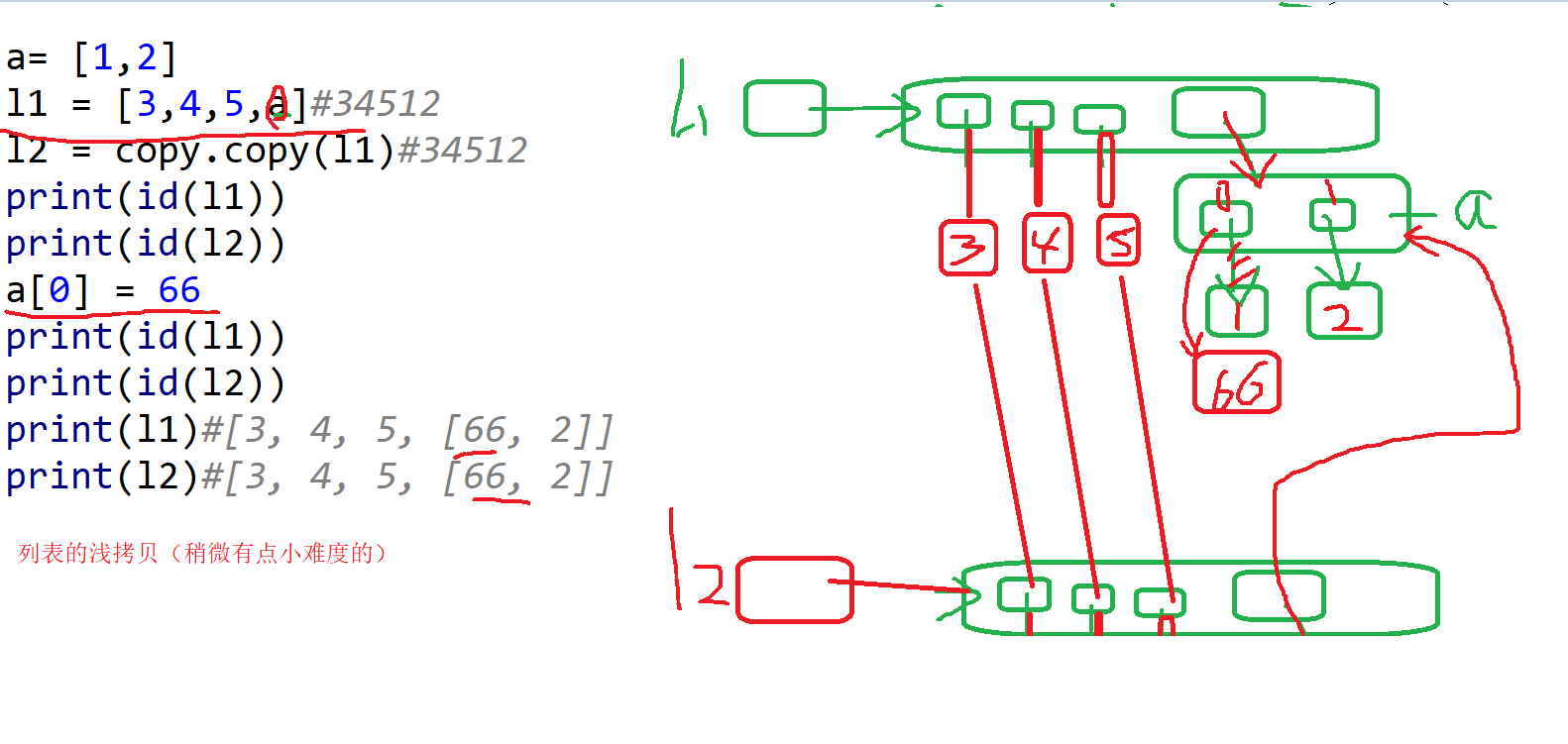
对于不可变对象的拷贝，对象的引用并没有发生变化，那么可变对象的拷贝会不会和不可变对象一样了？我们接着往下看。

通过下面这个实例可以看出：可变对象的拷贝，会在内存中开辟一个新的空间来保存拷贝的数据。当再改变之前的对象时，对拷贝之后的对象没有任何影响。

现在再回到刚才那个问题，是不是浅拷贝就可以解决原始数据在函数传递之后不变的问题了？下面看一个稍微复杂一点的数据结构。

通过下面这个实例可以发现：复杂对象在拷贝时，并没有解决数据在传递之后，数据改变的问题。 出现这种原因，是copy() 函数在拷贝对象时，只是将指定对象中的所有引用拷贝了一份，如果这些引用当中包含了一个可变对象的话，那么数据还是会被改变。 这种拷贝方式，称为浅拷贝。

**原理图如下：**



对于上边这种状况，Python还提供了另一种拷贝方式(深拷贝)来解决。

**深拷贝**

区别于浅拷贝只拷贝顶层引用，深拷贝会逐层进行拷贝，直到拷贝的所有引用都是不可变引用为止。

 特此奉上祁老师的大图

