**Python内存管理与垃圾回收**

————20190826：韵方

内存管理：

现在有一个int字段为1，我们定义如下两条字段：

a = 1

b = 1

那么，a和b指向的将是同一个地址，也就是字段1所在的地址，这就是python最基本的引用场景，不管是对于字段还是对象都是如此。也就是我们所说的对象池，python将各种各样的对象（“a”，1，2，list1）放入对象池中，我们需要的时候就去里面取，这样就不用自己创造了，以此达到速度快的目的。

为了便于理解引入如下图所示的代码和拓扑图：





我们已经了解了python的基本引用机制，那将话题再深入一点，python对于内存是如何管理的呢。Python将内存管理分成了4层，整体成金字塔形。最底层当然是c语言，也就是molloc和free函数，这两个函数是啥自行百度哈，就是动态分配内存函数和释放函数（此处建议学学看c语言基础和操作系统内存篇，没学过的补课哈）

然后第一层第二层是内存池，有python的接口函数PyMem\_Malloc函数实现，当对象的内容小于256k时，他会直接进行内存的分配，而大于时，才会进行内存的动态分配，因为频繁的调用c的molloc函数会造成效率的下降。这里要注意的是，动态分配内存时，它每一块也是256k，而且经内存池调用过的内存块，会回收到内存池，而不会释放掉，以便于下次继续调用。

最上面一层就是我们对python对象的操作。

由于引用计数法和以上这种内存管理机制的存在， python中也是存在深拷贝和浅拷贝这种类似于js的特性存在的。

数值，字符串，元组采用的是深拷贝，他们让A=B时，虽然还是指向同一个地址，但一旦A或者B改变了，他不会改变影响到另外一个，而是为其重新分配了内存，这在以下的代码中可以提现：



但像字典，列表等数据结构，他们采用的就是浅拷贝，他们改变一个就会改变另一个：



大概讲了下内存管理后我们来讲下python的垃圾回收机制，由于python时采取引用计数法的，那么垃圾回收的条件很容易的可以联想到，当计数为0时，将其回收。Python将垃圾回收分成了3代，我们将它记为0，1，2三代，某一代经历垃圾回收后依旧活着，那么就将它归入下一代中。所以，垃圾回收扫描0，1，2代的频率是不相同的，默认是扫描10次0代扫描1次1代，扫描10次1代扫描1次2代，这个频率可以通过python自带函数set.threshohold(700,10,10) 来进行调整。

采用这样的垃圾处理方式可以处理大多数的情况，只是有那么有一种情况，对象自成环时，计数永远不可能为0，自然不可能被回收，那么这种情况时，python是怎么做处理的呢。：

在python中，它将每个对象A都计数为gc\_ref\_i，如果该对象有引用，那么就将这个计数减去1，那如果不为环，那么一个链路到最后，必有一个节点不为0，那么对于不为0的对象和这些对象引用的对象，以及更下游的，均保留。而如果为环，那么就都是0，这样的自然就可以被回收了因为自成环的引用没有什么意义。