1. 小整数池

【-5， 256】

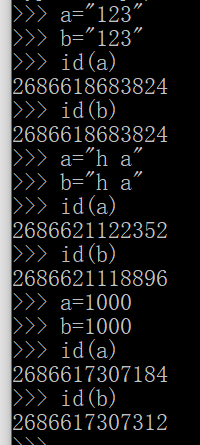
小整数池内的整数只存在一个

1. 大整数池

除了小整数池内的整数，其他数字都是用一个就创建一个，并放入大整数池。

1. 字符串

相同的字符串（不含空格或特殊字符）默认只保留一份，但若是有空格或特殊字符的字符串，就不是用同一个。



内存管理机制：

1. GC机制

垃圾回收机制

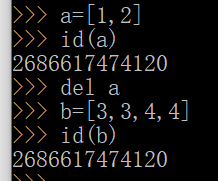
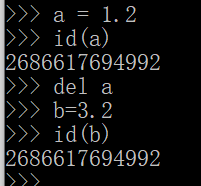
1. 引用计数
2. 引用计数减一时，检查是否为0 ，若是则回收
3. 例如：执行V =0.3

源码内部：开辟内存；初始化工作（对象值为0.3 对象类型为float 对象引用计数为1）；将对象加入双向链表中。

1. 分代回收和标记清除都是为了解决循环引用的问题。

分代就是分0 1 2 代，回收时对要回收的某一代对象进行标记清除

1. 每个对象销毁后内存放在该对象类型所属的内存池内，避免下次申明同类型对象时重新申请内存。如图：



1. **内存管理怎么说：**

Python是由c语言开发的，操作都是基于底层c语言实现，创建每个对象内部都会基于c语言的结构体维护一些值，这两个结构体是PyObject（包含 上指针、下指针、计数器、类型） 、PyVarObject（包含PyObject、内部元素个数）。

**一般单个元素组成的对象（float）使用PyObject结构体，多个元素组成的对象（其他类型）使用PyVarObject结构体。**

你认为：

PyObject：int/float/bool/str

PyVarObject: list/dict/set/tuple

**但实际上，底层是c实现的，c语言里字符串是由多个字符组成，所以str属于PyVarObject，**

**在python3中，int实际上是long类，长度不限制，因为底层是字符串实现的，所以int也是PyVarObject，**

**bool是由01组成的，所以bool也是基于int类型，所以bool也是PyVarObject。**

**因此只有float是由PyObject结构体直接构成，再加一个val属性存放具体值。**

**因此每个对象肯定至少维护四个值：双向链表、ob\_refcnt(引用计数器)、ob\_type（对象类型），如果对象由多个元素组成，再维护一个元素个数的值。（实际上float对象有一个结构体PyFloatObject，里面包含一个PyObject类和一个ob\_fval变量（表示float变量的具体数值））**

**创建对象时python所做的事情（例如v=1.2）**

1. **开辟内存，存放这个对象要维护的至少四个值。**
2. **初始化（所维护的值）。**
3. **将对象加入双向链表ref\_chain。**

创建对象时每个对象至少维护PyObject里四个值：双向链表、ob\_refcnt(引用计数器)、ob\_type（对象类型），之后对内存中的数据初始化，（初始化的本质是引用计数器=1和对其他的量进行赋值），然后将对象添加到双向链表中，链表名为refchain，

以后再有其他引用指向这个内存，引用计数加一，销毁某引用，则找到指向的内存，将对象的引用计数减一。

引用计数器减一时检查，为0则进行内存回收，为了避免内存返回申请和销毁，可能在内部有缓存机制，如float /list /int 最开始不会真正销毁，而是放在该类型的名为free\_list的单向链表中，以后创建同类型数据时，会从链表中取出对象（也就是内存空间），然后再对对象进行初始化。另外对于小整数池内的整数和一般字符串（无空格和特殊字符）也有缓存机制。

1. 垃圾回收机制

以引用计数为主，分代回收和标记清除为辅。

引用计数如上所述，但是引用计数会出现许多循环引用：例子：

a=[1,2]

b=[5,6]

a.append(b)

b.append(a)

del a

del b

1. 标记清除解决循环引用：针对容器类对象，python会将其放入双向链表中，做定期扫描，判断是否有循环引用，如果有则各自的引用计数减1，如果-1后等于0 就直接回收。

分代回收：循环引用的查找和删除过程很繁琐，需要处理每一个容器对象，为了提高效率，为了少扫描对象，将没有问题的对象放到上一级链表中，默认下一级扫描十次，下一秒扫描一次

只有容器对象才会出现循环引用的情况。

由单个元素组成的对象放在一个链表中，这个链表不作标记清除（只做引用计数判断），将容器对象放在另一个链表中，python定期对该链表扫描是否有循环引用。

1. 问：如何解决循环引用的问题？

利用标记清除和分代回收

当分配对象次数和销毁对象次数的差值到到阈值700时执行标记清除。

Python遍历容器对象链表，将每个容器中的对象其分为两组，分别是root object和unreachable，将每个对象所引用对象的引用计数在头部信息中复制一份，并减1，如果减到0就放在unreachable里，否则放在root object里。此时unreachable里肯定包含所有循环引用对象，也包含一些可以从root object间接访问的对象，所以需要再次遍历root object。然后遍历root object，将这些对象里面所引用的对象放到root object里，此时unreachable里都是不可达对象了，将这些对象回收内存。

分代回收主要是为了提高标记清除的效率，减少对寿命长的对象的扫描次数。

对象分为012代，首次创建的对象为第0代，10次0代对象的回收会进行一次1代对象的回收，每次垃圾回收没有被清除的对象会归入下一代。

1. 问：python需要不断开辟内存回收内存，是怎么样解决的？

首先python有一个缓存机制，对于小整数池内的整数都是只存在一份，另外不含空格和其他特殊字符的字符串也只存在一份。

Python有内存池机制，销毁的对象并不会将内存返还给操作系统，而是将该对象放入该对象对应类型的ref\_list链表中。当有相同类型的对象创建时，从该链表中取出一个对象重新进行初始化。（这里的对象指的是底层c语言实现的结构体，初始化就是对该结构体内部的量赋值，结构体需要维护一些值，这就是它为什么占用内存）

总结：

* 内存管理
* 引用计数
* 内存池
* 小整数和非特殊字符串的缓存
* 垃圾回收
* 引用计数
* 标记清除
* 分代回收