# 内存管理和垃圾回收

## python的垃圾回收机制原理

* 引用计数为主（缺点：循环引用无法解决）
* 引入标记清除和分代回收解决引用计数的问题
* 引用计数为主+标记清除和分代回收为辅

### 引用计数

a = None, del a， 离开对象的作用域等等，减少引用计数

getrefcount(a)

什么时候引用计数增加呢？

* 对象创建 a = 1
* 对象被引用 b = a
* 对象作为参数传递 func(a)
* 对象存储在容器中 l = [a]

什么时候引用计数会减少呢？

* 显示使用del a
* 引用指向了别的对象 b = None
* 离开的对象的作用域(比如函数执行结束)
* 从一个容器移除对象或者销毁容器

**优缺点**

优点

* 简单
* 实时性：一旦没有引用，内存就直接释放了。不用像其他机制等到特定时机。

缺点

* 需要额外的空间维护引用计数。
* 不能解决对象的循环引用。(主要缺点)

### 循环引用

标记清除，从根对象找到所有可达的点，不可达的清除掉

标记清除主要是解决循环引用问题。

标记清除算法是一种基于追踪回收（tracing GC）技术实现的垃圾回收算法。

它分为两个阶段：第一阶段是标记阶段，GC 会把所有的 活动对象 打上标记，第二阶段是把那些没有标记的对象 非活动对象 进行回收。那么 GC 又是如何判断哪些是活动对象哪些是非活动对象的呢？

对象之间通过引用（指针）连在一起，构成一个有向图，对象构成这个有向图的节点，而引用关系构成这个有向图的边。从根对象（root object）出发，沿着有向边遍历对象，可达的（reachable）对象标记为活动对象，不可达的对象就是要被清除的非活动对象。根对象就是全局变量、调用栈、寄存器。

标记清除算法作为 Python 的辅助垃圾收集技术主要处理的是容器对象(container，上面讲迭代器有提到概念)，比如 list、dict、tuple 等，因为对于字符串、数值对象是不可能造成循环引用问题。Python 使用一个双向链表将这些容器对象组织起来。

Python 这种简单粗暴的标记清除算法也有明显的缺点：清除非活动的对象前它必须顺序扫描整个堆内存，哪怕只剩下小部分活动对象也要扫描所有对象。

### 分代回收

三代，每隔一段时间，分别对每一代进行标记回收 gc.get\_threshold()

分代回收是一种以空间换时间的操作方式。

Python 将内存根据对象的存活时间划分为不同的集合，每个集合称为一个代，Python 将内存分为了 3“代”，分别为年轻代（第 0 代）、中年代（第 1 代）、老年代（第 2 代），他们对应的是 3 个链表，**它们的垃圾收集频率与对象的存活时间的增大而减小。**新创建的对象都会分配在年轻代，年轻代链表的总数达到上限时，Python 垃圾收集机制就会被触发，把那些可以被回收的对象回收掉，而那些不会回收的对象就会被移到中年代去，依此类推，老年代中的对象是存活时间最久的对象，甚至是存活于整个系统的生命周期内。同时，**分代回收是建立在标记清除技术基础之上。**

分代回收同样作为 Python 的辅助垃圾收集技术处理容器对象

## 参考资料

《全面剖析 Python 面试知识点》

https://gitbook.cn/books/5ca40fd11763103ff10b0e43/index.html