随堂笔记-内存分析

# 主要内容

* 内存分析
* 垃圾回收

# 课程内容

## 内存监控工具的操作

【项目开发[开发环节][测试环节]】

开发环节：要求针对开发代码功能可用性进行开发测试，同时保证代码[**逐行**]使用效率的最优化[调优]

memory\_profiler

测试环节：要求针对开发完成的功能代码进行可用性**完整**测试，同时保证**功能的处理**效率的最优化[调优]

guppy

## memory\_profiler的使用

安装：pip install memory\_profiler

基于函数/方法级别的内存监控使用模块

两种方式可以监控内存操作情况[推荐第二种]

|  |
| --- |
| import memory\_profiler  @memeory\_profiler.profile  def test():  pass  if \_\_name\_\_ == “\_\_main\_\_”:  test()  执行结果中，就会输出test()函数执行过程中的每行代码的执行内存分配情况  不推荐[内存的监控操作，涉及到代码的修改] |
| 通过memoery\_profiler提供的运行脚本，直接运行python代码文件，监控代码中执行过程中的内存分配情况  查看内存使用情况  mprof run \*\*.py 执行python程序并查看内存使用数据报表[数据列表]  查看内存使用可视化分析界面[需要依赖matplotlib 数据分析可视化 模块]  pip install matplotlib  mprof plot 执行命令将程序中内存操作情况输出成可视化图形界面进行调优处理  推荐【包含数据报表、包含可视化分析图】 |
| 所有的内存分析操作，主要目的是附注在测试报告中，提供给运维工程师进行程序性能的调优依据。 |

## 内存的分配

程序运行过程中，为了能有效的利用有限的内存空间，通常情况，python解释器会将运行中的内存空间划分为不同的区域

栈内存：内存空间的分配、释放效率较高，但是数据的稳定性较差

栈内存中数据的分配先进后出、后进先出

分配数据：压栈

释放数据：弹栈

堆内存：内存空间的分配、释放消耗系统较多，存放的数据较为稳定

主要用于存放用户创建的各种自定义对象等等

常量区/静态区：为了提高程序中数据的高可用性，所以将一些数据可以在程序运行申请内存时、程序运行过程中，将这些数据直接存储在这个区域并且常驻内存，在使用到这些数据时直接使用不用再次分配内存空间，提高了处理效率

* -5~256 这一部分数据，认为是在项目中出现频率最高、使用次数最高的数字区间，所以python中将这一部分数据在程序运行前申请内存空间时，就已经将数据存储在申请的内存的静态存储区了。
* 其他不可变数据，在第一次赋值使用时，在常量区/静态区申请并分配内存空间，之后该数据的使用直接赋值内存地址，不需要再次申请/分配内存空间。

|  |
| --- |
| 备注：PYTHON代码的执行，和其他语言稍有不同；PYTHON有自己的交互模式。  python程序正常运行执行时，一个完整的代码模块，会申请一次内存空间。  **交互模式下，每行代码的执行，都是一个完整代码块的运行。** |

## 对象数据运算符

内存可以按照上述的方式进行分配，内存中存储的对象，怎么进行比较？

a b怎么判断变量a中存储的对象和变量b中存储的对象是同一个对象？

a b怎么判断变量a中存储的对象数据和变量b中存储的对象数据是相同的？

### ==

判断操作符号两端的变量中，存储的数据是否相等

### is

判断操作符号两端的变量中，存储的是否同一个对象[内存地址]