**Profile使用总结**

1. **介绍**

脚本语言需要具有先天的性能劣势，所以程序走到最后会出现性能上的问题，Profile是python的标准库，可以统计程序里每一个函数的运行时间，并且提供了多样化的报表。使用profile来分析一个程序很简单。

1. **使用方法**

简单介绍Profile的使用方法，以便更好的理解Profile

* 终端使用方法

Python –m profile example.py

* 代码中修改

if \_\_name\_\_ == "\_\_main\_\_":

import profile

profile.run("example()")

Profile的输出结果一般都是在终端，我们可以在profile.run( )函数再添加一个参数，自定义文件名，保存至该文件。

1. **Pstats的使用**

Profile只解决了我们一个需求，我们还有一个需求就是以不同的方式来输出结果，我们可以通过另一个模块Pstats来解决，它定义了一个stats类，stats接受一个参数参数，就是profile的输出文件名，stats可以对输出结果进行排序和输出控制等。

* 例子：

if \_\_name\_\_ == "\_\_main\_\_":

import profile

profile.run("example ()", " example.txt")

import pstats

p = pstats.Stats("example.txt")

p.sort\_stats("time").print\_stats()

sort\_stats()里的参数表示按不同要求排序

* pstats 也提供了友好的命令行交互环境

(1)python –m pstats

(2)%read /path/example.txt

(3)Stats 0.1

* 注：这里的example.txt必须为二进制文件