**基于python的日志数据自动分析脚本**

1. **需求**

针对飞行数据保存的多组log文件，进行自动化分析，统计震动等相关数据，生成统计表格和可视化显示。

1. **思路**

* 通过循环读取待处理的log日志文件
* 按照log文件中存储的数据格式，通过正则匹配和字符串查找来获取变量数据
* 开启多个进程并行进行处理数据
* 需要获取解锁状态的数据包数据，用于滤去未解锁的状态的数据。因为ADR变量的数据频率和VIBE的数据频率可能不一样。为了方便，采用统一为百分比的形式处理。
* 统计分析和画图

ps上述过程中时间复杂度较高是通过字符串查找变量包和变量数据。当待处理日志组数过多，时间较长。为了加速该过程，采用多进程思路并行处理，虽然计算机开销巨大，但是多组日志处理时间大大缩短。

1. **脚本处理过程**

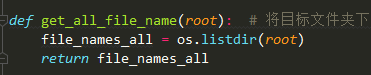
* **设置待处理的文件夹和输出文件路径**



注，待处理日志文件为.log格式，需提前转换好

* **获取该目录下所有.log文件**

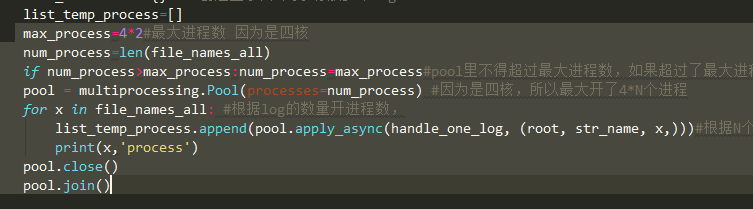




函数get\_all\_file\_name()将读取输入目录下所有文件名，返回list

* **申请进程池，开启多进程**

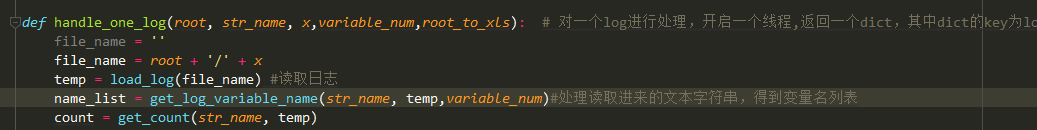
1. 这里申请了process—pool，其实也可以申请线程池，但是多线程运行存在时间片调度和变量锁的问题，即GIL,可能会导致实际运行效率更低，所以采用多进程而不是多线程。



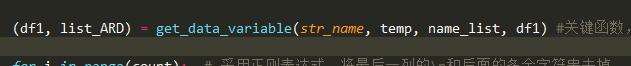
1. 首先设置了8进程为最大进程数，因为进程数一般设置为cpu核数或者其N倍。当进程超过4个时，cpu占用率超过100%，电脑明显卡顿。
2. 其次按照log文件数开始多进程，即N组log文件就向pool申请N组进程，通过for循环实现该操作，每个进程均跑函数handle\_one\_log
3. Pool中每个process赋予函数handle\_one\_log后，需要close和join。当进程数大于8时，例如log组数为9，先开启8个进程，然后完成一个进程后再开始第9个进程。
4. Handle\_one\_log（）函数返回是一个list，里面是 ：[log文件名，tuple]，tuple里面是（数据表格df1，时间窗）

* **每个进程中的日志处理 （handle\_one\_log）**

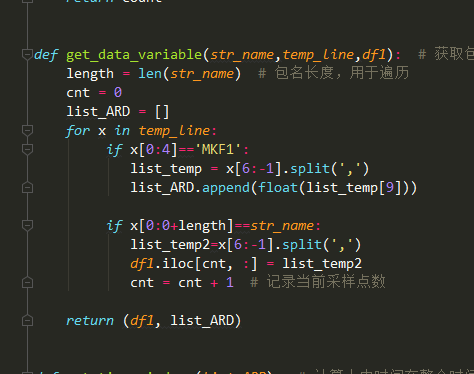
1. 首先需要读取该日志，通过load\_log()函数读取，返回list为存在内存中的文本字符串，因为用的函数是file.readline()，所以读进来的字符串是每行为一个元素
2. Get\_log\_variable\_name()函数，获取str包中的变量名列表，返回list,即得到需要看的变量包里的变量列表，方法是通过字符串的find函数查找
3. get\_count()函数是得到时间序列的采样点数，即行数



1. Get\_data\_variable()是关键函数，用于获取变量包各变量的时间序列，返回值是tuple，里面dataframe和list



* **关键函数，获取各变量的时间序列表格 （get\_data\_variable())**



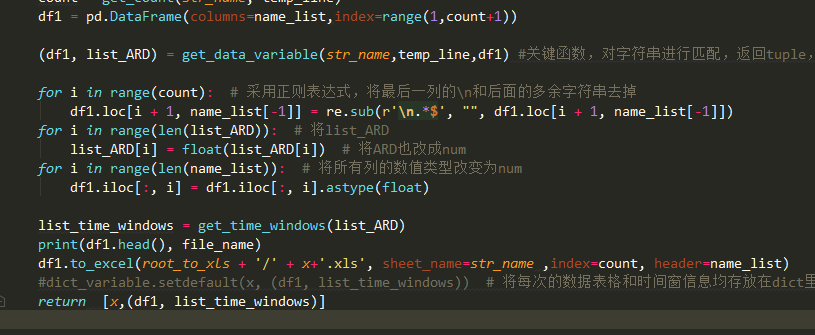
1. 这里对内存中的temp字符串进行了遍历，需要查找两个结果，一个是目标变量包的数据，一个是MKF1变量包的数据，MKF1变量包的数据主要是用来寻找ADR变量值，用于过滤未解锁状态下的数据，list\_ADR是一个list
2. 字符串遍历查找的方式是通过按行查找方式，因为temp\_line是每行字符串组成一个元素，所以只需要查找到对应的数据行即可，找到该行后采用spilt函数分割字符串后即可得到。



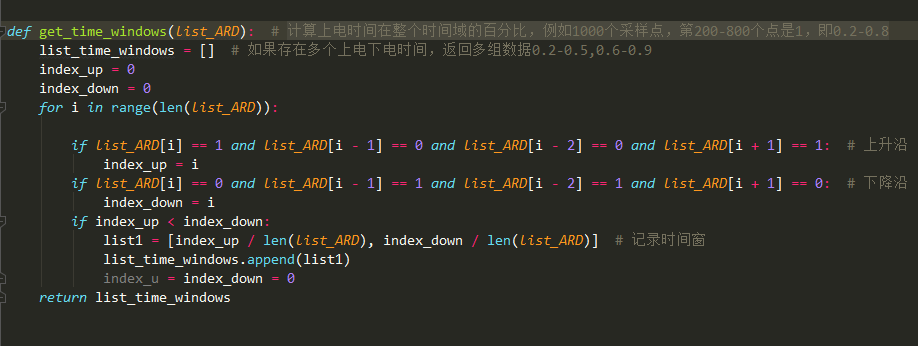
1. 该函数返回的是一个tuple，里面是dataframe表格和list。List是ADR的状态，dataframe里是变量包的数据

* **每个进程中的日志处理 （handle\_one\_log） 后续处理**

1. Handle\_one\_log函数中得到df1和list-ADR后，需要对数据类型进行处理，删除多余的字符串



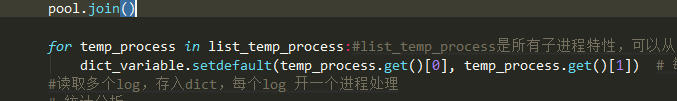
1. Get\_time\_windows（）用于根据list\_adr的值计算解锁状态百分比，因为ADR数据频率和VIBE的数据频率可能不一样，所以用百分比的形式来处理，即计算解锁时间在整个时间域的百分比，例如1000个采样点，第200-800点解锁，即解锁时间百分比为0.2-0.8，如果存在多组解锁状态，则返回多组数据的list，例如[[0.1,0.5],[0.6,0.9]]



1. Get\_time\_windows（）中对ADR的状态进行判断，就是0和1的状态翻转，即上升沿和下降沿判断
2. 最后在handle\_one\_log函数中将每个log文件的数据表格保存，函数返回list

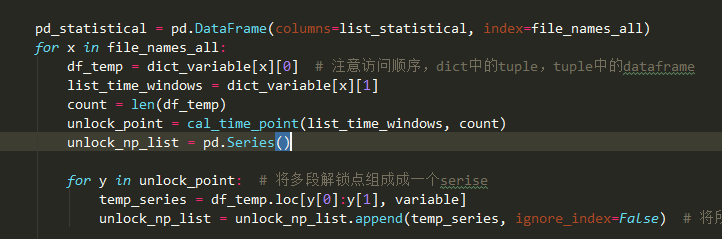
* **子进程和主进程的通信**

1. 进程结束后返回值为函数handle\_one\_log的值，返回值为一个list=[log名，（tuple）]，采用dict的形式存储，log名为key值，tuple为value值。采用dict是因为dict查找速度极快，当log文件过多时，不会影响查找速度。

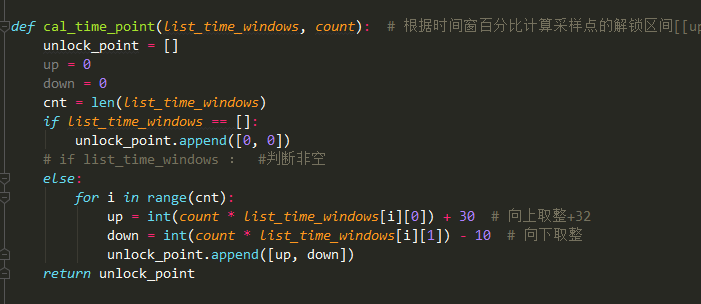


1. 所有进程在list\_temp\_process里，通过get()函数得到其返回值，，注意应该把dict放在主进程中赋值，而不是分别在子进程中赋值，虽然dict是全局作用域，但是在子进程中可能会导致多个进程同时往里赋值，虽然dict也能够接受这种方法。

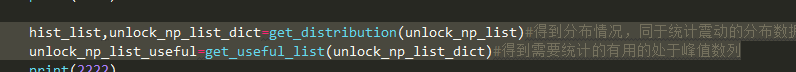
* **统计分析**



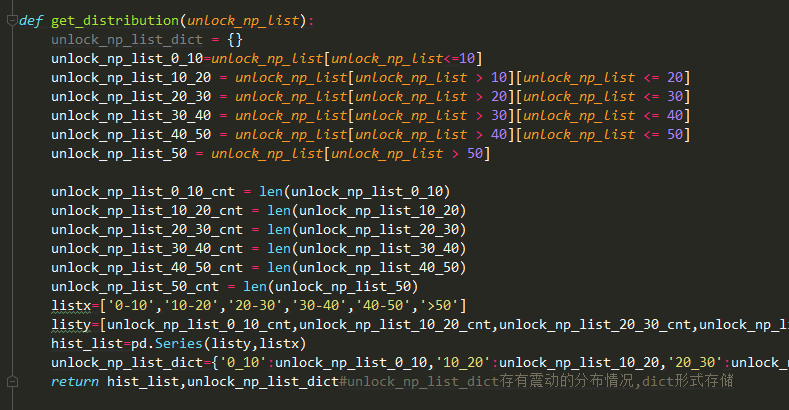
1. 现在所有log数据均存储在dict中，key里是log名，对应的value是数据和时间窗百分比，通过cal\_time\_point()函数计算解锁的采样点，然后对游泳的采样点进行统计。
2. Cal\_time\_point（）是根据时间窗百分比计算解锁的采样点数据，当存在多组解锁状态时，计算出多组解锁状态的采样区间，返回list，最后根据计算对dataframe行筛选，最后多组解锁数据一起组合



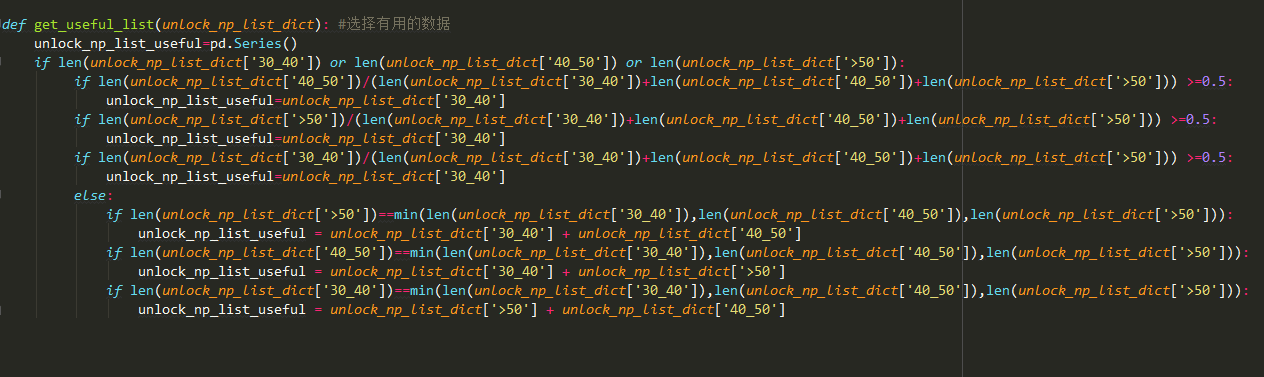
* **对震动数据统计**



以上两个函数实现对震动数据的分类和统计，并绘制柱状图



将震动数据以10为单位分成六组



最后绘制统计表格和饼状图，并保存到输出目录

* **生成exe**

**采用pyinstaller工具生成.EXE,进入目录下，采用以下命令：**

pyinstaller -F 文件名.py

**注意多进程在windows下无法打包成exe的解决方案**

1. **注意**

如果要在本机上跑muil\_process\_log.py脚本，需要事前安装好import的库。另外

Import库中有一个 mul\_process\_package，是为了解决windows下不能多进程打包exe的问题而写的脚本，如果仅仅在本机上运行py脚本，可以将以下两条代码注释掉：





或者将muil\_process\_log.py和mul\_process\_package放在同一目录下运行也可以