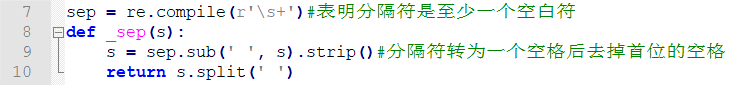
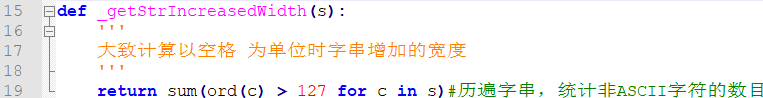
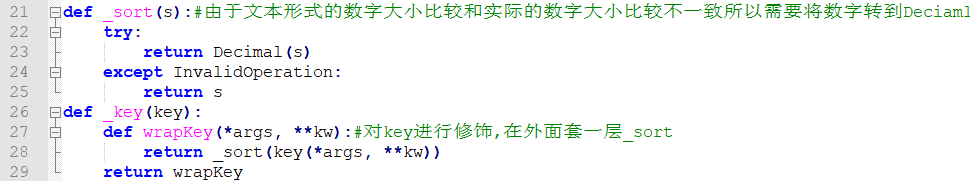
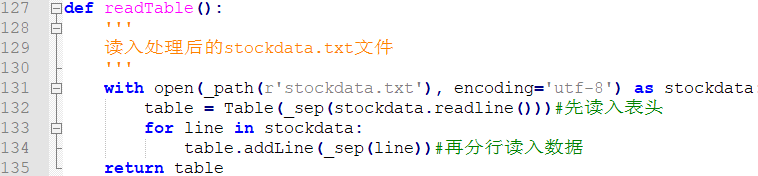
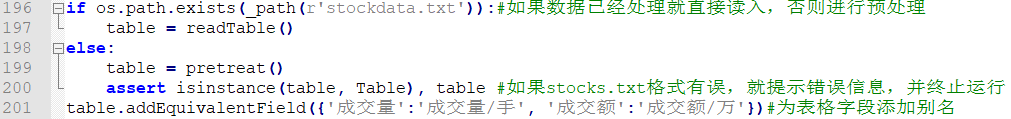
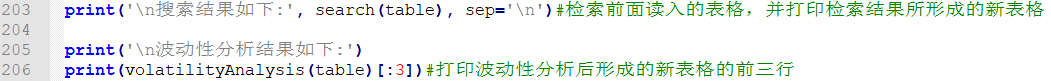
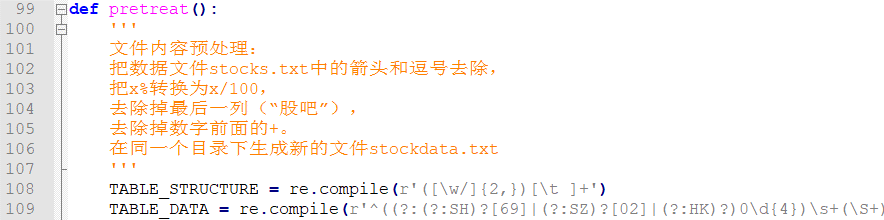
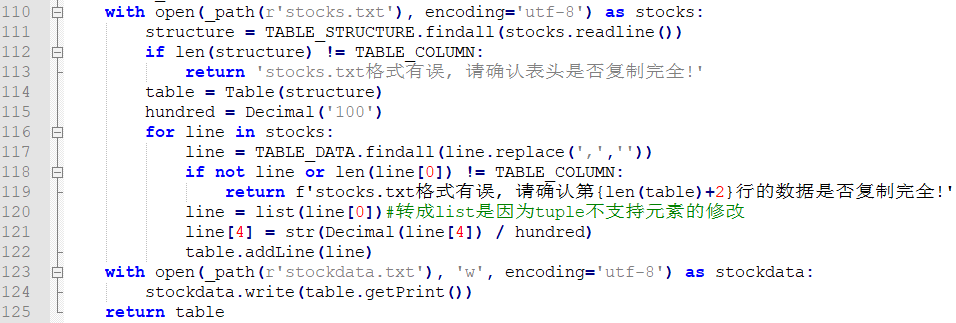
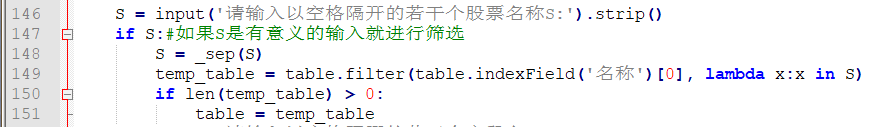
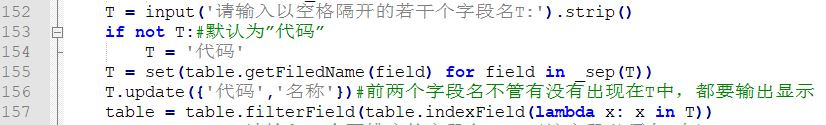
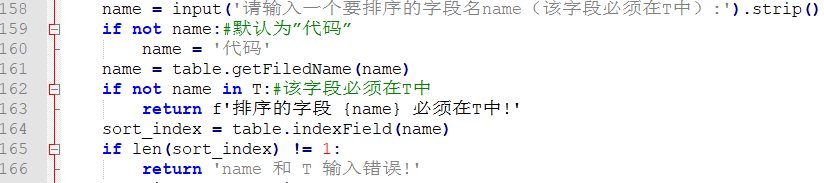
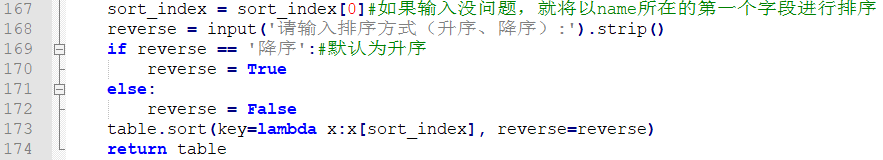
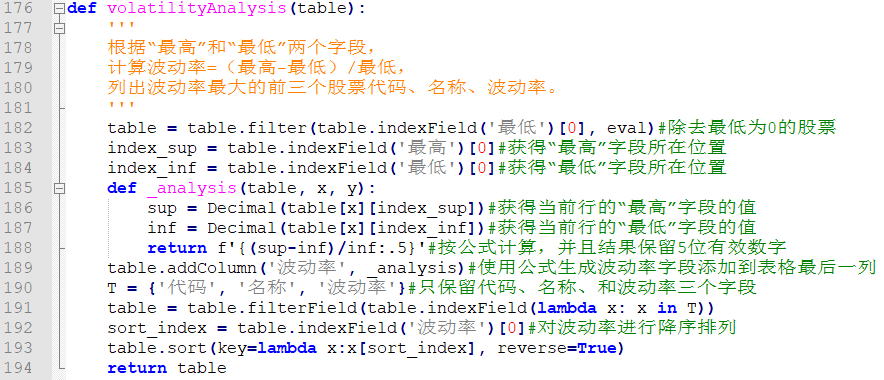
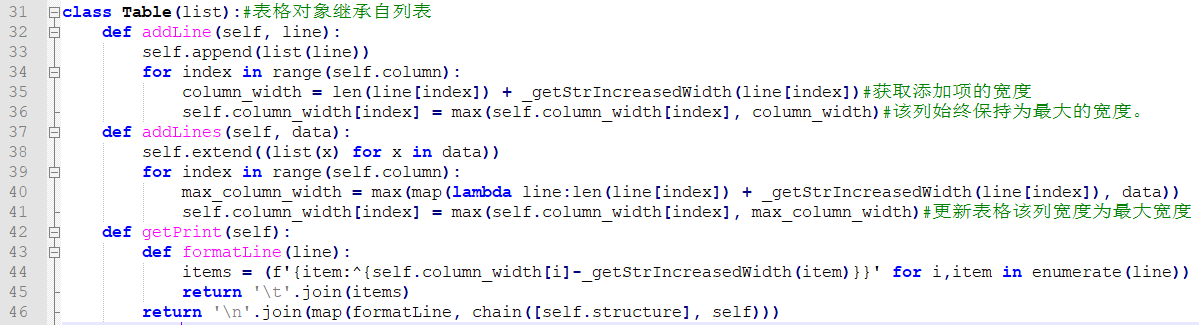
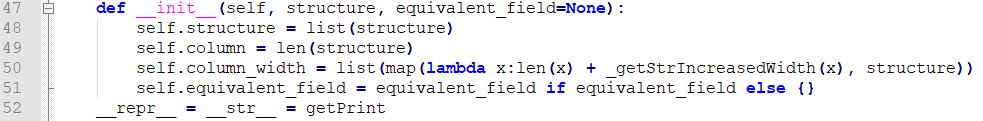
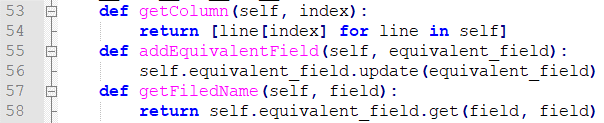
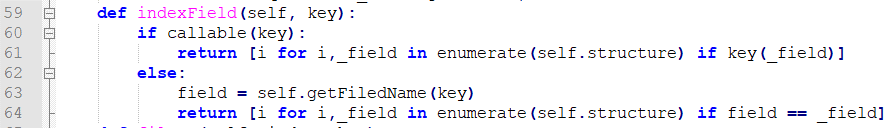
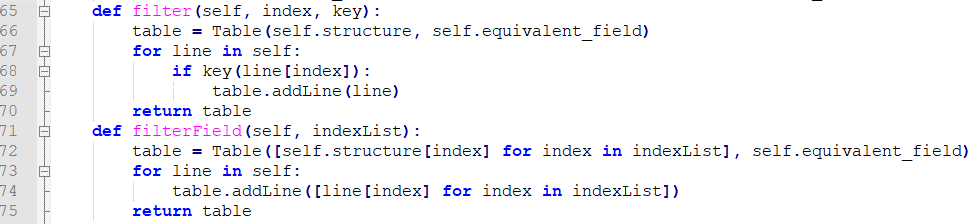
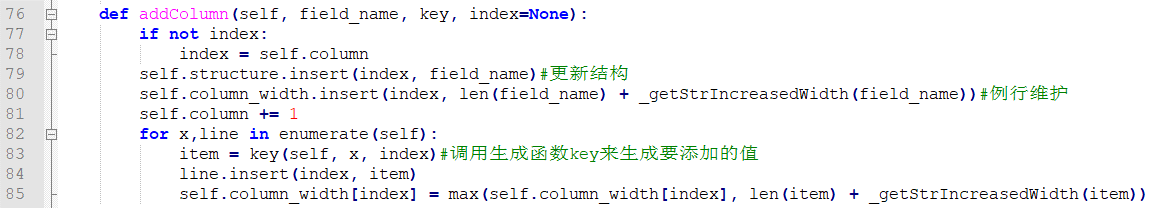
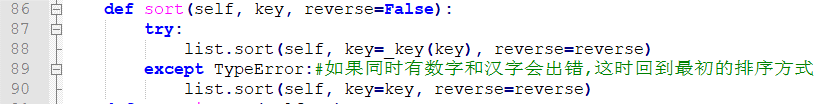
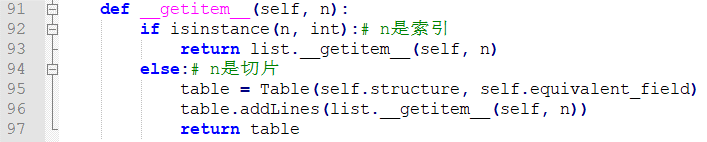
## 设计思路

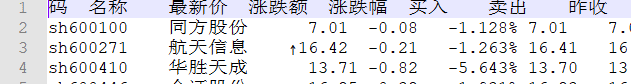
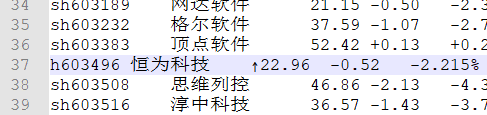
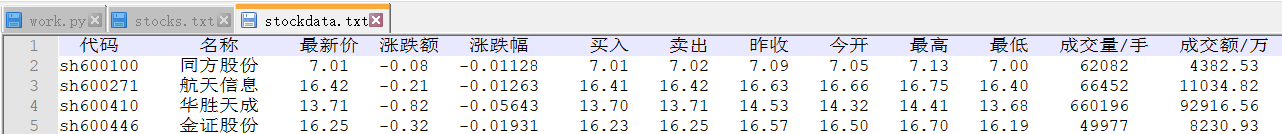
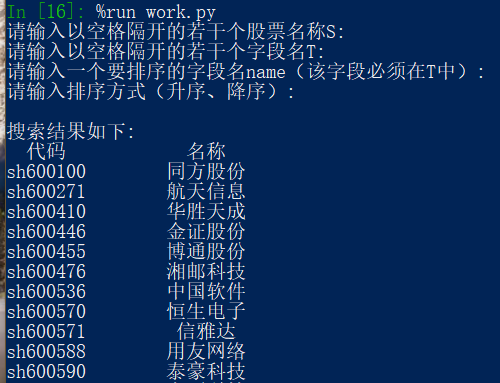
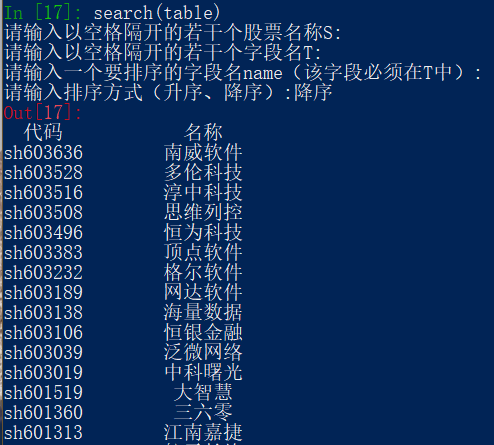
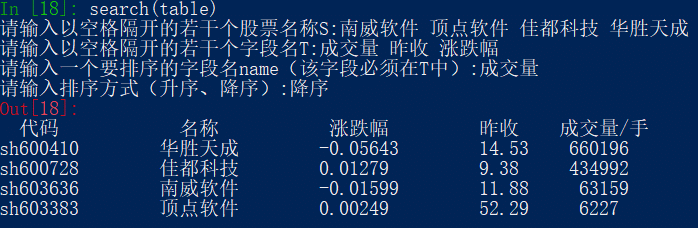
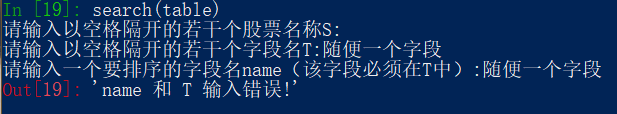
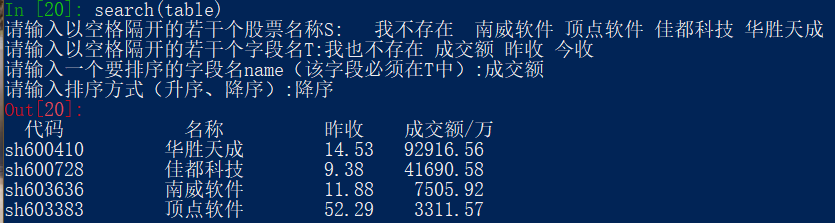
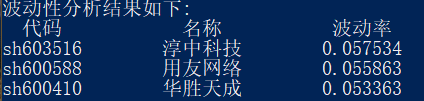
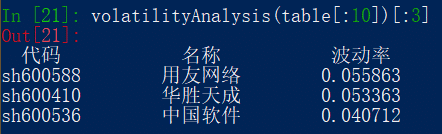
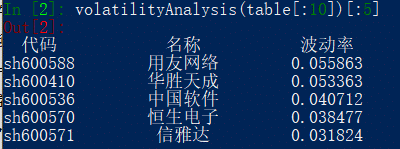
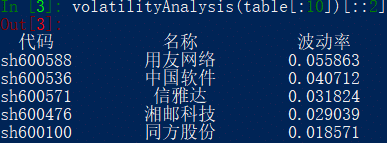
本程序的核心是实现一个表格，支持以文本的形式进行输出显示，支持筛选、排序、切片和自定义公式计算等功能。

* 1. 预处理的实现：  
      一行一行读入数据，采用正则分别提取表头和数据，去除功能可在正则表达式中顺便实现，替换逗号的功能使用字串对象的replace方法实现，百分号转化使用Decimal精确计算。  
      对于stocks.txt不正确的将使用assert 抛出错误，同时提示是表头错误还是数据错误。如果是数据错误，将同时指出发生错误的文本文件的行位置
  2. 检索功能的实现：  
      使用自己实现的表格对象的筛选方法，构造行筛选函数和列筛选函数，传入表格的筛选方法中，其中筛选函数主要利用集合的性质实现。  
      排序可使用表格重写的列表的排序方法构造排序函数实现。
  3. 波动性分析的实现：  
      首先使用表格的筛选功能去除最低为0的行，然后根据公式定义计算函数，传入表格中构造新列“波动率”，再根据“波动率”字段进行排序，最后利用表格的切片功能控制显示前三行。  
      其中公式函数使用Decimal进行精确计算，利用带f标记的字串中的{:.5}保留五位有效数字（非五位小数）
  4. 自定义表格对象的实现：
     1. 数据结构的选择：  
         可选思路有两条，一是用列表储存行后将行储存为列表，二是用列表储存字段后将字段储存为列表，总之都是使用二级列表。为方便文本输出的实现，这里采用第一种思路，继承列表对象来储存每一个行列表，字段名作为表格的结构定义单独储存在structure属性中。采用这种思路还有一个好处——可以直接继承列表的排序方法。
     2. 初始化方法的实现：  
         传入structure来定义表格的结构，同时有一个可选参数equivalent\_field作为字段的各种别名，比如“成交量”作为“成交量/手”的别名，保证这两个都可以定位到同一个字段。structure为可迭代对象（内部自动转为列表后储存为属性），equivalent\_field为字典。同时维护column属性表示列数（字段数），和column\_width（列表类型）属性表示每一列输出时的宽度（即动态维护其对应项数值为当前列表该列的最大宽度项的宽度）  
         宽度采用半角空格为单位，即一个汉字是2宽度而一个字母为1宽度。实现思路为len加上其中非ASCII字符的数目，虽然不一定准确，但对于本题足够使用，毕竟如果要精确区分Unicode字符集中的字符宽度对本题来说太过复杂也无必要。
     3. 添加行方法的实现：  
         利用append方法实现添加一行，同时更新column\_width属性  
         利用extend方法实现添加多行，同时更新column\_width属性
     4. 文本形式输出方法的实现：  
         同时输出表头和数据，列之间以\t间隔，行之间以\n间隔。主要使用map和itertools的chain组合实现。
     5. 行筛选方法的实现：  
         传入筛选字段的位置index和决定函数key。根据筛选的类型构造新的表格对象的结构，根据新的结构和equivalent\_field创建新的表格，历遍self的行，每一行对应于index的项传入到key函数中，根据key的返回值决定是否添加的新的表格对象中，最后返回新创建的表格对象
     6. 字段筛选方法的实现：  
         传入要保留的indexList，根据此列表创建新的表格对象。
     7. 从字段名获取index方法的实现：  
         传入key，如果key是callable则用列表生成式结合enumerate，传入字段名，根据key的返回值决定是否保留index到返回的列表中。如果key是不可callable的，则使用equivalent\_field.get方法将获得准确的字段名，用列表生成式结合enumerate，根据字段名是否精确匹配决定是否保留index到返回的列表中。
     8. 添加公式计算列的实现：  
         传入新列的字段名field\_name， 和该列的生成函数key。可选参数为插入位置index，默认为表格最后。  
         首先更新结构相关信息和column\_width，之后历遍每一行，调用key函数，传入self，当前行位置x和插入位置index，将key函数的返回值插入到当前行对应的位置，并更新column\_width。
     9. 排序方法的实现：  
         对传入的key进行修饰，如果key的返回值可以转化为Decimal就转化为Decimal，否则返回原始字串。如果一列同时存在可以转化为Decimal的值和不可以转化为Decimal的值，则全部当成文本进行排序。
     10. 切片方法的实现：  
          如果索引是数，则直接使用列表的索引，如果索引是切片，则创建新的表格对象，调用列表的切片来更新新的表格对象的行数据。

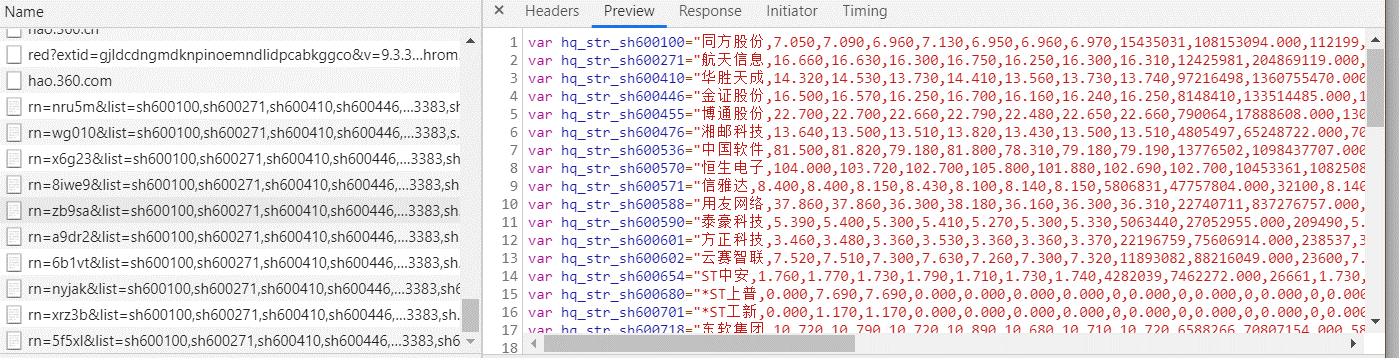
## 主要代码的解释

* 1. 一些中间代码的解释：
     1. 文本分割函数：  
          
        第9行目的是避免分割时中因首尾有空白符而导致结果中出现［‘’,……,’’]这样的情况
     2. 获取数据文件位置的函数：  
          
        可选参数和root or os.getcwd（）表示默认根目录是当前程序运行目录，但可另行指定。使用os.path.join方法的目的是隔离操作系统环境的区别。
     3. 获取非ASCII字符数目的函数：  
        
     4. 排序修饰函数：  
        
     5. 读入处理好的文件的函数：  
        使用了前面的中间函数\_sep进行分割。
     6. 决定怎样读入数据：  
        
     7. 检索和波动性分析的输出：  
        
  2. 预处理代码的解释：  
       
     先定义表头的正则表达式和每一行数据的正则表达式，表头的正则表达式很好理解，其中“股吧”由于式\n结尾，所以不会被匹配上，实现了表头中“股吧”的去除  
     第109行的表达式有点长，具体内容可见源码，其中  
       
     是根据上交所，深交所和港交所对于股票代码的定义构造的正则表达式，之后会有空白符分隔。  
       
     这里的标志表示不区分大小写，因为SH，SZ，HK是小写字母也是可以的。  
       
     这个表示可能会有箭头，但不保留到所获取的匹配中，所以在小括号外  
         
     这三处同理，表示不保留+，%，股吧到所获取的匹配中。  
       
     这里的原理在设计思路中已经说过了，第102行len（table）+2之所以加2是因为该行还没有添加到表格且表格的表头是单独储存的，不在行数中，但文本文件的行数是都包括后的行数。
  3. 检索功能的代码解释：  
     第149行的lambda函数表示如果名称字段的项在S中就返回True进行保留，最后的筛选结果先保存到临时的表格中，第150行表示S的输入是正确的，即不全是错误的股票名称，就保留筛选的结果，否则也按S没有输入处理，默认为所有股票。其中第149行的[0]是因为字段名在一般情况下是可重复的那么就会有多个index与之对应，当然，此题我们知道只有一个。  
       
     第155行表示对别名进行处理，即“成交量”会被转化为“成交量/手”之类的，然后转化为集合  
     第157行表示通过筛选函数对字段进行筛选，筛选函数是判断字段是否在T中。  
       
     第165行是由于前面是允许T中输入不存在的字段名的，如果此时name再输入不存在的字段名就必须进行提示，然后终止此次搜索。  
       
     第173行表示使用前面确定的字段进行排序。
  4. 波动性分析的代码解释：  
     
  5. 表格对象代码解释：  
       
     第40行表示获取data这个二级列表中每一列的最大宽度   
     第44行的self.column\_width[i]-\_getStrIncreasedWidth(item)是因为format居中对齐的宽度计算是不考虑汉字有两个空格的宽度的，所以只好将len+\_getStrIncreasedWidth= self.column\_width[i]进行移项，变左边加为右边减。  
     第45行用\t分隔保证列的对齐更加可靠。  
     第46行的chain是将表头和数据连成一个可迭代对象而不额外进行拷贝，这是考虑到惰性求值可以减少不必要的内存消耗。  
       
     第51行之所以不直接在参数里写equivalent\_field={}，是考虑到后面会对equivalent\_field进行更新。  
     第52行重写\_\_repr\_\_方法和\_\_str\_\_方法为返回文本形式输出的内容，方便调试和简单输出。  
       
     第55行就是对equivalent\_field的更新，第58行就是通过equivalent\_field.get实现可能的别名的转化。  
       
     这个方法之所以会有else分支，是因为通过字段名获得位置太常用了，所以干脆集成到方法里而不再需要构造lambda传入了。  
       
     这是筛选思路的代码实现，应该没有什么特别的  
     第85行也是例行的列宽维护。  
       
     重写父对象的排序方法  
       
     思路里切片的代码实现

## 测试及截图

* 1. 预处理的测试（**先删除stockdata.txt**）：  
       
     当表头少复制了一个字时：  
       
     会终止运行，并提示确认表头是否复制完全。  
       
     当数据少复制了一个字母时：  
       
     会提示去检查对应的行是否有误。  
     格式正确时会生成正确的stockdata.txt，并进入搜索功能  
       
     
  2. 搜索功能测试：  
     一路回车可得到默认结果，已经根据“代码”字段进行了升序排列：  
       
     测试一下降序排列，没有发现问题：  
       
     测试一下别名的功能和排序功能：  
       
     再测试一下错误处理：  
       
     测试一下容错能力(**今收也是不存在的，只有今开**)：  
     
  3. 波动性分析测试：  
     默认是对全部进行分析：  
       
     对前十条进行分析：  
       
     多显示一些：  
       
     间隔显示：  
     

## 存在的问题说明

* 1. 采用数据需要手动复制，过于繁琐，可使用Scrapy爬虫框架编写简单的爬虫获取数据,而且通过Chrome的调试功能发现要爬取应该不困难.  
     
  2. 数据使用文本文件储存，不够方便，可以考虑python内置的sqlite3模块，结合python的metaclass功能，编写ORM框架进行替代。