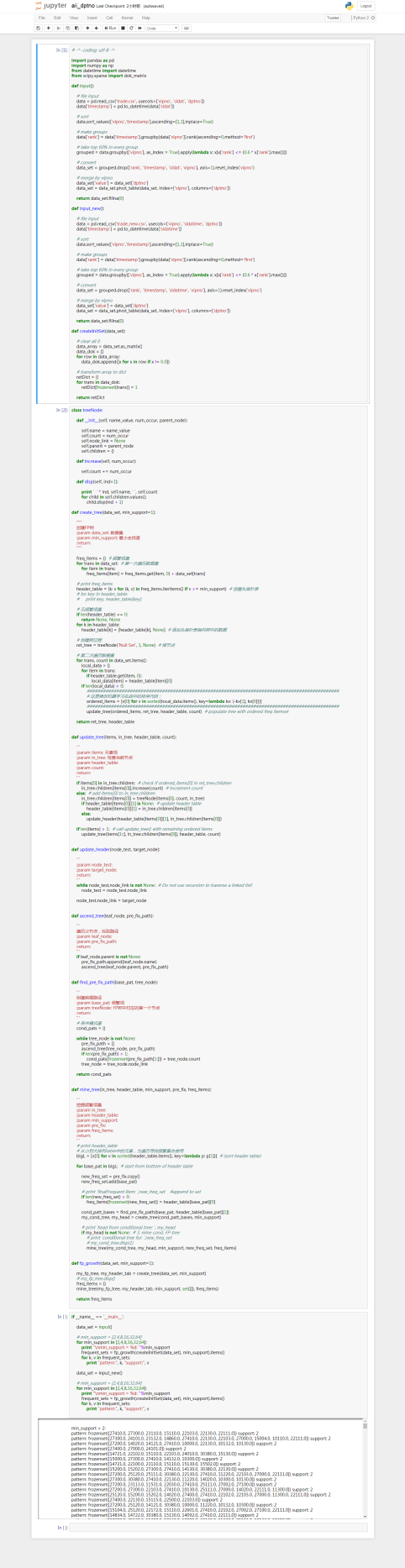
## 代码运行

[点击查看原图](aii_dptno.png)



### 讨论分析

1. 本题采用FP-Growth算法计算商品购买记录中，以单个用户进行全部购买为单位的频繁项集。
2. 算法参考： *Machine Learning in Action*, Peter Harrington
3. FP-Growth算法对于每个频繁项，构造它的[条件投影](https://baike.baidu.com/item/%E6%9D%A1%E4%BB%B6%E6%8A%95%E5%BD%B1/3393313)数据库和投影[FP-tree](https://baike.baidu.com/item/FP-tree/9801330)。对每个新构建的FP-tree重复这个过程，直到构造的新FP-tree为空，或者只包含一条路径。当构造的FP-tree为空时，其前缀即为频繁模式；当只包含一条路径时，通过枚举所有可能组合并与此树的前缀连接即可得到频繁模式。
4. 具体步骤包含：构建FP-tree、频繁挖掘模式。
5. 实际输入为全部用户的单笔订单样本，此处itemno为bndno。
6. 输出为对应频繁项集合。
7. 分析输出可见，频繁集中大部分或者说support较高的频繁项都是单个itemno构成，相当于该itemno在数据集中出现的次数统计。这样的数据符合频繁集要求但是在之后的预测计算中将予以过滤。
8. 本小题中代码产生的输出过大，同时对计算内存要求较高，因此使用Jupyter Notebook 仅运行了trade.csv文件的部分，需要运行全部代码或者查看具体输出信息，请使用cmd运行源码文件中的code.py文件。

### 性能比较

完成两个数据集的频繁集挖掘及输出，总计运行时间为：4374.094s

1. aii相较于ai，使用的单个样本数据长度更长，从算法本身进行分析，即产生的FP-tree有着更大的深度，因此在进行频繁挖掘的过程中会消耗更多的时间。
2. 本小题中产生的频繁项集过大，大量耗时集中于频繁项集的输出，导致最后运行时间过长。
3. 由于本小题耗时过长，此后的性能图表都将使用对数纵坐标表示，基数为10。