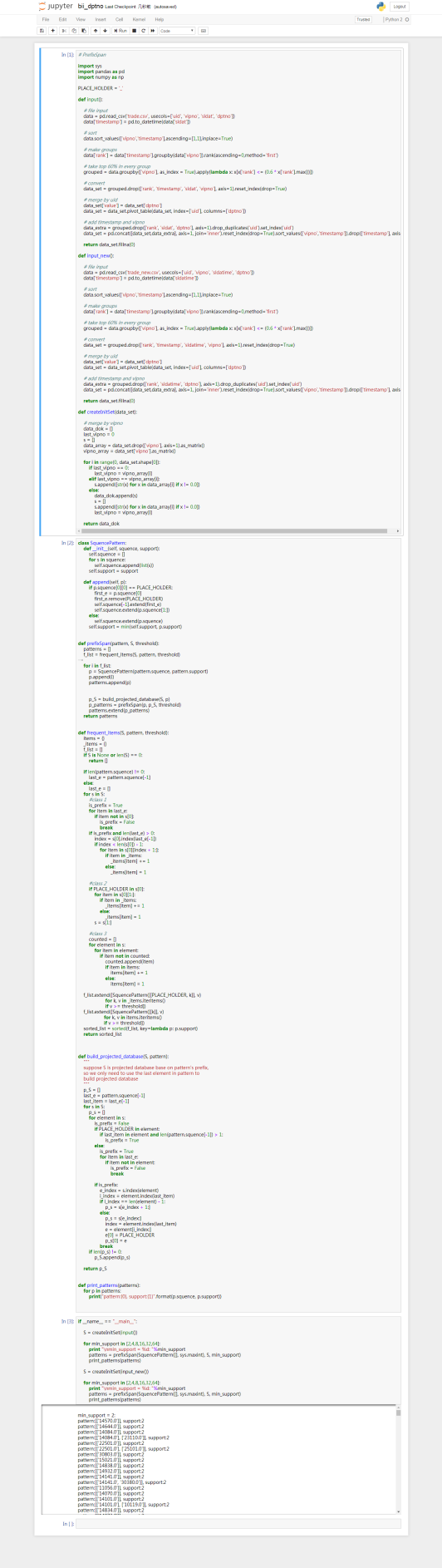
## 代码运行

[点击查看原图](bii_dptno.png)



### 讨论分析

1. 本题采用PrefixSpan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时序频繁项集。
2. 算法参考： *Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach*, Jian Pei, Member, IEEE Computer Society, Jiawei Han, Senior Member, IEEE, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, Member, IEEE Computer Society, and Mei-Chun Hsu(<http://www.philippe-fournier-viger.com/spmf/prefixspan.pdf>)
3. PrefixSpan算法思想与FP-Growth类似：

输入：序列数据集S和支持度阈值α

输出：所有满足支持度要求的频繁序列集

　　　　1）找出所有长度为1的前缀和对应的投影数据库

　　　　2）对长度为1的前缀进行计数，将支持度低于阈值α的前缀对应的项从数据集S删除，同时得到所有的频繁1项序列，i=1.

　　　　3）对于每个长度为i满足支持度要求的前缀进行递归挖掘：

　　　　　　a) 找出前缀所对应的投影数据库。如果投影数据库为空，则递归返回。

　　　　　　b) 统计对应投影数据库中各项的支持度计数。如果所有项的支持度计数都低于阈值αα，则递归返回。

　　　　　　c) 将满足支持度计数的各个单项和当前的前缀进行合并，得到若干新的前缀。

　　　　　　d) 令i=i+1，前缀为合并单项后的各个前缀，分别递归执行第3步具体

1. 实际输入为全部用户的单笔订单样本，此处itemno为dptno。
2. 输出为对应频繁项集合。
3. 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。
4. 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：1220.117s

1. bii与aii除算法外基本相同，具有一定可比性。考虑到PrefixSpan产生的结果数量约等于FP-Growth，可见PrefixSpan在计算性能上不输于FP-Growth。
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。