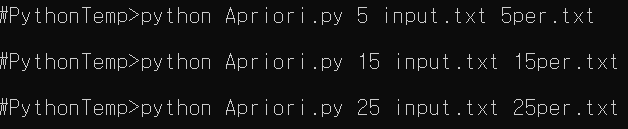
**Apriori Report**

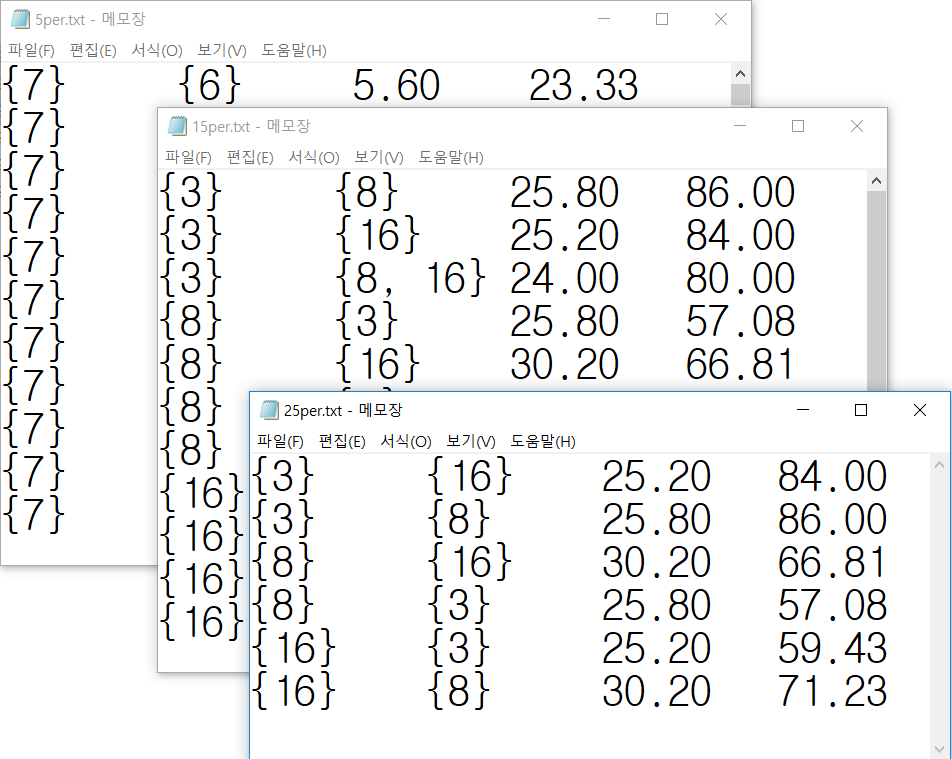
2013011082 이상옥

**1. Result & How to run program**

(python version 3.6)



**Output result**

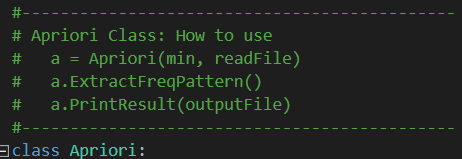


**2. Class: *Apriori***

Code is annotated sufficiently(but in Korean) So, this document is only re-describing in English.

Most of function are concentrated on [class *Apriori*]

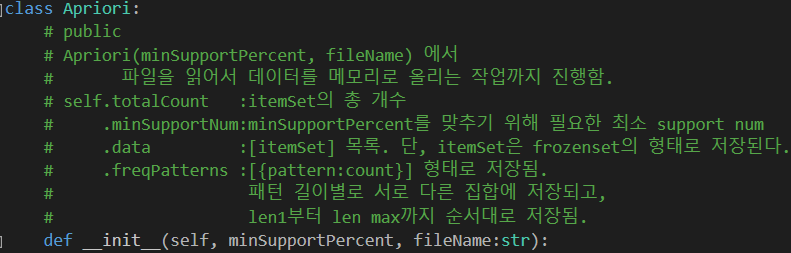
To use my code, you only need to know only 3 functions displayed as annotation below. (These 3 functions will be called as ‘Public’ even if python do not support public/private concept)



**2.1. Public functions of Apriori class**

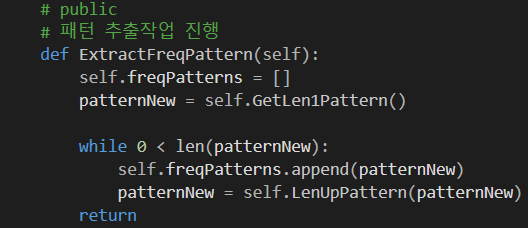
**2.1.1. Constuctor**

When *Apriori* ( *minSupportPercent*, *fileName*) called, it reads data from *fileName* and store it on it’s attribute *data* as *python frozenset list*. Calculate *self. minSupportNum* from *minSupportPercent* and length of *self. data* and set it. Get ready to data mining.

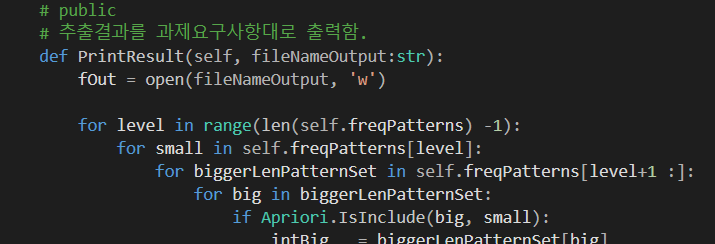


**2.1.2. *self. ExtractFreqPattern()***

Start data mining from *self. data*. Follow Apriori algorithm by using private functions. These private functions are parts of Apriori algorithm. They will be explained later.



**2.1.3. *self. PrintResult(fileNameOutput)***



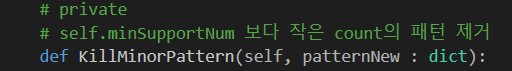
Print mined knowledge on *fileNameOutput* in required format of this assignment.

As I stored frequent patterns group by its length in *self. freqPatterns,* there is no need to compare supper pattern candidate and shorter pattern. So I compare *small, big* patterns.

Level == pattern\_length -1

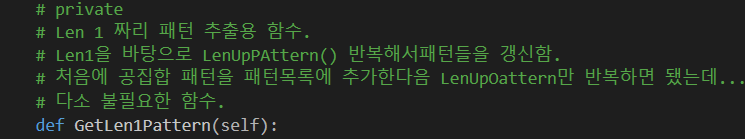
**2.2. Private Functions of *Apriori* Class**

**2.2.2. *self. KillMinorPattern(patternNew)***



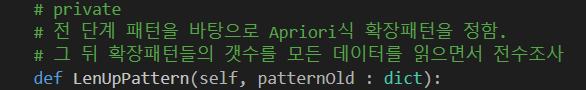
Remove patterns count less than *self. minSupportNum*

**2.2.2. *self. GetLen1Pattern()***



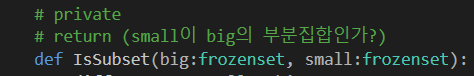
Extract item element from whole data, count them.

**2.2.3. *self. LenUpPattern(patternOld)***



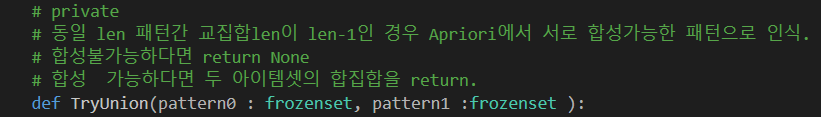
Create New pattern candidate from patternOld,

**2.2.4. *Apriori. IsSubset()***



return bool type answer.

**2.2.5. *Apriori. TryUnion(pattern0, pattern1)***



Used for making new pattern candidate with 1 more length.

If *pattern0* and *pattern1* are suitable for generate new candidate, return that. Or return None.

Apriori.TryUnion ( {1}, {2} ) -> {1,2}

Apriori.TryUnion ( {1,2}, {1,4} ) -> {1,2,4}

Apriori.TryUnion ( {1,2}, {3,4} ) -> None