CSE-3019

DATA MINING

LAB: L47 +L48

Name: Nimit Agrawal

REG. NO: 16BCE0170

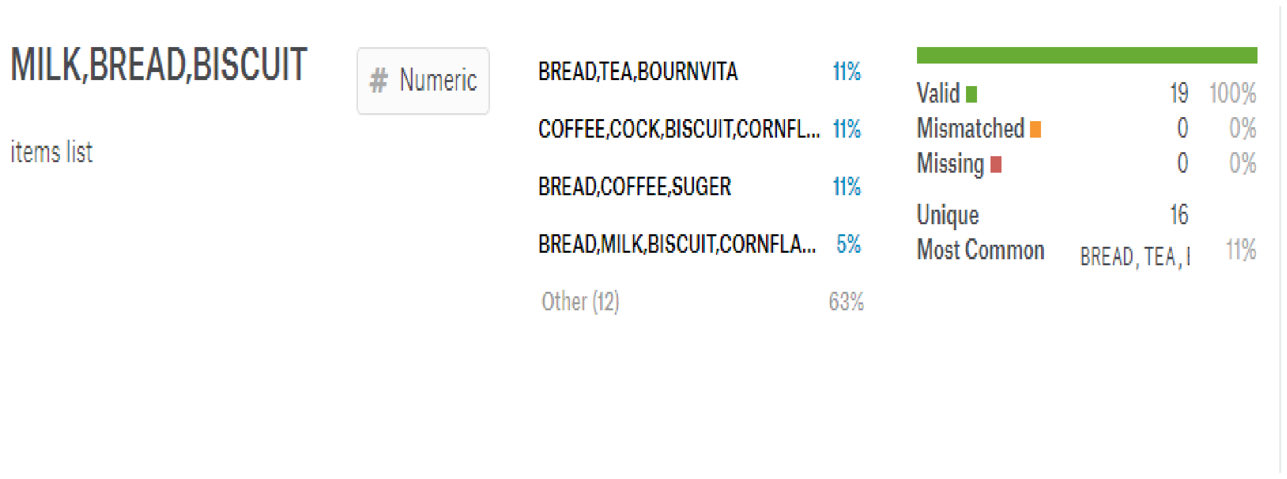
Lab-4

Apply Apriori Algorithm to a dataset and perform market basket analysis with varying support and confidence values.

Output should have the association rule with higher confidence**.**

Dataset:

https://www.kaggle.com/shazadudwadia/supermarket



MILK,BREAD,BISCUIT

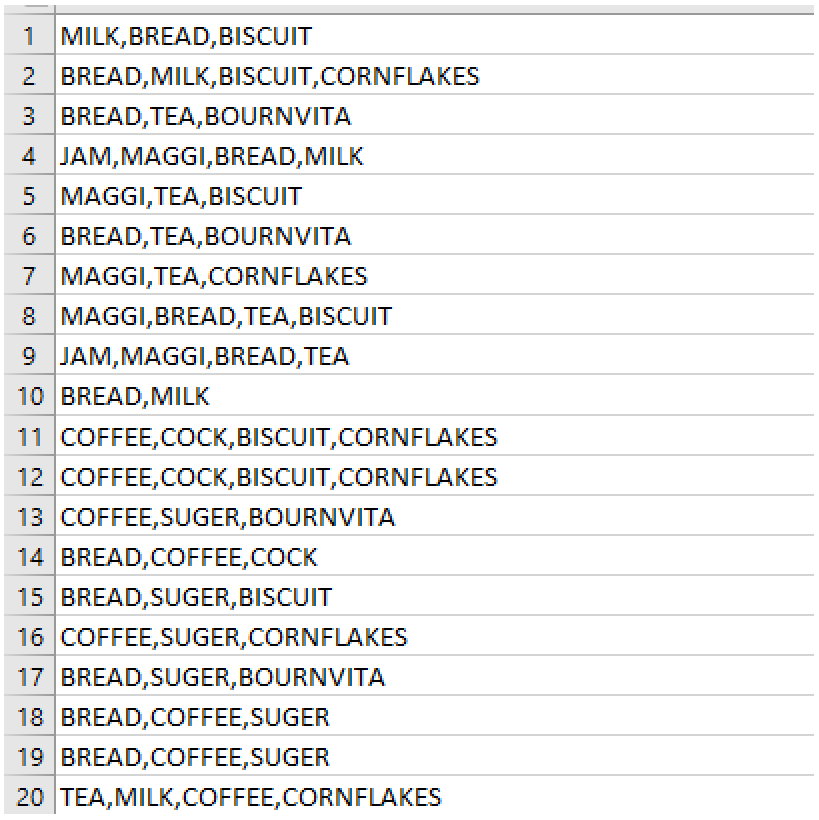
items list

Numeric

BREAD,TEA,BOURNVITA	11%
COFFEE,COCK,BISCUIT,CORNFL...	11%
BREAD,COFFEE,SUGER	11%
BREAD,MILK,BISCUIT,CORNFLA...	5%
Other (12)	63%

Valid 🟩	19	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Unique	16	
Most Common	BREAD, TEA, I	11%

Dataset

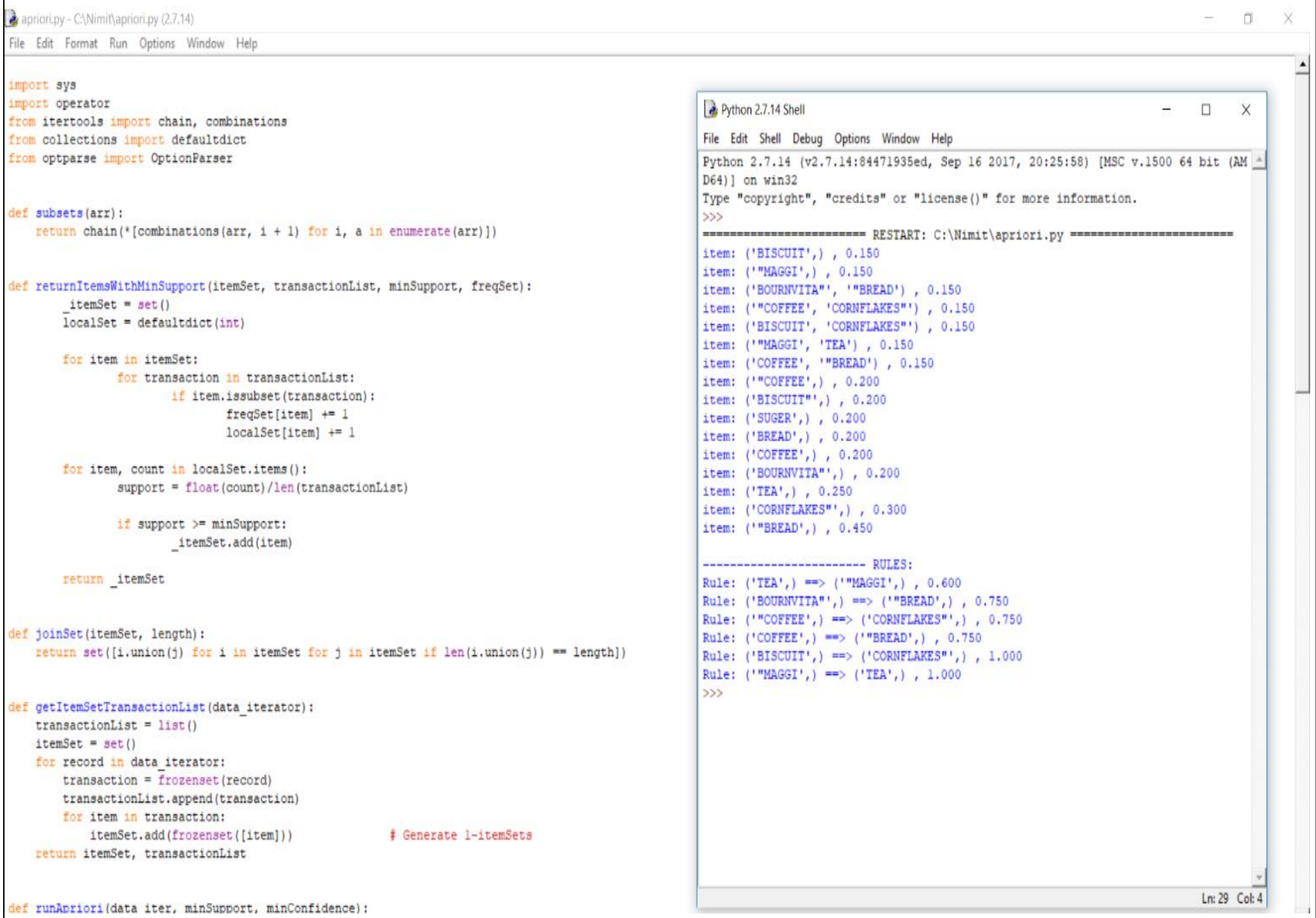


1	MILK,BREAD,BISCUIT
2	BREAD,MILK,BISCUIT,CORNFLAKES
3	BREAD,TEA,BOURNVITA
4	JAM,MAGGI,BREAD,MILK
5	MAGGI,TEA,BISCUIT
6	BREAD,TEA,BOURNVITA
7	MAGGI,TEA,CORNFLAKES
8	MAGGI,BREAD,TEA,BISCUIT
9	JAM,MAGGI,BREAD,TEA
10	BREAD,MILK
11	COFFEE,COCK,BISCUIT,CORNFLAKES
12	COFFEE,COCK,BISCUIT,CORNFLAKES
13	COFFEE,SUGER,BOURNVITA
14	BREAD,COFFEE,COCK
15	BREAD,SUGER,BISCUIT
16	COFFEE,SUGER,CORNFLAKES
17	BREAD,SUGER,BOURNVITA
18	BREAD,COFFEE,SUGER
19	BREAD,COFFEE,SUGER
20	TEA,MILK,COFFEE,CORNFLAKES

Apriori Code:

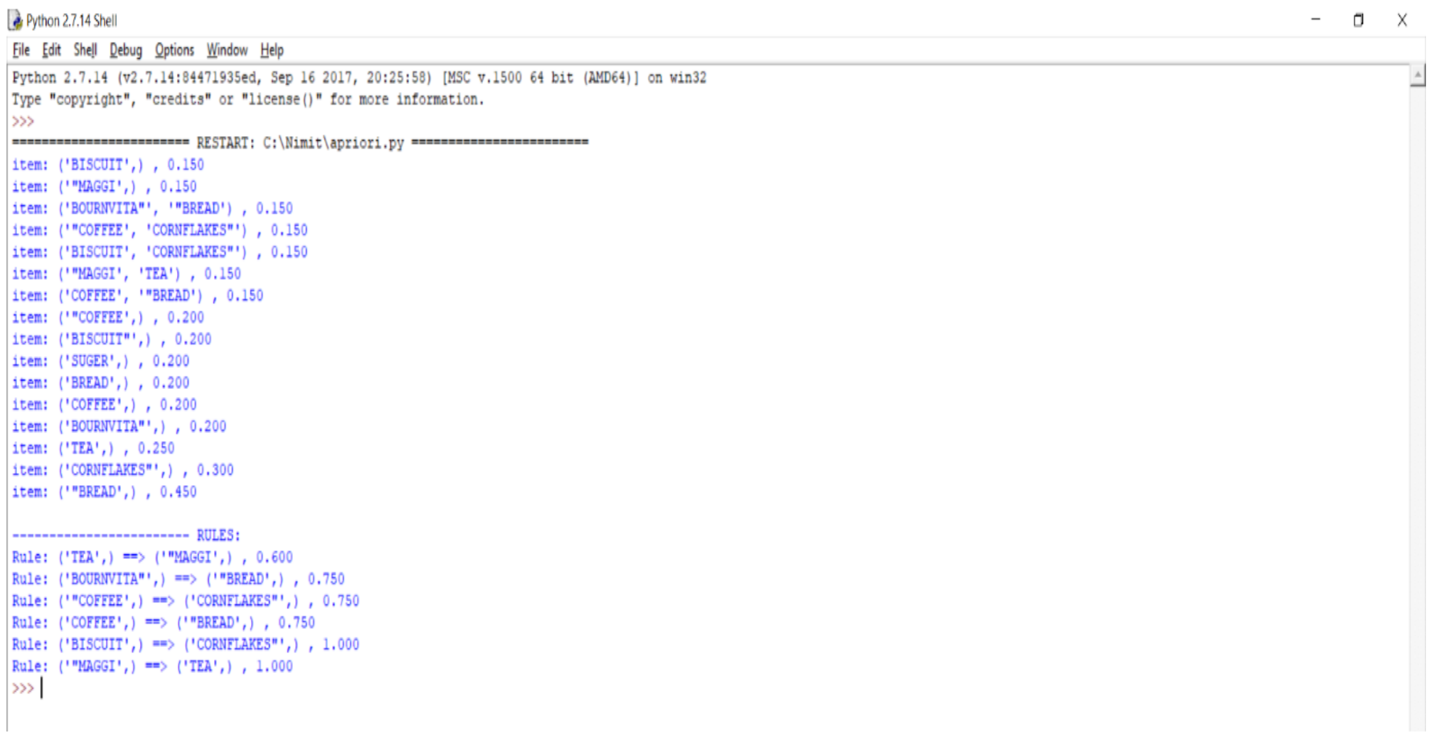
```
import sys

import operator

from itertools import chain, combinations

from collections import defaultdict

from optparse import OptionParser



def subsets(arr):

    return chain(*[combinations(arr, i + 1) for i, a in enumerate(arr)])
```

```
def returnItemsWithMinSupport(itemSet, transactionList, minSupport, freqSet):
        _itemSet = set()
        localSet = defaultdict(int)


        for item in itemSet:
                for transaction in transactionList:
                        if item.issubset(transaction):
                                freqSet[item] += 1
                                localSet[item] += 1


        for item, count in localSet.items():
                support = float(count)/len(transactionList)


                if support >= minSupport:
                        _itemSet.add(item)


        return _itemSet



def joinSet(itemSet, length):
    return set([i.union(j) for i in itemSet for j in itemSet if len(i.union(j)) == length])



def getItemSetTransactionList(data_iterator):
    transactionList = list()
    itemSet = set()
    for record in data_iterator:
        transaction = frozenset(record)
        transactionList.append(transaction)
        for item in transaction:
        itemSet.add(frozenset([item]))          # Generate 1-itemSets
    return itemSet, transactionList



def runApriori(data_iter, minSupport, minConfidence):
    itemSet, transactionList = getItemSetTransactionList(data_iter)

    freqSet = defaultdict(int)
    largeSet = dict()
    # Global dictionary which stores (key=n-itemSets,value=support)
    # which satisfy minSupport

    assocRules = dict()
    # Dictionary which stores Association Rules

    oneCSet = returnItemsWithMinSupport(itemSet,
                                        transactionList,
                                        minSupport,
                                        freqSet)

    currentLSet = oneCSet
    k = 2
    while(currentLSet != set([])):
        largeSet[k-1] = currentLSet
        currentLSet = joinSet(currentLSet, k)
        currentCSet = returnItemsWithMinSupport(currentLSet,
                                                transactionList,
                                                minSupport,
                                                freqSet)
        currentLSet = currentCSet
        k = k + 1

    def getSupport(item):
        """local function which Returns the support of an item"""
        return float(freqSet[item])/len(transactionList)

    toRetItems = []
    for key, value in largeSet.items():
        toRetItems.extend([(tuple(item), getSupport(item))
                           for item in value])

    toRetRules = []
    for key, value in list(largeSet.items())[1:]:
        for item in value:
            _subsets = map(frozenset, [x for x in subsets(item)])
            for element in _subsets:
                remain = item.difference(element)
                if len(remain) > 0:
                    confidence = getSupport(item)/getSupport(element)
                    if confidence >= minConfidence:
                        toRetRules.append(((tuple(element), tuple(remain)),
                                           confidence))
    return toRetItems, toRetRules



def printResults(items, rules):
    for item, support in sorted(items, key=operator.itemgetter(1)):
        print("item: %s , %.3f" % (str(item), support))
    print("\n------------------------ RULES:")
    for rule, confidence in sorted(rules, key=operator.itemgetter(1)):
        pre, post = rule
        print("Rule: %s ==> %s , %.3f" % (str(pre), str(post), confidence))



def dataFromFile(fname):
        """Function which reads from the file and yields a generator"""
        file_iter = open(fname, 'rU')
        for line in file_iter:
                line = line.strip().rstrip(',')                         # Remove trailing comma
                record = frozenset(line.split(','))
                yield record



if __name__ == "__main__":
    inFile=dataFromFile("GroceryStoreDataSet.csv")
    minSupport = 0.15
    minConfidence = 0.6
    items, rules = runApriori(inFile, minSupport, minConfidence)

    printResults(items, rules)
```

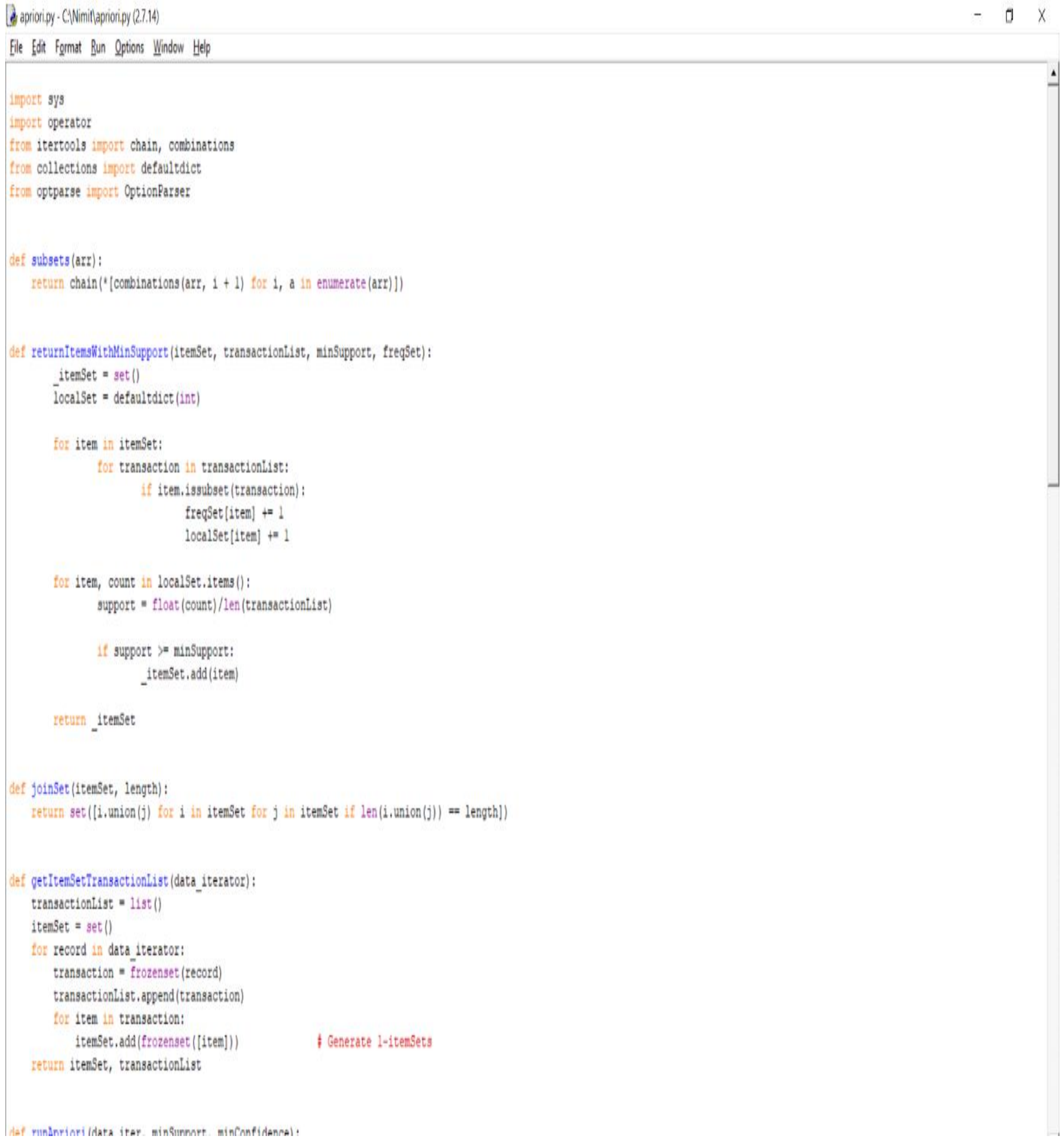

apriori.py - C:\Nimit\apriori.py (2.7.14)
File Edit Format Run Options Window Help

```
import sys
import operator
from itertools import chain, combinations
from collections import defaultdict
from optparse import OptionParser


def subsets(arr):
    return chain(*[combinations(arr, i + 1) for i, a in enumerate(arr)])


def returnItemsWithMinSupport(itemSet, transactionList, minSupport, freqSet):
        _itemSet = set()
        localSet = defaultdict(int)

        for item in itemSet:
                for transaction in transactionList:
                        if item.issubset(transaction):
                                freqSet[item] += 1
                                localSet[item] += 1

        for item, count in localSet.items():
                support = float(count)/len(transactionList)

                if support >= minSupport:
                        _itemSet.add(item)

        return _itemSet


def joinSet(itemSet, length):
    return set([i.union(j) for i in itemSet for j in itemSet if len(i.union(j)) == length])


def getItemSetTransactionList(data_iterator):
    transactionList = list()
    itemSet = set()
    for record in data_iterator:
        transaction = frozenset(record)
        transactionList.append(transaction)
        for item in transaction:
            itemSet.add(frozenset([item]))              # Generate 1-itemSets
    return itemSet, transactionList


def runApriori(data iter, minSupport, minConfidence):
```

```
Python 2.7.14 (v2.7.14:84471935ed, Sep 16 2017, 20:25:58) [MSC v.1500 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
======================= RESTART: C:\Nimit\apriori.py =======================
item: ('BISCUIT',) , 0.150
item: ('"MAGGI',) , 0.150
item: ('BOURNVITA"', '"BREAD') , 0.150
item: ('"COFFEE', 'CORNFLAKES"') , 0.150
item: ('BISCUIT', 'CORNFLAKES"') , 0.150
item: ('"MAGGI', 'TEA') , 0.150
item: ('COFFEE', '"BREAD') , 0.150
item: ('"COFFEE',) , 0.200
item: ('BISCUIT"',) , 0.200
item: ('SUGER',) , 0.200
item: ('BREAD',) , 0.200
item: ('COFFEE',) , 0.200
item: ('BOURNVITA"',) , 0.200
item: ('TEA',) , 0.250
item: ('CORNFLAKES"',) , 0.300
item: ('"BREAD',) , 0.450

------------------------ RULES:
Rule: ('TEA',) ==> ('"MAGGI',) , 0.600
Rule: ('BOURNVITA"',) ==> ('"BREAD',) , 0.750
Rule: ('"COFFEE',) ==> ('CORNFLAKES"',) , 0.750
Rule: ('COFFEE',) ==> ('"BREAD',) , 0.750
Rule: ('BISCUIT',) ==> ('CORNFLAKES"',) , 1.000
Rule: ('"MAGGI',) ==> ('TEA',) , 1.000
>>>
```



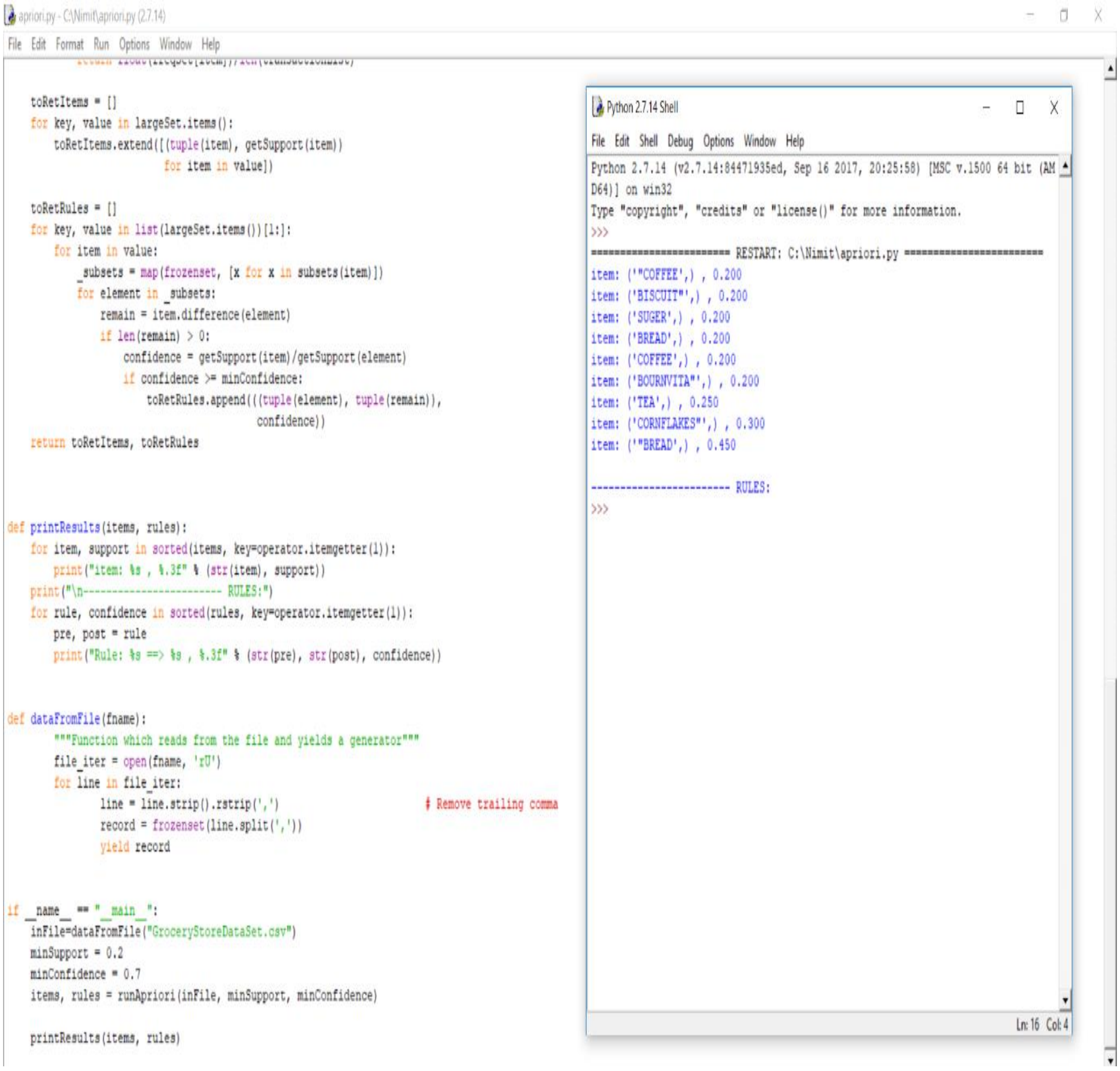
```
Python 2.7.14 (v2.7.14:84471935ed, Sep 16 2017, 20:25:58) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
======================= RESTART: C:\Nimit\apriori.py =======================
item: ('BISCUIT',) , 0.150
item: ('"MAGGI',) , 0.150
item: ('BOURNVITA"', '"BREAD') , 0.150
item: ('"COFFEE', 'CORNFLAKES"') , 0.150
item: ('BISCUIT', 'CORNFLAKES"') , 0.150
item: ('"MAGGI', 'TEA') , 0.150
item: ('COFFEE', '"BREAD') , 0.150
item: ('"COFFEE',) , 0.200
item: ('BISCUIT"',) , 0.200
item: ('SUGER',) , 0.200
item: ('BREAD',) , 0.200
item: ('COFFEE',) , 0.200
item: ('BOURNVITA"',) , 0.200
item: ('TEA',) , 0.250
item: ('CORNFLAKES"',) , 0.300
item: ('"BREAD',) , 0.450

------------------------ RULES:
Rule: ('TEA',) ==> ('"MAGGI',) , 0.600
Rule: ('BOURNVITA"',) ==> ('"BREAD',) , 0.750
Rule: ('"COFFEE',) ==> ('CORNFLAKES"',) , 0.750
Rule: ('COFFEE',) ==> ('"BREAD',) , 0.750
Rule: ('BISCUIT',) ==> ('CORNFLAKES"',) , 1.000
Rule: ('"MAGGI',) ==> ('TEA',) , 1.000
>>>
```


apriori.py - C:\Nimit\apriori.py (2.7.14)
File Edit Format Run Options Window Help

```
import sys
import operator
from itertools import chain, combinations
from collections import defaultdict
from optparse import OptionParser


def subsets(arr):
    return chain(*[combinations(arr, i + 1) for i, a in enumerate(arr)])


def returnItemsWithMinSupport(itemSet, transactionList, minSupport, freqSet):
        _itemSet = set()
        localSet = defaultdict(int)

        for item in itemSet:
                for transaction in transactionList:
                        if item.issubset(transaction):
                                freqSet[item] += 1
                                localSet[item] += 1

        for item, count in localSet.items():
                support = float(count)/len(transactionList)

                if support >= minSupport:
                        _itemSet.add(item)

        return _itemSet


def joinSet(itemSet, length):
    return set([i.union(j) for i in itemSet for j in itemSet if len(i.union(j)) == length])


def getItemSetTransactionList(data_iterator):
    transactionList = list()
    itemSet = set()
    for record in data_iterator:
        transaction = frozenset(record)
        transactionList.append(transaction)
        for item in transaction:
            itemSet.add(frozenset([item]))              # Generate 1-itemSets
    return itemSet, transactionList


def runApriori(data_iter, minSupport, minConfidence):
```

Support:0.2

Confidence:0.7

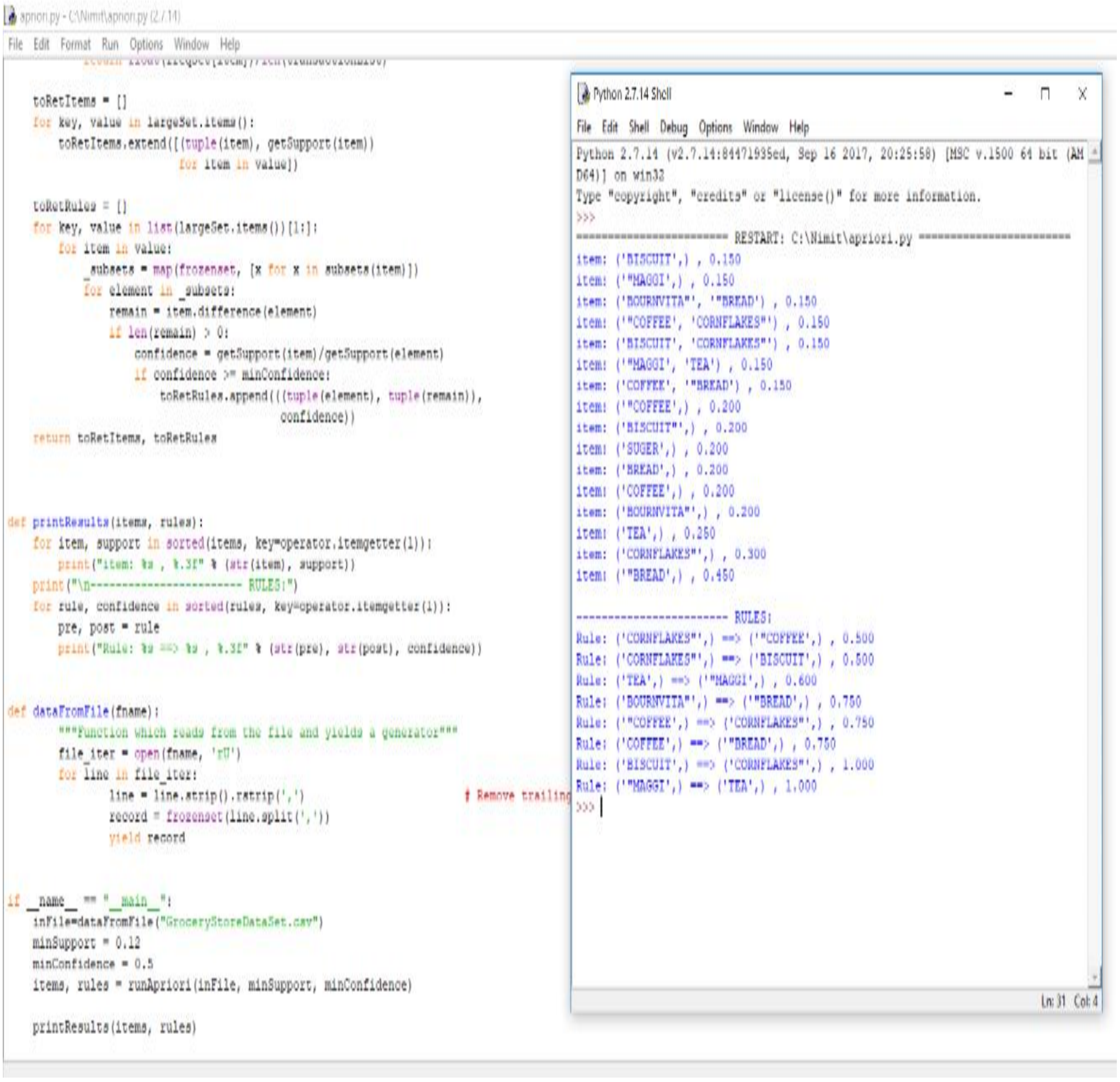


```
    toRetItems = []
    for key, value in largeSet.items():
        toRetItems.extend([(tuple(item), getSupport(item))
                           for item in value])

    toRetRules = []
    for key, value in list(largeSet.items())[1:]:
        for item in value:
            _subsets = map(frozenset, [x for x in subsets(item)])
            for element in _subsets:
                remain = item.difference(element)
                if len(remain) > 0:
                    confidence = getSupport(item)/getSupport(element)
                    if confidence >= minConfidence:
                        toRetRules.append(((tuple(element), tuple(remain)),
                                           confidence))
    return toRetItems, toRetRules



def printResults(items, rules):
    for item, support in sorted(items, key=operator.itemgetter(1)):
        print("item: %s , %.3f" % (str(item), support))
    print("\n------------------------ RULES:")
    for rule, confidence in sorted(rules, key=operator.itemgetter(1)):
        pre, post = rule
        print("Rule: %s ==> %s , %.3f" % (str(pre), str(post), confidence))


def dataFromFile(fname):
        """Function which reads from the file and yields a generator"""
        file_iter = open(fname, 'rU')
        for line in file_iter:
                line = line.strip().rstrip(',')                         # Remove trailing comma
                record = frozenset(line.split(','))
                yield record


if __name__ == "__main__":
    inFile=dataFromFile("GroceryStoreDataSet.csv")
    minSupport = 0.2
    minConfidence = 0.7
    items, rules = runApriori(inFile, minSupport, minConfidence)

    printResults(items, rules)
```

```
Python 2.7.14 (v2.7.14:84471935ed, Sep 16 2017, 20:25:58) [MSC v.1500 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
======================== RESTART: C:\Nimit\apriori.py ========================
item: ('"COFFEE',) , 0.200
item: ('BISCUIT"',) , 0.200
item: ('SUGER',) , 0.200
item: ('BREAD',) , 0.200
item: ('COFFEE',) , 0.200
item: ('BOURNVITA"',) , 0.200
item: ('TEA',) , 0.250
item: ('CORNFLAKES"',) , 0.300
item: ('"BREAD',) , 0.450

------------------------ RULES:
>>>
```

Support: 0.12

Confidence: 0.5

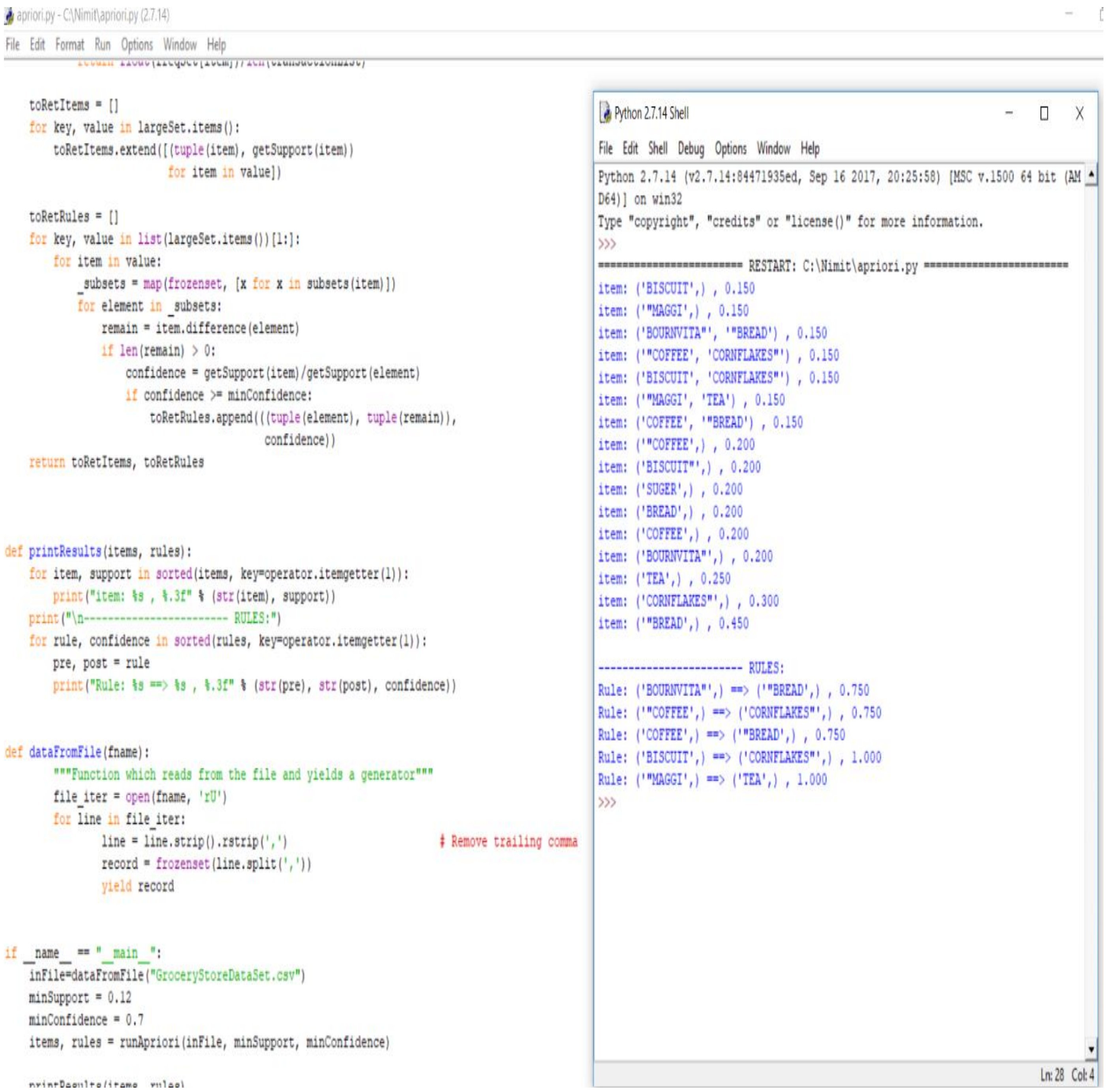

apriori.py - C:\Nimit\apriori.py (2.7.14)
File Edit Format Run Options Window Help
Python 2.7.14 Shell
File Edit Shell Debug Options Window Help
Ln: 31 Col: 4

```
        return float(freqSet[item])/len(transactionList)

    toRetItems = []
    for key, value in largeSet.items():
        toRetItems.extend([(tuple(item), getSupport(item))
                           for item in value])

    toRetRules = []
    for key, value in list(largeSet.items())[1:]:
        for item in value:
            _subsets = map(frozenset, [x for x in subsets(item)])
            for element in _subsets:
                remain = item.difference(element)
                if len(remain) > 0:
                    confidence = getSupport(item)/getSupport(element)
                    if confidence >= minConfidence:
                        toRetRules.append(((tuple(element), tuple(remain)),
                                           confidence))
    return toRetItems, toRetRules



def printResults(items, rules):
    for item, support in sorted(items, key=operator.itemgetter(1)):
        print("item: %s , %.3f" % (str(item), support))
    print("\n------------------------ RULES:")
    for rule, confidence in sorted(rules, key=operator.itemgetter(1)):
        pre, post = rule
        print("Rule: %s ==> %s , %.3f" % (str(pre), str(post), confidence))


def dataFromFile(fname):
        """Function which reads from the file and yields a generator"""
        file_iter = open(fname, 'rU')
        for line in file_iter:
                line = line.strip().rstrip(',')                         # Remove trailing
                record = frozenset(line.split(','))
                yield record


if __name__ == "__main__":
    inFile=dataFromFile("GroceryStoreDataSet.csv")
    minSupport = 0.12
    minConfidence = 0.5
    items, rules = runApriori(inFile, minSupport, minConfidence)

    printResults(items, rules)
```

```
Python 2.7.14 (v2.7.14:84471935ed, Sep 16 2017, 20:25:58) [MSC v.1500 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
======================= RESTART: C:\Nimit\apriori.py =======================
item: ('BISCUIT',) , 0.150
item: ('"MAGGI',) , 0.150
item: ('BOURNVITA"', '"BREAD') , 0.150
item: ('"COFFEE', 'CORNFLAKES"') , 0.150
item: ('BISCUIT', 'CORNFLAKES"') , 0.150
item: ('"MAGGI', 'TEA') , 0.150
item: ('COFFEE', '"BREAD') , 0.150
item: ('"COFFEE',) , 0.200
item: ('BISCUIT"',) , 0.200
item: ('SUGER',) , 0.200
item: ('BREAD',) , 0.200
item: ('COFFEE',) , 0.200
item: ('BOURNVITA"',) , 0.200
item: ('TEA',) , 0.250
item: ('CORNFLAKES"',) , 0.300
item: ('"BREAD',) , 0.450

------------------------ RULES:
Rule: ('CORNFLAKES"',) ==> ('"COFFEE',) , 0.500
Rule: ('CORNFLAKES"',) ==> ('BISCUIT',) , 0.500
Rule: ('TEA',) ==> ('"MAGGI',) , 0.600
Rule: ('BOURNVITA"',) ==> ('"BREAD',) , 0.750
Rule: ('"COFFEE',) ==> ('CORNFLAKES"',) , 0.750
Rule: ('COFFEE',) ==> ('"BREAD',) , 0.750
Rule: ('BISCUIT',) ==> ('CORNFLAKES"',) , 1.000
Rule: ('"MAGGI',) ==> ('TEA',) , 1.000
>>> 
```

Support:0.12

Confidence:0.7



apriori.py - C:\Nimit\apriori.py (2.7.14)

```python
        return float(freqSet[item])/len(transactionList)

    toRetItems = []
    for key, value in largeSet.items():
        toRetItems.extend([(tuple(item), getSupport(item))
                           for item in value])

    toRetRules = []
    for key, value in list(largeSet.items())[1:]:
        for item in value:
            _subsets = map(frozenset, [x for x in subsets(item)])
            for element in _subsets:
                remain = item.difference(element)
                if len(remain) > 0:
                    confidence = getSupport(item)/getSupport(element)
                    if confidence >= minConfidence:
                        toRetRules.append(((tuple(element), tuple(remain)),
                                           confidence))
    return toRetItems, toRetRules



def printResults(items, rules):
    for item, support in sorted(items, key=operator.itemgetter(1)):
        print("item: %s , %.3f" % (str(item), support))
    print("\n------------------------ RULES:")
    for rule, confidence in sorted(rules, key=operator.itemgetter(1)):
        pre, post = rule
        print("Rule: %s ==> %s , %.3f" % (str(pre), str(post), confidence))


def dataFromFile(fname):
        """Function which reads from the file and yields a generator"""
        file_iter = open(fname, 'rU')
        for line in file_iter:
                line = line.strip().rstrip(',')                         # Remove trailing comma
                record = frozenset(line.split(','))
                yield record


if __name__ == "__main__":
    inFile=dataFromFile("GroceryStoreDataSet.csv")
    minSupport = 0.12
    minConfidence = 0.7
    items, rules = runApriori(inFile, minSupport, minConfidence)

    printResults(items, rules)
```

Python 2.7.14 Shell

```
Python 2.7.14 (v2.7.14:84471935ed, Sep 16 2017, 20:25:58) [MSC v.1500 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
======================== RESTART: C:\Nimit\apriori.py ========================
item: ('BISCUIT',) , 0.150
item: ('"MAGGI',) , 0.150
item: ('BOURNVITA"', '"BREAD') , 0.150
item: ('"COFFEE', 'CORNFLAKES"') , 0.150
item: ('BISCUIT', 'CORNFLAKES"') , 0.150
item: ('"MAGGI', 'TEA') , 0.150
item: ('COFFEE', '"BREAD') , 0.150
item: ('"COFFEE',) , 0.200
item: ('BISCUIT"',) , 0.200
item: ('SUGER',) , 0.200
item: ('BREAD',) , 0.200
item: ('COFFEE',) , 0.200
item: ('BOURNVITA"',) , 0.200
item: ('TEA',) , 0.250
item: ('CORNFLAKES"',) , 0.300
item: ('"BREAD',) , 0.450

------------------------ RULES:
Rule: ('BOURNVITA"',) ==> ('"BREAD',) , 0.750
Rule: ('"COFFEE',) ==> ('CORNFLAKES"',) , 0.750
Rule: ('COFFEE',) ==> ('"BREAD',) , 0.750
Rule: ('BISCUIT',) ==> ('CORNFLAKES"',) , 1.000
Rule: ('"MAGGI',) ==> ('TEA',) , 1.000
>>>
```

File Edit Format Run Options Window Help
File Edit Shell Debug Options Window Help
Ln: 28 Col: 4