Homework 1. Frequent itemset

Double Click here to edit this cell

- Name: 김동규
- Student ID: 201800615
- Submission date:

We want to practice the python skills in Chap 2 Crash course

Remark. Do not use numpy, pandas, sklearn, or any module implementing the solution directly. If you use them, you get 0 point

Frequent itemset

- ***Support*** is an indication of how frequently the itemset X appears in the dataset T.
- The support of X with respect to T is defined as the proportion of transactions t in the dataset which contains the itemset X.

$$\text{supp}(X) = \frac{|\{t \in T; X \subseteq t\}|}{|T|}$$

- Frequent itemset is an itemset whose support $\geq$ ***min_sup***.

Data set

- Each line in the following can be imagined as a market basket, which contains items you buy.

入力 [1]:

```
# DO NOT EDIT THIS CELL
data_str = 'apple,beer,rice,chicken₩n'
data_str += 'apple,beer,rice₩n'
data_str += 'apple,beer₩n'
data_str += 'apple,mango₩n'
data_str += 'milk,beer,rice,chicken₩n'
data_str += 'milk,beer,rice₩n'
data_str += 'milk,beer₩n'
data_str += 'milk,mango'
```

Problem 1 (2 pts)

- Define a function ***record_gen*** generating a list of items each ***next***.
- It must be a generator.
- Use ***yield*** instead of ***return***

入力 [2]:

```
# YOUR CODE MUST BE HERE
import re
def gen_record(n):
    n=re.split('₩n',n)
    """줄별로 분리"""
    a=0
    for a in range(len(n)):
        k=n[a]
        k=re.split(',',k)
        """쉼표로 분리"""
        yield k
```

入力 [3]:

```
# DO NOT EDIT THIS CELL
test = gen_record(data_str)
print(sorted(next(test)))
```

```
['apple', 'beer', 'chicken', 'rice']
```

Your output must be:

```
['apple', 'beer', 'chicken', 'rice']
```

入力 [4]:

```
# DO NOT EDIT THIS CELL
print(sorted(next(test)))
```

```
['apple', 'beer', 'rice']
```

Your output must be:

```
['apple', 'beer', 'rice']
```

Problem 2 (10 pts)

- Define a function ***gen_frequent_1_itemset*** generating 1-itemset.
- It must be a generator.
- We want to find frequent 1-itemset (itemset containing only 1 item)
- Use "set, reduce, or map" at least once
- Your output must be a sorted list of items

입力 [5]:

```
# YOUR CODE MUST BE HERE
from functools import reduce
from collections import Counter
def gen_frequent_1_itemset(n,min_sup):
    """리스트를 하나씩 분리한다"""
    seplist=Counter()
    for i in range(len(n)):
        """하나씩 분리시켜 카운터 딕셔너리로 만듦"""
        seplist+=Counter(n[i])
    resultk=list(seplist.keys())
    """키 값만 분리시켜 리스트로 변경"""
    resultv=list(seplist.values())
    """밸류 값만 분리시켜 리스트로 변경"""
    end=[]
    for i in range(len(seplist)):
        """각각의 퍼센트 계산"""
        if (resultv[i]/len(n))>=min_sup:
            end.append(resultk[i])
    return sorted(end)
```

입力 [6]:

```
# DO NOT EDIT THIS CELL
dataset = list(gen_record(data_str))
print(sorted(list(gen_frequent_1_itemset(dataset, 0.5))))
```

```
['apple', 'beer', 'milk', 'rice']
```

Your output must be (sorted list):

```
['apple', 'beer', 'milk', 'rice']
```

입力 [7]:

```
# DO NOT EDIT THIS CELL
dataset = list(gen_record(data_str))
print(sorted(list(gen_frequent_1_itemset(dataset, 0.7))))
```

```
['beer']
```

Your output must be (sorted list):

```
['beer']
```

입力 [8]:

```
# DO NOT EDIT THIS CELL
dataset = list(gen_record(data_str))
print(sorted(list(gen_frequent_1_itemset(dataset, 0.2))))
```

```
['apple', 'beer', 'chicken', 'mango', 'milk', 'rice']
```

Your output must be (sorted list):

```
['apple', 'beer', 'chicken', 'mango', 'milk', 'rice']
```

Problem 3 (10 pts)

- Define a function ***gen_frequent_2_itemset*** generating 2-itemset.
- It must be a generator.
- We want to find frequent 2-itemset (itemset containing only 2 items)
- Use "set, reduce, or map" at least once
- Your output must be a sorted list of sorted 2-tuple

입力 [9]:

```
# YOUR CODE MUST BE HERE
from collections import Counter
from itertools import combinations
from functools import reduce

def gen_frequent_2_itemset(n2, min_sup):
    result1 = list(gen_frequent_1_itemset(n2, min_sup))
    """1개짜리 결과받음"""
    newlist = list(combinations(result1, 2))
    """콤비네이션으로 새로 만듦"""
    seplist = reduce(lambda a, b: a + b, map(Counter, n2))
    """한개짜리 갯수세기"""
    good = {w: seplist.get(w) for w in result1}
    """1개짜리 결과중에서 한개짜리 갯수 가져오기"""
    result1v = Counter()
    for t in newlist:
        """2개짜리 갯수 세기"""
        for u in n2:
            if t[0] in u and t[1] in u:
                result1v[t] += 1
    real = reduce(lambda a, b: a + b, [t for t in newlist if result1v[t] / len(n2) >= min_sup])
    """각자의 확률 계산"""
    rreal = [tuple(real[i:i+2]) for i in range(0, len(real), 2)]
    """조건에 맞는 조합추가"""
    return sorted(rreal)
```

입力 [10]:

```
# DO NOT EDIT THIS CELL
dataset = list(gen_record(data_str))
print(sorted(list(gen_frequent_2_itemset(dataset, 0.5))))
```

```
[('beer', 'rice')]
```

Your output must be:

```
[('beer', 'rice')]
```

입力 [11]:

```
# DO NOT EDIT THIS CELL
dataset = list(gen_record(data_str))
print(sorted(list(gen_frequent_2_itemset(dataset, 0.3))))
```

```
[('apple', 'beer'), ('beer', 'milk'), ('beer', 'rice')]
```

Your output must be:

```
[('apple', 'beer'), ('beer', 'milk'), ('beer', 'rice')]
```

입力 [12]:

```
# DO NOT EDIT THIS CELL
dataset = list(gen_record(data_str))
print(sorted(list(gen_frequent_2_itemset(dataset, 0.2))))
```

```
[('apple', 'beer'), ('apple', 'rice'), ('beer', 'chicken'), ('beer', 'milk'), ('b
eer', 'rice'), ('chicken', 'rice'), ('milk', 'rice')]
```

Your output must be:

```
[('apple', 'beer'), ('apple', 'rice'), ('beer', 'chicken'), ('beer', 'milk'), ('beer',
'rice'), ('chicken', 'rice'), ('milk', 'rice')]
```

Problem 4 (10 pts)

- Define a function ***gen_frequent_3_itemset*** generating 3-itemset.
- It must be a generator.
- We want to find frequent 3-itemset (itemset containing only 3 items)
- Use "set, reduce, or map" at least once
- Your output must be a sorted list of sorted 3-tuple

입力 [13]:

```
# YOUR CODE MUST BE HERE
from collections import Counter
from itertools import combinations
from functools import reduce

def gen_frequent_3_itemset(n3, min_sup):
    result2 = list(gen_frequent_2_itemset(n3, min_sup))
    """2개짜리 결과 받음"""
    l_seplist=[]
    for i in result2:
        """하나씩 분리시켜 카운터 딕셔너리로 만듦"""
        if i[0] not in l_seplist:
            l_seplist.append(i[0])
        if i[1] not in l_seplist:
            l_seplist.append(i[1])
    newlist = list(combinations(l_seplist, 3))
    """2개짜리를 이루고 있는 요소들로 3개짜리 조합만듦"""
    seplist = reduce(lambda a, b: a + b, map(Counter, n3))
    """1개짜리 갯수 세기"""
    result2v = Counter()
    for k in n3:
        for t in newlist:
            if set(t).issubset(set(k)):
                """3개가 다 포함되어 있는 경우 추가"""
                result2v.setdefault(t)
                if result2v[t]==None:
                    """초기화"""
                    result2v[t]=0
                result2v[t]+=1
    l_result2v=list(result2v)
    real=[]
    for i in range(len(result2v)):
        """확률 계산"""
        a=result2v[l_result2v[i]]
        result=a/len(n3)
        if result>=min_sup:
            """조건에 맞으면 추가"""
            change=tuple(l_result2v[i])
            real.append(change)
    for i in range(len(real)):
        """정렬"""
        real[i]=tuple(sorted(real[i]))

    return real
```

입力 [14]:

```
# DO NOT EDIT THIS CELL
dataset = list(gen_record(data_str))
print(sorted(list(gen_frequent_3_itemset(dataset, 0.25))))
```

```
[('apple', 'beer', 'rice'), ('beer', 'chicken', 'rice'), ('beer', 'milk', 'ric
e')]
```

Your output must be:

```
[('apple', 'beer', 'rice'), ('beer', 'chicken', 'rice'), ('beer', 'milk', 'rice')]
```

입力 [15]:

```
# DO NOT EDIT THIS CELL
dataset = list(gen_record(data_str))
print(sorted(list(gen_frequent_3_itemset(dataset, 0.1))))
```

```
[('apple', 'beer', 'chicken'), ('apple', 'beer', 'rice'), ('apple', 'chicken', 'r
ice'), ('beer', 'chicken', 'milk'), ('beer', 'chicken', 'rice'), ('beer', 'milk',
'rice'), ('chicken', 'milk', 'rice')]
```

Your output must be:

```
[('apple', 'beer', 'chicken'), ('apple', 'beer', 'rice'), ('apple', 'chicken', 'rice'),
('beer', 'chicken', 'milk'), ('beer', 'chicken', 'rice'), ('beer', 'milk', 'rice'), ('c
hicken', 'milk', 'rice')]
```

입力 [16]:

```
# DO NOT EDIT THIS CELL
dataset = list(gen_record(data_str))
print(sorted(list(gen_frequent_3_itemset(dataset, 0.3))))
```

```
[]
```

Your output must be:

```
[]
```

Problem 5 (20 pts)

- We want to check the performance of your implementation
- bread_basket.csv must be in the folder of this notebook

입力 [17]:

```
import pandas as pd
bread = pd.read_csv('bread_basket.csv')
```

入力 [18]:

```
bread.head()
```

出力[18]:

	Transaction	Item	date_time	period_day	weekday_weekend
0	1	Bread	30-10-2016 09:58	morning	weekend
1	2	Scandinavian	30-10-2016 10:05	morning	weekend
2	2	Scandinavian	30-10-2016 10:05	morning	weekend
3	3	Hot chocolate	30-10-2016 10:07	morning	weekend
4	3	Jam	30-10-2016 10:07	morning	weekend

入力 [19]:

```
bread_transactions = bread[["Transaction", "Item"]]
bread_transactions.head(10)
```

出力[19]:

	Transaction	Item
0	1	Bread
1	2	Scandinavian
2	2	Scandinavian
3	3	Hot chocolate
4	3	Jam
5	3	Cookies
6	4	Muffin
7	5	Coffee
8	5	Pastry
9	5	Bread

入力 [20]:

```
bread_transactions.tail(10)
```

出力[20]:

	Transaction	Item
20497	9681	Tea
20498	9681	Spanish Brunch
20499	9681	Christmas common
20500	9682	Muffin
20501	9682	Tacos/Fajita
20502	9682	Coffee
20503	9682	Tea
20504	9683	Coffee
20505	9683	Pastry
20506	9684	Smoothies

入力 [21]:

```
bread_transactions.shape
```

出力[21]:

```
(20507, 2)
```

入力 [22]:

```
bread_transactions_list = bread_transactions.values.tolist()
bread_transactions_list[:10]
```

出力[22]:

```
[[1, 'Bread'],
 [2, 'Scandinavian'],
 [2, 'Scandinavian'],
 [3, 'Hot chocolate'],
 [3, 'Jam'],
 [3, 'Cookies'],
 [4, 'Muffin'],
 [5, 'Coffee'],
 [5, 'Pastry'],
 [5, 'Bread']]
```

入力 [23]:

```
import random
random.shuffle(bread_transactions_list)
bread_transactions_list[:10]
```

出力[23]:

```
[[7039, 'Hot chocolate'],
 [8193, 'Coffee'],
 [8355, 'Jam'],
 [4031, 'Bread'],
 [9402, 'Bread'],
 [7706, 'Cake'],
 [8806, 'Bread'],
 [4083, 'Tea'],
 [7568, 'Bread'],
 [8220, 'Coffee']]
```

Transform bread_transactions_list to bread_dataset, which is a list of lists of items

入力 [24]:

```
# YOUR CODE MUST BE HERE
bread_transactions_list = bread_transactions.values
bread_dataset=[]
before=str(bread.iloc[0][0])
"""초기값 지정"""
print(type(before))
setofbread=[]
for i in range(len(bread_transactions_list)):
    if str(bread.iloc[i][0]) == before:
        """번호 같은거끼리 묶기"""
        setofbread.append(bread.iloc[i][1])
    else:
        """다르면 집어넣기"""
        bread_dataset.append(setofbread)
        before=str(bread.iloc[i][0])
        setofbread=[]
        """초기화하가 새로 묶기"""
        setofbread.append(bread.iloc[i][1])
```

```
<class 'str'>
```

入力 [25]:

```
# DO NOT EDIT THIS CELL

import time
start = time.time()
print(sorted(list(gen_frequent_3_itemset(bread_dataset, min_sup=0.01))))
end = time.time()

lapse = end - start
total = 10
weight = 1.5
grace = 6.0
my_point = int(total / (weight ** (lapse // grace)))
print(f'Total time taken : {lapse} seconds')
print(f'My point is {my_point}')
```

```
[('Bread', 'Cake', 'Coffee'), ('Bread', 'Coffee', 'Pastry'), ('Cake', 'Coffee',
'Tea')]
Total time taken : 2.553394079208374 seconds
My point is 10
```

Your output must be (total time taken must be around 4 seconds. Your point depends on time taken):

```
[('Bread', 'Cake', 'Coffee'), ('Bread', 'Coffee', 'Pastry'), ('Cake', 'Coffee', 'Tea')]
Total time taken : 4.363343238830566 seconds
My point is 10
```

入力 [26]:

```
# DO NOT EDIT THIS CELL

import time
start = time.time()
print(sorted(list(gen_frequent_2_itemset(bread_dataset, min_sup=0.03))))
end = time.time()

lapse = end - start
total = 10
weight = 1.5
grace = 2.0
my_point = int(total / (weight ** (lapse // grace)))
print(f'Total time taken : {lapse} seconds')
print(f'My point is {my_point}')
```

```
[('Bread', 'Coffee'), ('Cake', 'Coffee'), ('Coffee', 'Medialuna'), ('Coffee', 'Pa
stry'), ('Coffee', 'Sandwich'), ('Coffee', 'Tea')]
Total time taken : 0.3250553607940674 seconds
My point is 10
```

Your output must be (total time taken must be around 1.5 seconds. Your point depends on time taken):

```
[('Bread', 'Coffee'), ('Cake', 'Coffee'), ('Coffee', 'Medialuna'), ('Coffee', 'Pastr
y'), ('Coffee', 'Sandwich'), ('Coffee', 'Tea')]
```

Ethics:

If you cheat, you will get negative of the total points. If the homework total is 22 and you cheat, you get -22.

What to submit

- Run **all cells** after restarting the kernel
- Goto "File -> Print Preview"
- Print the page as pdf
- Submit the pdf file in google classroom
- Pdf file name must be in a form of: homework_1_홍길동_202300001.pdf
- No late homeworks will be accepted
- Your homework will be graded on the basis of correctness, performance, and programming skills

입力 [ ]:

입力 [ ]:

입力 [ ]:

입力 [ ]:

입力 [ ]:

입力 [ ]:

입力 [ ]:

入力 [ ]: