LAB ASSIGNMENT – 5

APRIORI ALGORITHM

NAME : PRATHAPANI SATWIKA

REG.NO. : 20BCD7160

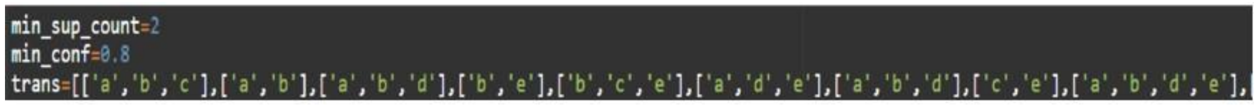
TASK-A:

Task 1: A python function to return count of each set in the list of element sets.

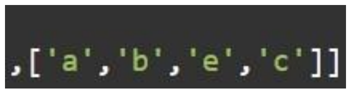
Input: Key: (a,b,c), Data: [[a,b,c,d],[b,c,d],[b,c,d,e],[a,b,c,d,e]]

Output: (a,b,c) → 2

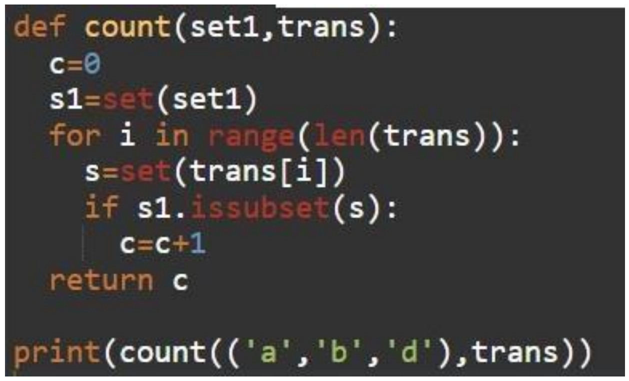
CODE :



```
min_sup_count=2
min_conf=0.8
trans=[['a','b','c'],['a','b'],['a','b','d'],['b','e'],['b','c','e'],['a','d','e'],['a','b','d'],['c','e'],['a','b','d','e'],[
```

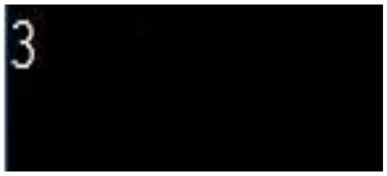


```
,['a','b','e','c']]
```



```
def count(set1,trans):
  c=0
  s1=set(set1)
  for i in range(len(trans)):
    s=set(trans[i])
    if s1.issubset(s):
      c=c+1
  return c

print(count(('a','b','d'),trans))
```

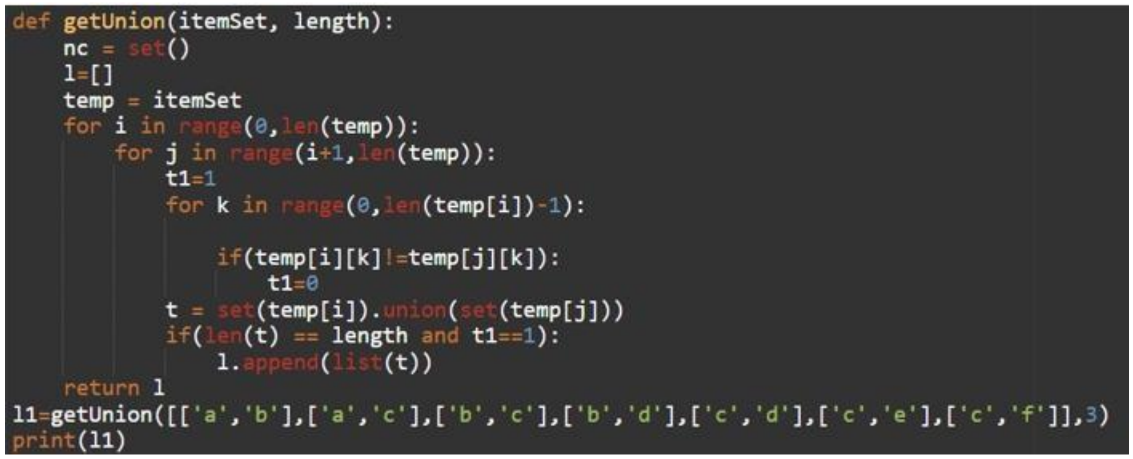
OUTPUT:



```
3
```

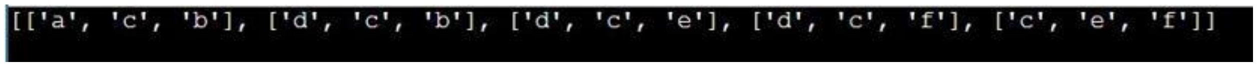
Task 2: A python function to perform self-join operation on a set of items (of size k) to yield unique set of items of size (k+1).
Input:((a,b),(a,c),(b,c),(b,d),(c,d),(c,e),(c,f))
Output:((a,b,c),(b,c,d),(c,d,e),(c,d,f),(c,e,f))
Note: **It should not generate (c,d,e,f) since given k=2. we should generate sets of k=3 but not k=4.**

CODE:



```
def getUnion(itemSet, length):
    nc = set()
    l=[]
    temp = itemSet
    for i in range(0,len(temp)):
        for j in range(i+1,len(temp)):
            t1=1
            for k in range(0,len(temp[i])-1):

                if(temp[i][k]!=temp[j][k]):
                    t1=0
            t = set(temp[i]).union(set(temp[j]))
            if(len(t) == length and t1==1):
                l.append(list(t))
    return l
l1=getUnion([['a','b'],['a','c'],['b','c'],['b','d'],['c','d'],['c','e'],['c','f']],3)
print(l1)
```

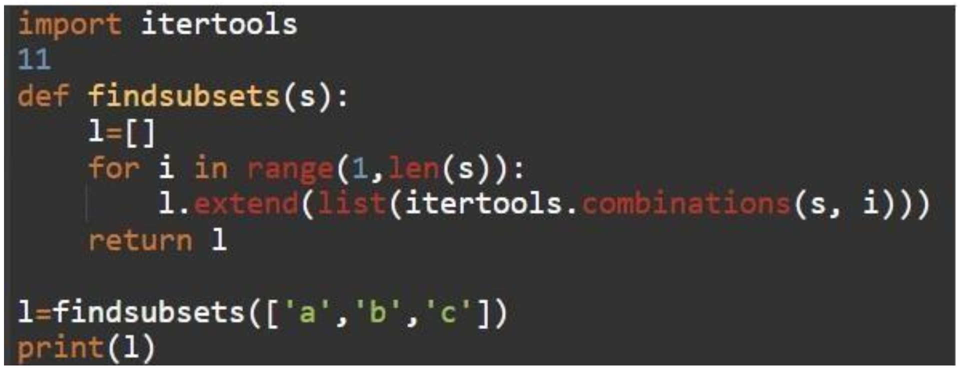
OUTPUT:



```
[['a', 'c', 'b'], ['d', 'c', 'b'], ['d', 'c', 'e'], ['d', 'c', 'f'], ['c', 'e', 'f']]
```

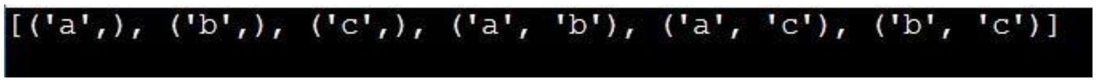
Task 3: A python function to generate nonempty subsets (excluding superset) for a given list of items. Input: [a,b,c] Ouput: [(a),(b),(c),(a,b),(a,c),(b,c)]

CODE:



```
import itertools
11
def findsubsets(s):
    l=[]
    for i in range(1,len(s)):
        l.extend(list(itertools.combinations(s, i)))
    return l

l=findsubsets(['a','b','c'])
print(l)
```

OUTPUT:



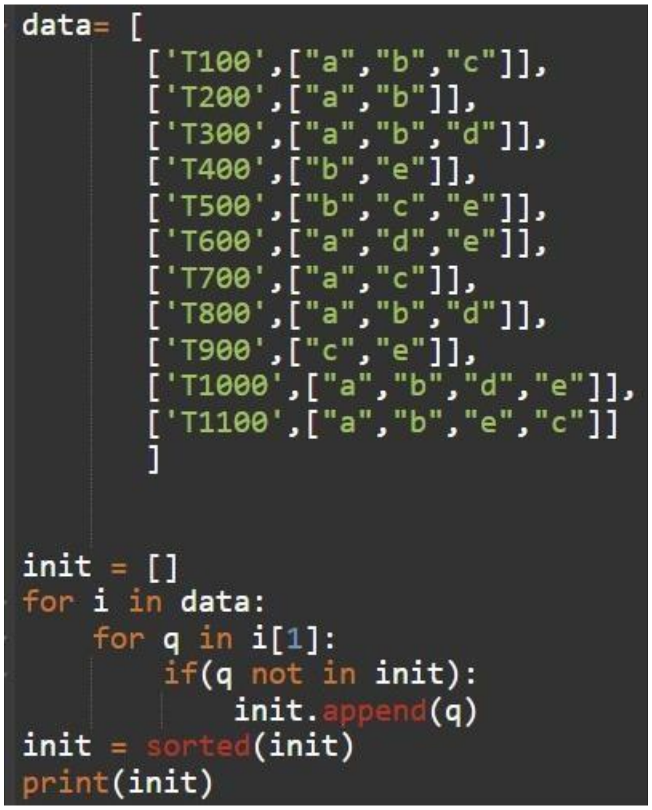
```
[('a',), ('b',), ('c',), ('a', 'b'), ('a', 'c'), ('b', 'c')]
```

TASK B

Extend the above implemented code for any number of transactions, any number of items, user specified thresholds.

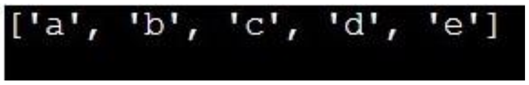
Develop a python code to apply Apriori algorithm for the following database transactions and generate strong association rules with minimum support count = 3 and minimum confidence = 80%.
(("a","b","c"), ("a","b"), ("a","b","d"), ("b","e"), ("b","c","e"), ("a","d","e"), ("a","c"), ("a","b","d"), ("c","e"), ("a","b","d","e"), ("a",'b','e','c')).

CODE:

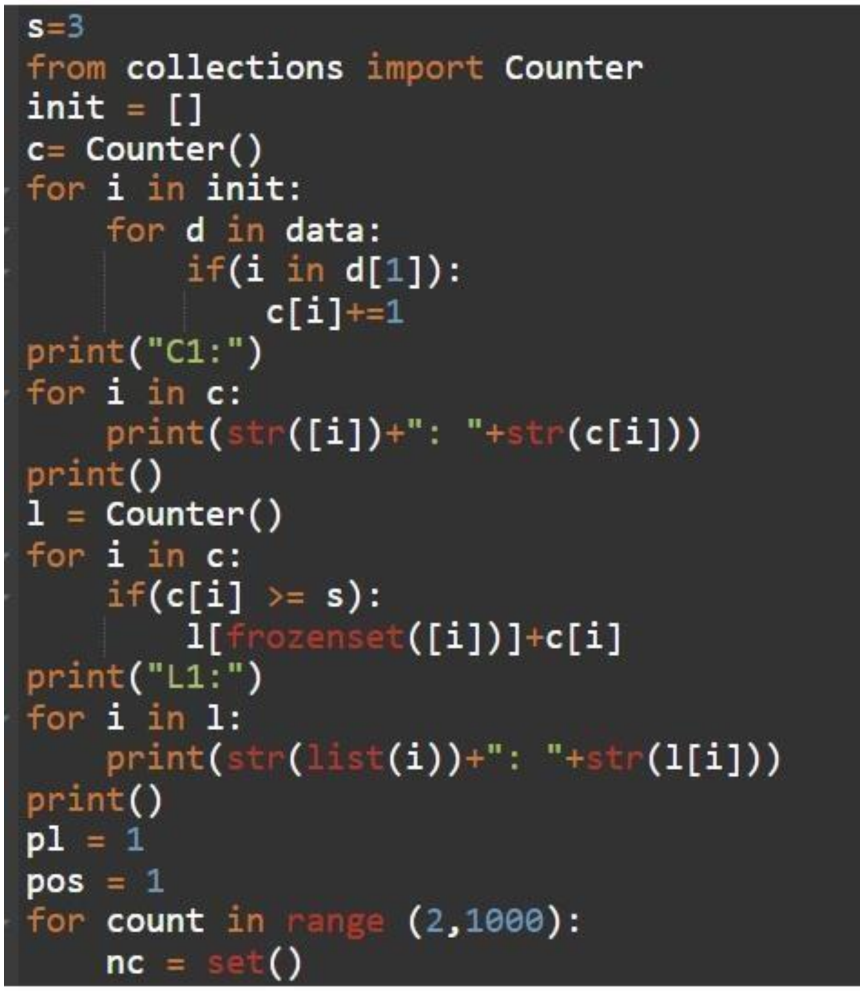


```
data= [
        ['T100',["a","b","c"]],
        ['T200',["a","b"]],
        ['T300',["a","b","d"]],
        ['T400',["b","e"]],
        ['T500',["b","c","e"]],
        ['T600',["a","d","e"]],
        ['T700',["a","c"]],
        ['T800',["a","b","d"]],
        ['T900',["c","e"]],
        ['T1000',["a","b","d","e"]],
        ['T1100',["a","b","e","c"]]
        ]


init = []
for i in data:
    for q in i[1]:
        if(q not in init):
            init.append(q)
init = sorted(init)
print(init)
```

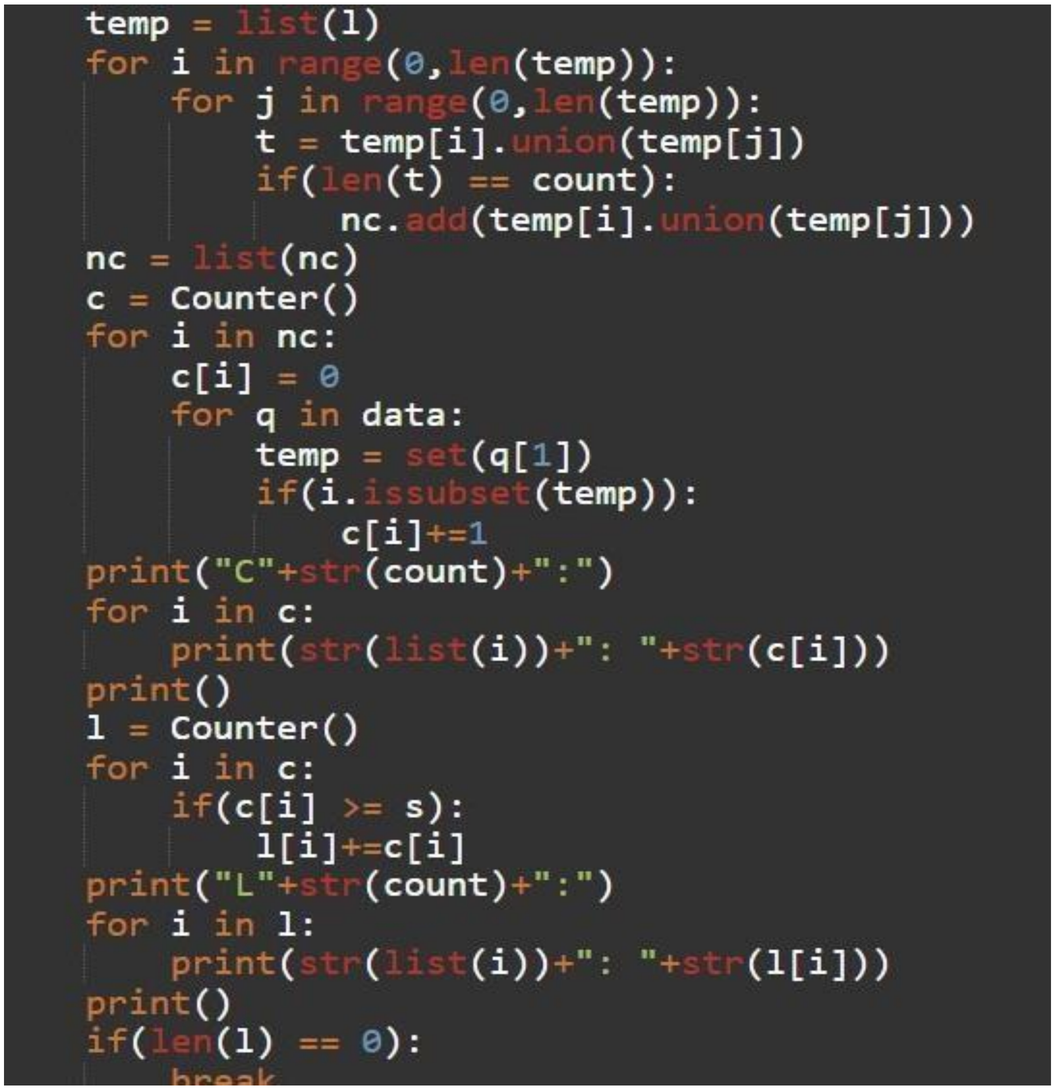
OUTPUT:



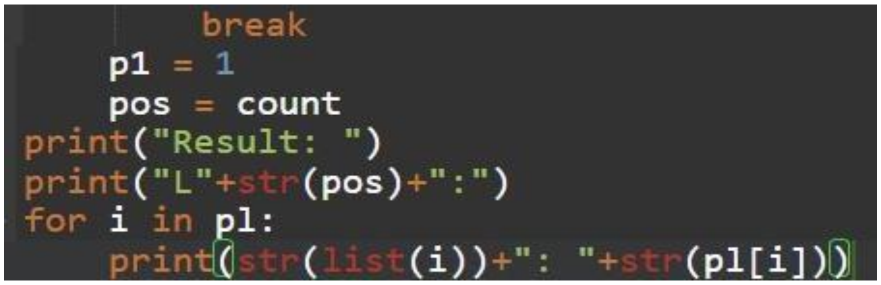
```
['a', 'b', 'c', 'd', 'e']
```



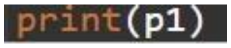
```
s=3
from collections import Counter
init = []
c= Counter()
for i in init:
    for d in data:
        if(i in d[1]):
            c[i]+=1
print("C1:")
for i in c:
    print(str([i])+": "+str(c[i]))
print()
l = Counter()
for i in c:
    if(c[i] >= s):
        l[frozenset([i])]+c[i]
print("L1:")
for i in l:
    print(str(list(i))+": "+str(l[i]))
print()
pl = 1
pos = 1
for count in range (2,1000):
    nc = set()
```



```
    temp = list(l)
    for i in range(0,len(temp)):
        for j in range(0,len(temp)):
            t = temp[i].union(temp[j])
            if(len(t) == count):
                nc.add(temp[i].union(temp[j]))
    nc = list(nc)
    c = Counter()
    for i in nc:
        c[i] = 0
        for q in data:
            temp = set(q[1])
            if(i.issubset(temp)):
                c[i]+=1
    print("C"+str(count)+":")
    for i in c:
        print(str(list(i))+": "+str(c[i]))
    print()
    l = Counter()
    for i in c:
        if(c[i] >= s):
            l[i]+=c[i]
    print("L"+str(count)+":")
    for i in l:
        print(str(list(i))+": "+str(l[i]))
    print()
    if(len(l) == 0):
```

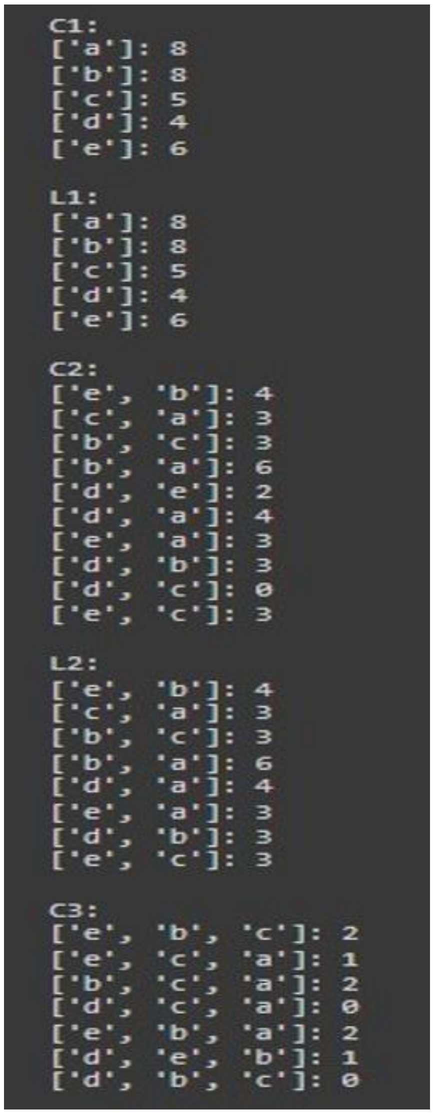


```
        break
    p1 = 1
    pos = count
print("Result: ")
print("L"+str(pos)+":")
for i in pl:
    print(str(list(i))+": "+str(pl[i]))
```

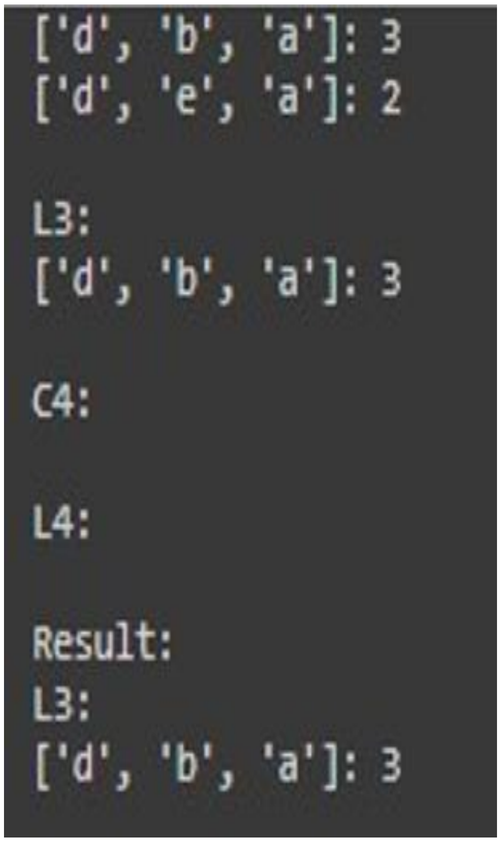


```
print(p1)
```

OUTPUT:

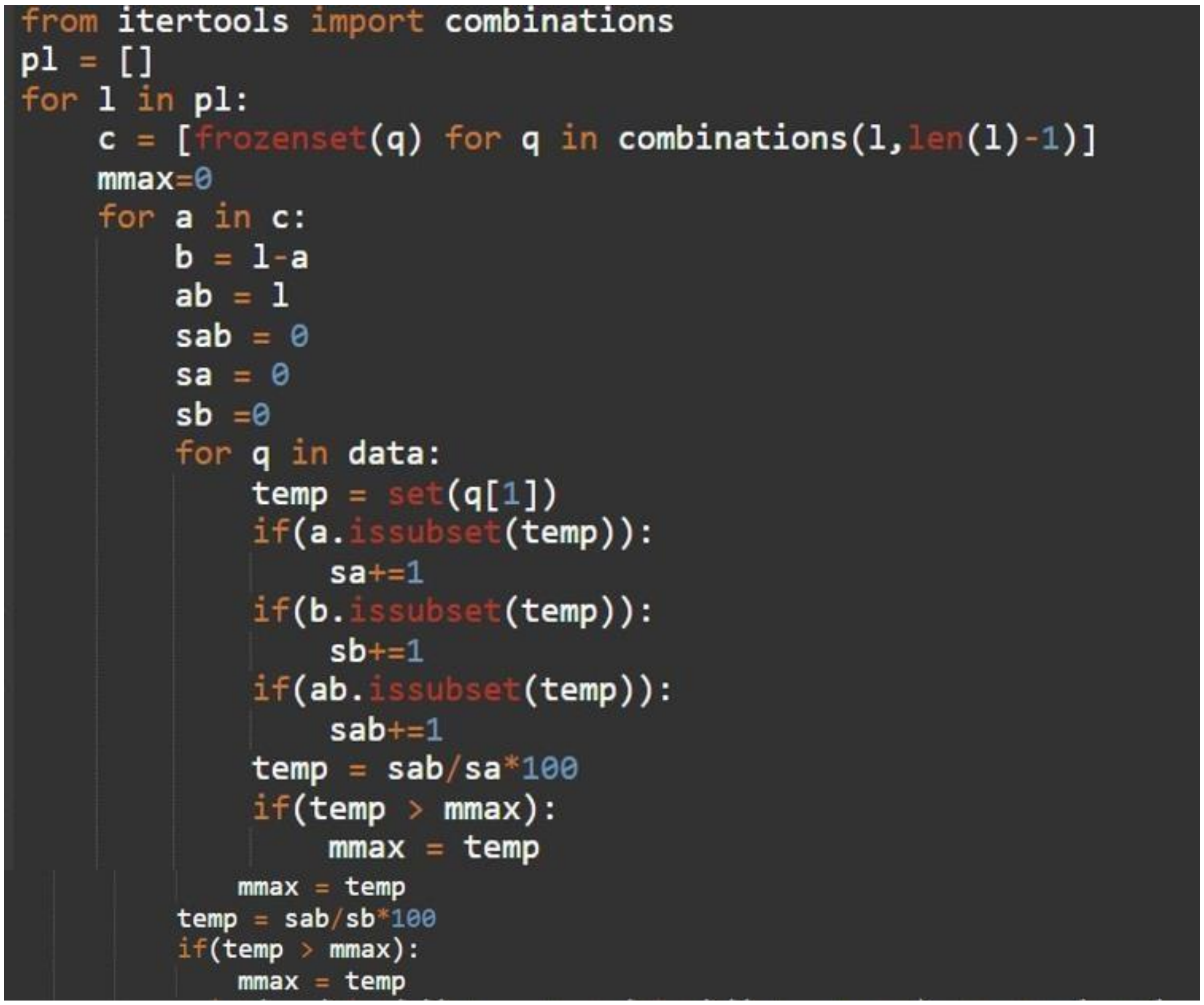


```
C1:
['a']: 8
['b']: 8
['c']: 5
['d']: 4
['e']: 6

L1:
['a']: 8
['b']: 8
['c']: 5
['d']: 4
['e']: 6

C2:
['e', 'b']: 4
['c', 'a']: 3
['b', 'c']: 3
['b', 'a']: 6
['d', 'e']: 2
['d', 'a']: 4
['e', 'a']: 3
['d', 'b']: 3
['d', 'c']: 0
['e', 'c']: 3

L2:
['e', 'b']: 4
['c', 'a']: 3
['b', 'c']: 3
['b', 'a']: 6
['d', 'a']: 4
['e', 'a']: 3
['d', 'b']: 3
['e', 'c']: 3

C3:
['e', 'b', 'c']: 2
['e', 'c', 'a']: 1
['b', 'c', 'a']: 2
['d', 'c', 'a']: 0
['e', 'b', 'a']: 2
['d', 'e', 'b']: 1
['d', 'b', 'c']: 0
```

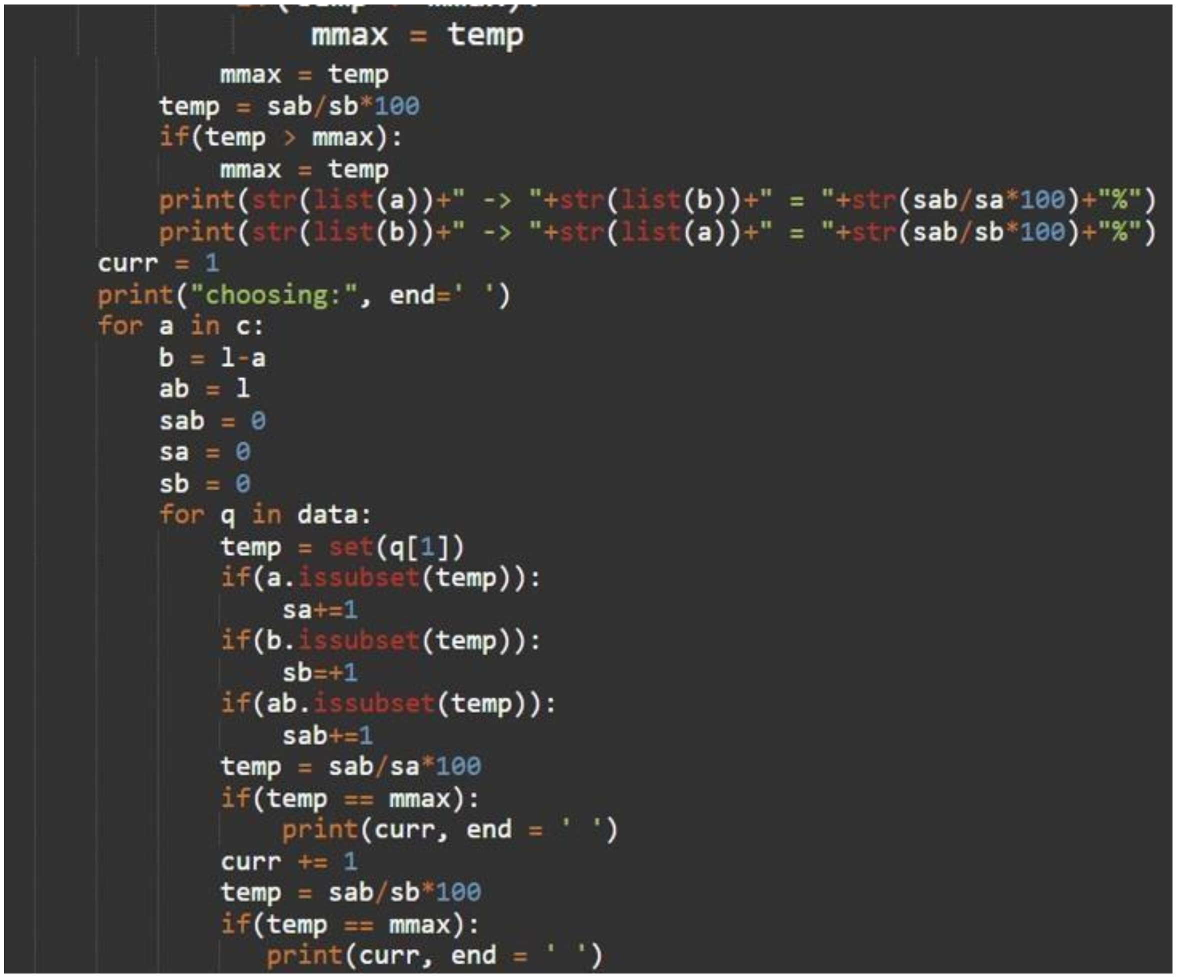


```
['d', 'b', 'a']: 3
['d', 'e', 'a']: 2

L3:
['d', 'b', 'a']: 3

C4:

L4:

Result:
L3:
['d', 'b', 'a']: 3
```

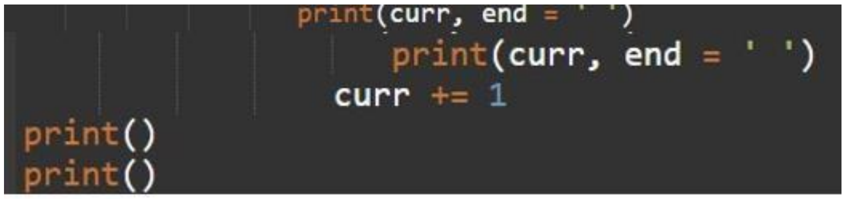
CODE :



```
from itertools import combinations
pl = []
for l in pl:
    c = [frozenset(q) for q in combinations(l,len(l)-1)]
    mmax=0
    for a in c:
        b = l-a
        ab = l
        sab = 0
        sa = 0
        sb =0
        for q in data:
            temp = set(q[1])
            if(a.issubset(temp)):
                sa+=1
            if(b.issubset(temp)):
                sb+=1
            if(ab.issubset(temp)):
                sab+=1
            temp = sab/sa*100
            if(temp > mmax):
                mmax = temp
                mmax = temp
            temp = sab/sb*100
            if(temp > mmax):
                mmax = temp
```

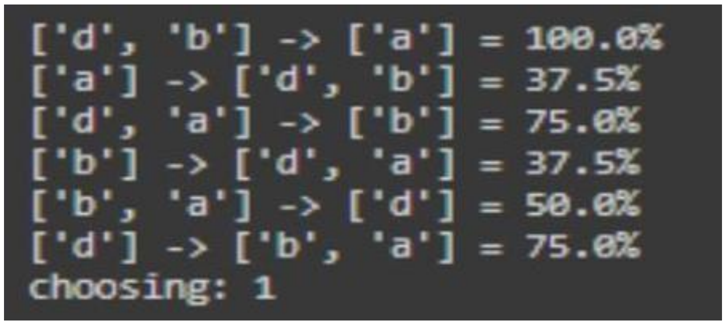


```
                mmax = temp
            mmax = temp
        temp = sab/sb*100
        if(temp > mmax):
            mmax = temp
        print(str(list(a))+" -> "+str(list(b))+" = "+str(sab/sa*100)+"%")
        print(str(list(b))+" -> "+str(list(a))+" = "+str(sab/sb*100)+"%")
    curr = 1
    print("choosing:", end=' ')
    for a in c:
        b = l-a
        ab = l
        sab = 0
        sa = 0
        sb = 0
        for q in data:
            temp = set(q[1])
            if(a.issubset(temp)):
                sa+=1
            if(b.issubset(temp)):
                sb=+1
            if(ab.issubset(temp)):
                sab+=1
        temp = sab/sa*100
        if(temp == mmax):
            print(curr, end = ' ')
        curr += 1
        temp = sab/sb*100
        if(temp == mmax):
            print(curr, end = ' ')
```



```
            print(curr, end = ' ')
        curr += 1
print()
print()
```

OUTPUT:



```
['d', 'b'] -> ['a'] = 100.0%
['a'] -> ['d', 'b'] = 37.5%
['d', 'a'] -> ['b'] = 75.0%
['b'] -> ['d', 'a'] = 37.5%
['b', 'a'] -> ['d'] = 50.0%
['d'] -> ['b', 'a'] = 75.0%
choosing: 1
```