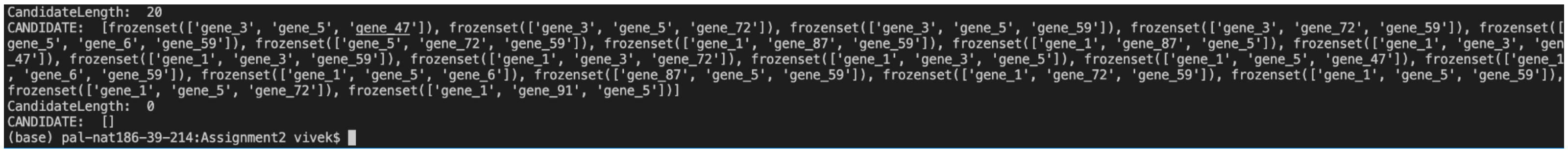
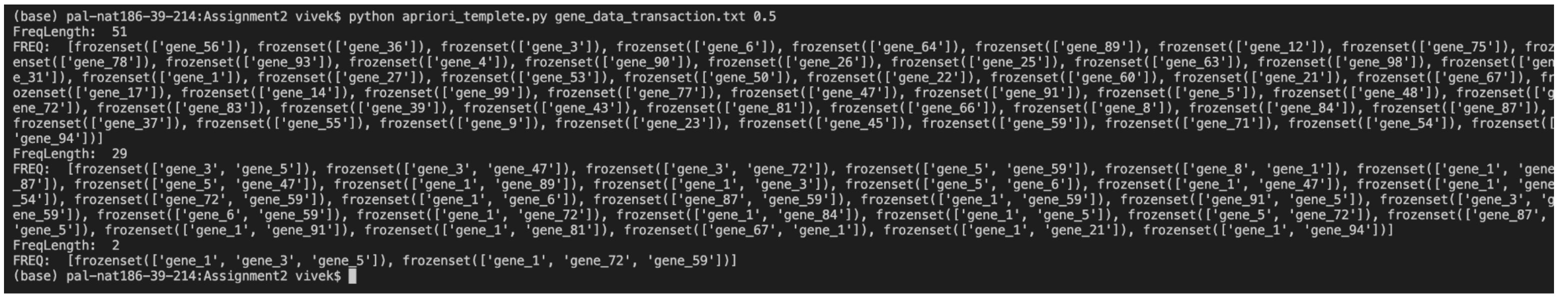
Vivek Khanolkar
ECE49595 - Data Mining
Assignment 2

Candidate list of 3 for gene:



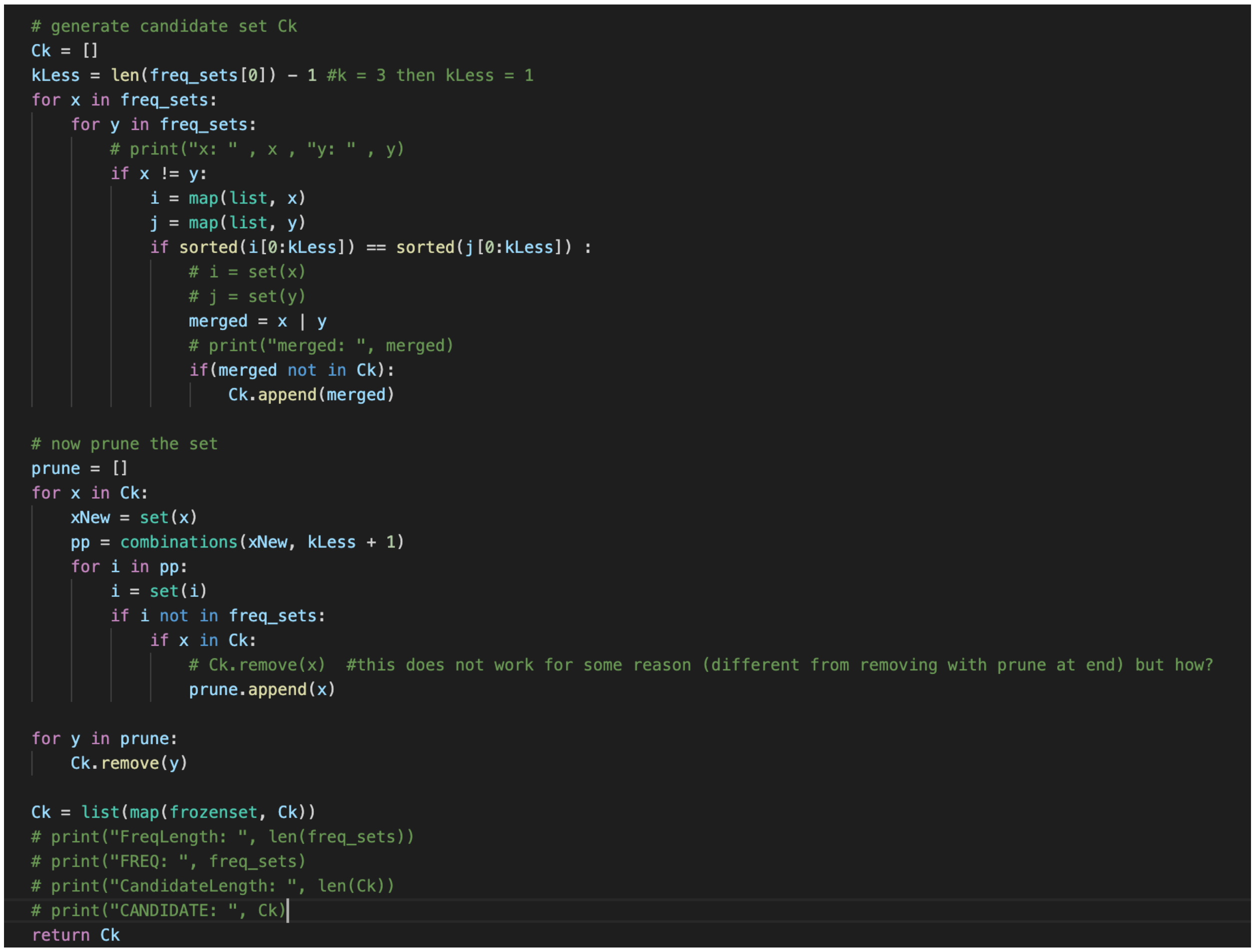
```
CandidateLength:  20
CANDIDATE:  [frozenset(['gene_3', 'gene_5', 'gene_47']), frozenset(['gene_3', 'gene_5', 'gene_72']), frozenset(['gene_3', 'gene_5', 'gene_59']), frozenset(['gene_3', 'gene_72', 'gene_59']), frozenset([
gene_5', 'gene_6', 'gene_59']), frozenset(['gene_5', 'gene_72', 'gene_59']), frozenset(['gene_1', 'gene_87', 'gene_59']), frozenset(['gene_1', 'gene_87', 'gene_5']), frozenset(['gene_1', 'gene_3', 'gen
_47']), frozenset(['gene_1', 'gene_3', 'gene_59']), frozenset(['gene_1', 'gene_3', 'gene_72']), frozenset(['gene_1', 'gene_3', 'gene_5']), frozenset(['gene_1', 'gene_5', 'gene_47']), frozenset(['gene_1
, 'gene_6', 'gene_59']), frozenset(['gene_1', 'gene_5', 'gene_6']), frozenset(['gene_87', 'gene_5', 'gene_59']), frozenset(['gene_1', 'gene_72', 'gene_59']), frozenset(['gene_1', 'gene_5', 'gene_59']),
frozenset(['gene_1', 'gene_5', 'gene_72']), frozenset(['gene_1', 'gene_91', 'gene_5'])]
CandidateLength:  0
CANDIDATE:  []
(base) pal-nat186-39-214:Assignment2 vivek$ 
```

Frequency lists for gene:



```
(base) pal-nat186-39-214:Assignment2 vivek$ python apriori_templete.py gene_data_transaction.txt 0.5
FreqLength:  51
FREQ:  [frozenset(['gene_56']), frozenset(['gene_36']), frozenset(['gene_3']), frozenset(['gene_6']), frozenset(['gene_64']), frozenset(['gene_89']), frozenset(['gene_12']), frozenset(['gene_75']), froz
enset(['gene_78']), frozenset(['gene_93']), frozenset(['gene_4']), frozenset(['gene_90']), frozenset(['gene_26']), frozenset(['gene_25']), frozenset(['gene_63']), frozenset(['gene_98']), frozenset(['gen
e_31']), frozenset(['gene_1']), frozenset(['gene_27']), frozenset(['gene_53']), frozenset(['gene_50']), frozenset(['gene_22']), frozenset(['gene_60']), frozenset(['gene_21']), frozenset(['gene_67']), fr
ozenset(['gene_17']), frozenset(['gene_14']), frozenset(['gene_99']), frozenset(['gene_77']), frozenset(['gene_47']), frozenset(['gene_91']), frozenset(['gene_5']), frozenset(['gene_48']), frozenset(['g
ene_72']), frozenset(['gene_83']), frozenset(['gene_39']), frozenset(['gene_43']), frozenset(['gene_81']), frozenset(['gene_66']), frozenset(['gene_8']), frozenset(['gene_84']), frozenset(['gene_87']),
frozenset(['gene_37']), frozenset(['gene_55']), frozenset(['gene_9']), frozenset(['gene_23']), frozenset(['gene_45']), frozenset(['gene_59']), frozenset(['gene_71']), frozenset(['gene_54']), frozenset([
'gene_94'])]
FreqLength:  29
FREQ:  [frozenset(['gene_3', 'gene_5']), frozenset(['gene_3', 'gene_47']), frozenset(['gene_3', 'gene_72']), frozenset(['gene_5', 'gene_59']), frozenset(['gene_8', 'gene_1']), frozenset(['gene_1', 'gene
_87']), frozenset(['gene_5', 'gene_47']), frozenset(['gene_1', 'gene_89']), frozenset(['gene_1', 'gene_3']), frozenset(['gene_5', 'gene_6']), frozenset(['gene_1', 'gene_47']), frozenset(['gene_1', 'gene
_54']), frozenset(['gene_72', 'gene_59']), frozenset(['gene_1', 'gene_6']), frozenset(['gene_87', 'gene_59']), frozenset(['gene_1', 'gene_59']), frozenset(['gene_91', 'gene_5']), frozenset(['gene_3', 'g
ene_59']), frozenset(['gene_6', 'gene_59']), frozenset(['gene_1', 'gene_72']), frozenset(['gene_1', 'gene_84']), frozenset(['gene_1', 'gene_5']), frozenset(['gene_5', 'gene_72']), frozenset(['gene_87',
'gene_5']), frozenset(['gene_1', 'gene_91']), frozenset(['gene_1', 'gene_81']), frozenset(['gene_67', 'gene_1']), frozenset(['gene_1', 'gene_21']), frozenset(['gene_1', 'gene_94'])]
FreqLength:  2
FREQ:  [frozenset(['gene_1', 'gene_3', 'gene_5']), frozenset(['gene_1', 'gene_72', 'gene_59'])]
(base) pal-nat186-39-214:Assignment2 vivek$ 
```

Code is below.

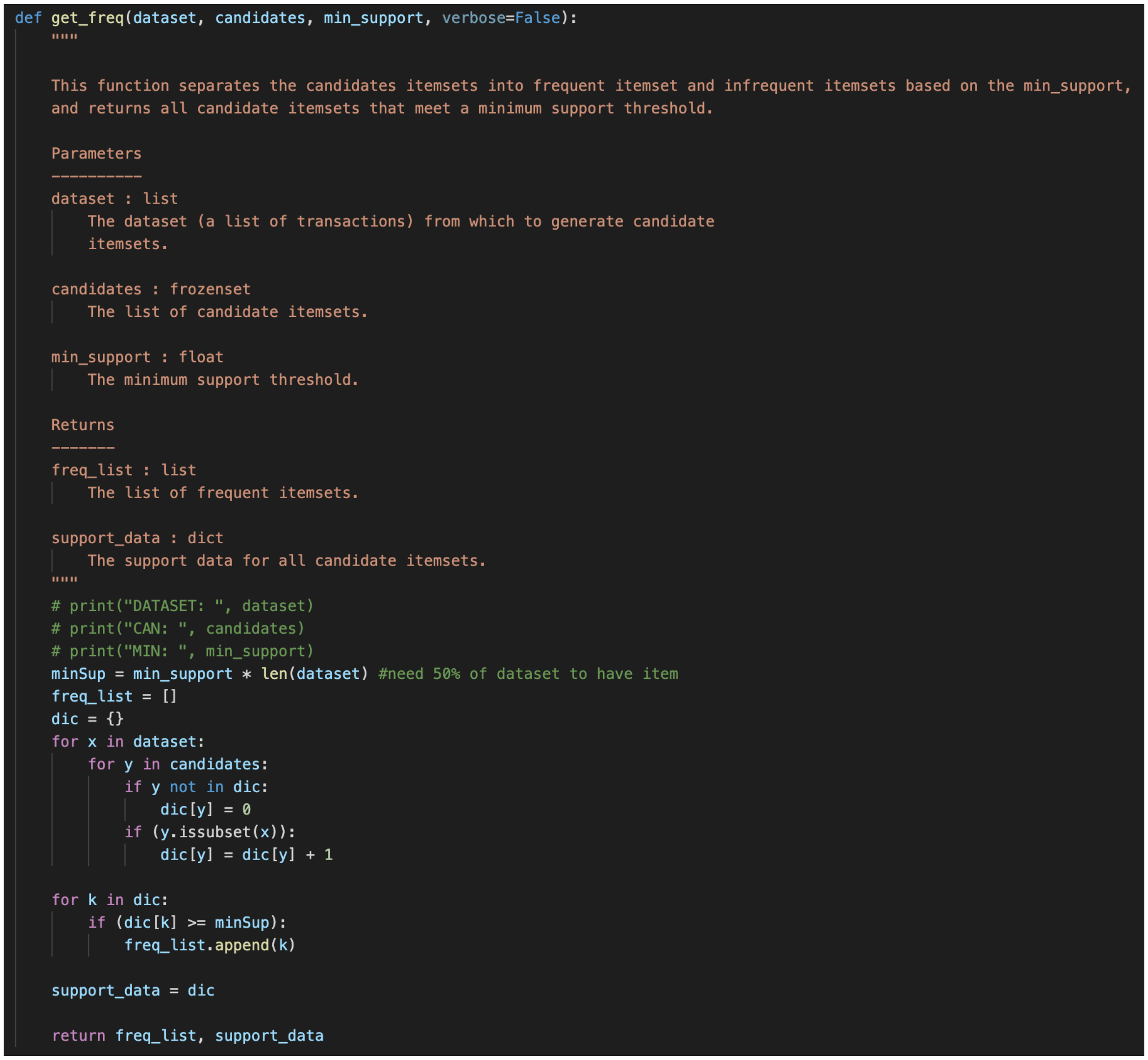


```
# generate candidate set Ck
Ck = []
kLess = len(freq_sets[0]) - 1 #k = 3 then kLess = 1
for x in freq_sets:
    for y in freq_sets:
        # print("x: " , x , "y: " , y)
        if x != y:
            i = map(list, x)
            j = map(list, y)
            if sorted(i[0:kLess]) == sorted(j[0:kLess]) :
                # i = set(x)
                # j = set(y)
                merged = x | y
                # print("merged: ", merged)
                if(merged not in Ck):
                    Ck.append(merged)

# now prune the set
prune = []
for x in Ck:
    xNew = set(x)
    pp = combinations(xNew, kLess + 1)
    for i in pp:
        i = set(i)
        if i not in freq_sets:
            if x in Ck:
                # Ck.remove(x)  #this does not work for some reason (different from removing with prune at end) but how?
                prune.append(x)

for y in prune:
    Ck.remove(y)

Ck = list(map(frozenset, Ck))
# print("FreqLength: ", len(freq_sets))
# print("FREQ: ", freq_sets)
# print("CandidateLength: ", len(Ck))
# print("CANDIDATE: ", Ck)
return Ck
```



```python
def get_freq(dataset, candidates, min_support, verbose=False):
    """

    This function separates the candidates itemsets into frequent itemset and infrequent itemsets based on the min_support,
    and returns all candidate itemsets that meet a minimum support threshold.

    Parameters
    ----------
    dataset : list
        The dataset (a list of transactions) from which to generate candidate
        itemsets.

    candidates : frozenset
        The list of candidate itemsets.

    min_support : float
        The minimum support threshold.

    Returns
    -------
    freq_list : list
        The list of frequent itemsets.

    support_data : dict
        The support data for all candidate itemsets.
    """
    # print("DATASET: ", dataset)
    # print("CAN: ", candidates)
    # print("MIN: ", min_support)
    minSup = min_support * len(dataset) #need 50% of dataset to have item
    freq_list = []
    dic = {}
    for x in dataset:
        for y in candidates:
            if y not in dic:
                dic[y] = 0
            if (y.issubset(x)):
                dic[y] = dic[y] + 1

    for k in dic:
        if (dic[k] >= minSup):
            freq_list.append(k)

    support_data = dic

    return freq_list, support_data
```