

Week 1 – HomeWork 1

Paritosh Rai



7335 Machine Learning 2



HW

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\parit> conda activate ML1

PS C:\Users\parit> & C:/Users/parit/Anaconda/envs/ML1/python.exe

Python 3.7.1 (default, Oct 28 2018, 08:39:03) [MSC v.1912 64 bit (AMD64)] :: Anaconda, Inc. on win32

Python Code:

# please fill in an explanation of each function and an example of how to use it below.

# List

#append() => Add an elemnet to list

colors = ['Red','Green','Blue']

print(colors)

colors.append('Black')

print(colors)

#extend() => Add elements of one list to another list

colors = ['Red','Green','Blue']

print(colors)

car = ['Honda', 'BMW', 'Audi']

print(car)

colors.extend(car)

print(colors)

# index() => Retuens the position of specif element. Python is zero based indexing

vowels = ['a', 'e', 'i', 'o', 'u', 'i', 'a', 'i']

# Index of 'u' in the list

index1 = vowels.index('u')

print("The index of 'u' is",index1)

index2 = vowels.index('i') # index of 1st i will be returened

print("The index of 'i' is",index2)

# index(value, integer) Find the index or position of elemnet in list in given range. Range is defined as start and stop.

#index(value,start,stop)

#vowels = ['a', 'e', 'i', 'o', 'u', 'i', 'a', 'i']

#index3 = vowels.index('a',1,6)

# we expect error 'a' does not exist between index 1 and 5.

# Start of index is included end is not included

#print("The index of 'a' is",index3)

index3 = vowels.index('i',1,7)

print("The index of 'i' is",index3)

index4 = vowels.index('i', 3)

print("The index of 'i' after 4th position is",index4)

#insert(position) => Insert an element at given position

fruit = ['apple', 'mango', 'banana']

fruit.insert(1, 'orange') # insert only one element. Use extend() to merge two list

print(fruit)

#remove() => remove the specifed element

fruit = ['apple', 'mango', 'banana']

fruit.remove('mango')

print(fruit)

#pop() => use pop to remove specified index pop()

fruit = ['apple', 'mango', 'banana']

fruit.pop(1) #remove 2nd element from the list

print(fruit)

#pop(0) => Remove 1st element from the list

fruit = ['apple', 'mango', 'banana']

fruit.pop(0) #remove 1st element from the list

print(fruit)

#count() => Returns number of times element appeared in the list

alphabet = ['a','b','c','d','b','d','c','b','b']

x=alphabet.count('b')

print(x)

#reverse() => reverse the elements of the in place

apple =['a','p','p','l','e']

apple.reverse()

print(apple)

#sort => Sort the order of the item in place

number = [2,4,5,6,4,7,4,1,6,9,3,8]

number.sort()

print('New sorted list is:',number)

#sorted

number = [2,4,5,6,4,7,4,1,6,9,3,8]

s\_num=sorted(number,key=None,reverse=False)

print(s\_num)

# [1]+[1] => combine two list to one (Concatenation)

join\_lst = [1] + [1]

print(join\_lst)

#[2]\*2 => create a list of elemnets by repeating the elements multiple times.

# In this case repeating number 2 twice (Repetition).

list\_2=[2]\*2

print(list\_2)

list\_5=[2]\*5 # repeate 2 in the list 5 times

print(list\_5)

# [1,2][1:] => Slicing the list. In this case select all elements after (including) index 1.

[1,2][1:]

# Slicing can be done within range. select elemnets with nedex 1 to 3. Offset start at zero => [list][start:stop]

[1,2,5,7,8,9][1:4]

# Slicing can be done using Negative: count from the right

[1,2,5,7,8,9][-2] # second number from the right

#[x for x in [2,3]] => This is used for list comprehension. You to comprehend list based on existing list

#good replacemnet for "for" loop

square=[(x\*\*2) for x in [2,3,4]] # square the numbers in existing list

print(square)

# [x for x in [1,2] if x ==1] => This helps to put in condition with list comprehension

#conditinal for loop

even\_sq = [(x\*\*2) for x in [1,2,3,4,5,6,7] if x%2 ==0] #square the numbers that are even

print(even\_sq)

#[y\*2 for x in [[1,2],[3,4]] for y in x] => nested comprehension sort of nexted loop

[y\*2 for x in [[1,2],[3,4]] for y in x] #create on list combining two list and multiply by two

# Tuple

#count() => search the given element in tuple and count its occurance.

alphabet = ('a','b','c','d','b','d','c','b','b')

x=alphabet.count('b')

print(x)

#index() => find the location of the element in tuple

alphabet = ('a','b','c','d','b','d','c','b','b')

#alphabet.index('c') # location of element "c"

print("location of elemnet 'c' is: ", alphabet.index('c'))

#build a dictionary from tuples

tuple1 = (('a','apple'), ('b','ball'))

dict(tuple1)

#unpack tuples => extracting the values from tuple

names = ('John', 'Mary', 'Tim', 'Ann')

(alpha, beta, gamma, delta) = names

print(alpha)

print(beta)

print(gamma)

print(delta)

#Dicts:

a\_dict = {'I hate':'you', 'You should':'leave'} # create dictionary

# two key elements key and value(s)

car ={

    'brand':'Ford',

    'model':'mustang',

    'year' : 1964

}

print(car)

#values() => return the values from dictionary

car ={

    'brand':'Ford',

    'model':'mustang',

    'year' : 1964

}

v = car.values()

print('Values in the car dictionary is:',v)

#keys() => returns the key from dictionary

car ={

    'brand':'Ford',

    'model':'mustang',

    'year' : 1964

}

k = car.keys()

print('Values in the car dictionary is:',k)

#items()

#keys() => returns the key and value pairs from dictionary

car ={

    'brand':'Ford',

    'model':'mustang',

    'year' : 1964

}

i = car.items()

print(i)

# has\_key() => search key in the dictionary return true if key is available else return false

#car ={

    #'brand':'Ford',

    #'model':'mustang',

    #'year' : 1964

#}

#print(car.has\_key('year'))

#print(car.has\_key('color'))

# The has\_key () method has been omitted in Python 3.x versions and hence can be only used in older versions.

#‘never’ in a\_dict => Can not

a\_dict = dict()

'never' in a\_dict

a\_dict = dict()

'never' in a\_dict

a\_dict ={

    'never':'yes'

} # add never to a\_dict

'never' in a\_dict

# del a\_dict['me'] => Remove the item from with specified key name

car ={

    'brand':'Ford',

    'model':'mustang',

    'year' : 1964

}

del car['year'] # removed key 'year'

print(car) # updated dictionary

#a\_dict.clear() => empties the dictionary empties the dictionary

car ={

    'brand':'Ford',

    'model':'mustang',

    'year' : 1964

}

car.clear()

print(car)

#sets => collection of unordered and unindexed elements

new\_set={'Don','John','Ron'} # create set

print(new\_set)

# for x in set => search in set and say Ture if item is prersent else false

new\_set={'Don','John','Ron'}

x = 'Ron' in new\_set

y = 'Ram' in new\_set

print(x)

print(y)

#set.add() => Add item to set

new\_set={'Don','John','Ron'}

new\_set.add('Ram')

print(new\_set)

#set.remove() => Remove items from the set

new\_set={'Don','John','Ron'}

new\_set.remove('Ron')

print(new\_set)

#set1.union(set2) => Returns all items in both the sets

set1 = {'Don','John','Ron'}

set2 = {'10','20','30'}

new\_set = set1.union(set2)

print(new\_set)

#Strings

#creat string

a ='Hello World'

print(a)

#len() => String length

a = 'Hello World'

print(len(a))

# Slicing Strings str[start:End]

a = 'Hello World'

print(a[2:7])

#str.strip() => Remove Whitespace  from beginning and the end of the string

a = '   Hello World   '

print(a.strip())

# str.replace() => replace elements of string from other element

a = '   Hello World   '

print(a.replace('H','J')) #replace H from J

# str.split() => Split the string in substrings

a = 'Hello, World!'

print(a.split(','))

# from collections import Counter => Use to stroe information in Memory

# counter keep track of how many times equivalent values are added

# imprt Collection => Importing this module is critical for collection

# collection.xyz

#Creat and Update Counter

from collections import Counter  # import module

lang\_list = ['C++', 'Python', 'R', 'Java', 'C', 'C++', 'Java','Python', 'R','R', 'C++'] # create a list

print(Counter(lang\_list)) # count nuber of languages in lang\_list

#count specific element => Counter(list)['a']

from collections import Counter  # import module

lang\_list = ['C++', 'Python', 'R', 'Java', 'C', 'C++', 'Java','Python', 'R','R', 'C++']

c=Counter(lang\_list)

print('The occurance of python is:',c['Python']) # count how many times python w]occured in list

#Count in a String => count number of each element in string

from collections import Counter  # import module

str = "Hello World"

c=Counter(str) #count elemnets in string

print(c)

#Count single charter in String

from collections import Counter  # import module

str = "Hello World"

c=Counter(str)

print("The occurance of 'l' in string is:",c['l']) # count number of time 'c' appeared in the string

# using counter with tuple

from collections import Counter  # import module

tup\_num = (8,9,8,7,6,5,7,9,4,7,8,5)

c=Counter(tup\_num)

print(c)

#Arthmetic operation with counter

# using counter with tuple

from collections import Counter  # import module

tup\_num1 = (8,9,8,7,6,5,7,9,4,7,8,5)

tup\_num2 = (9,9,9,7,8,7,6,7,5,8,3)

c1=Counter(tup\_num1)

c2=Counter(tup\_num2)

print(c1+c2)

print(c1-c2)

print(c1&c2)

# from itertools import => use to implemnet a number of iterator building blocks by constructs

# it is a memory efficent tool

#zip objects

from itertools import \*

import itertools

Counter = itertools.count()

data =[10,20,30,40] # data point

sl\_data = list(zip(itertools.count(),data)) # zip (combine tow data)

print(sl\_data)

# Count itertool

from itertools import \*

import itertools

for i in itertools.count(42,8): # In this code, 42 is the starting point and 8 is the step

    print(i)

    if i > 60: # stops at 60

        break

#combinations => iterate all possible combination of list

from itertools import \*

import itertools

let = ['a', 'b', 'c', 'd']

res=itertools.combinations(let,2) # all possibl combination of two letters order does not matter

for item in res:

    print(item)

#permutations => iterate all possible combination of list where order does matter

from itertools import \*

import itertools

let = ['a', 'b', 'c', 'd']

res=itertools.permutations(let,2) # all possibl combination of two letters

for item in res:

    print(item)

# flower\_orders

# 1. Build a counter object and use the counter and confirm they have the same values.

from collections import Counter  # import module

flower\_orders = ['W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','B/Y','B/Y','B/Y','B/Y','B/Y','R/B/Y','R/B/Y','R/B/Y','R/B/Y','R/B/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/G','W/G','W/G','W/G','R/Y','R/Y','R/Y','R/Y','N/R/V/Y','N/R/V/Y','N/R/V/Y','N/R/V/Y','W/R/B/V','W/R/B/V','W/R/B/V','W/R/B/V','W/N/R/V/Y','W/N/R/V/Y','W/N/R/V/Y','W/N/R/V/Y','N/R/Y','N/R/Y','N/R/Y','W/V/O','W/V/O','W/V/O','W/N/R/Y','W/N/R/Y','W/N/R/Y','R/B/V/Y','R/B/V/Y','R/B/V/Y','W/R/V/Y','W/R/V/Y','W/R/V/Y','W/R/B/V/Y','W/R/B/V/Y','W/R/B/V/Y','W/N/R/B/Y','W/N/R/B/Y','W/N/R/B/Y','R/G','R/G','B/V/Y','B/V/Y','N/B/Y','N/B/Y','W/B/Y','W/B/Y','W/N/B','W/N/B','W/N/R','W/N/R','W/N/B/Y','W/N/B/Y','W/B/V/Y','W/B/V/Y','W/N/R/B/V/Y/G/M','W/N/R/B/V/Y/G/M','B/R','N/R','V/Y','V','N/R/V','N/V/Y','R/B/O','W/B/V','W/V/Y','W/N/R/B','W/N/R/O','W/N/R/G','W/N/V/Y','W/N/Y/M','N/R/B/Y','N/B/V/Y','R/V/Y/O','W/B/V/M','W/B/V/O','N/R/B/Y/M','N/R/V/O/M','W/N/R/Y/G','N/R/B/V/Y','W/R/B/V/Y/P','W/N/R/B/Y/G','W/N/R/B/V/O/M','W/N/R/B/V/Y/M','W/N/B/V/Y/G/M','W/N/B/V/V/Y/P']

c =Counter(flower\_orders) # count each color combination

print(c)

# Count color combination using 'for' function without using 'counter module'

c1=dict((x,flower\_orders.count(x)) for x in set(flower\_orders)) # use set to get unique values

print(c1)

# 2. Count how many objects have color W in them.

from collections import Counter  # import module

flower\_orders = ['W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','B/Y','B/Y','B/Y','B/Y','B/Y','R/B/Y','R/B/Y','R/B/Y','R/B/Y','R/B/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/G','W/G','W/G','W/G','R/Y','R/Y','R/Y','R/Y','N/R/V/Y','N/R/V/Y','N/R/V/Y','N/R/V/Y','W/R/B/V','W/R/B/V','W/R/B/V','W/R/B/V','W/N/R/V/Y','W/N/R/V/Y','W/N/R/V/Y','W/N/R/V/Y','N/R/Y','N/R/Y','N/R/Y','W/V/O','W/V/O','W/V/O','W/N/R/Y','W/N/R/Y','W/N/R/Y','R/B/V/Y','R/B/V/Y','R/B/V/Y','W/R/V/Y','W/R/V/Y','W/R/V/Y','W/R/B/V/Y','W/R/B/V/Y','W/R/B/V/Y','W/N/R/B/Y','W/N/R/B/Y','W/N/R/B/Y','R/G','R/G','B/V/Y','B/V/Y','N/B/Y','N/B/Y','W/B/Y','W/B/Y','W/N/B','W/N/B','W/N/R','W/N/R','W/N/B/Y','W/N/B/Y','W/B/V/Y','W/B/V/Y','W/N/R/B/V/Y/G/M','W/N/R/B/V/Y/G/M','B/R','N/R','V/Y','V','N/R/V','N/V/Y','R/B/O','W/B/V','W/V/Y','W/N/R/B','W/N/R/O','W/N/R/G','W/N/V/Y','W/N/Y/M','N/R/B/Y','N/B/V/Y','R/V/Y/O','W/B/V/M','W/B/V/O','N/R/B/Y/M','N/R/V/O/M','W/N/R/Y/G','N/R/B/V/Y','W/R/B/V/Y/P','W/N/R/B/Y/G','W/N/R/B/V/O/M','W/N/R/B/V/Y/M','W/N/B/V/Y/G/M','W/N/B/V/V/Y/P']

print("Number of color combination with 'W':",len([x for x in flower\_orders if "W" in x])) # using for function to count color combination with 'W'

# 3. Make histogram of colors

# let's build a dict with counts of colors

# somehow it not working in VScode. but working fine in Jupiternotebook

from collections import Counter  # import counter module

import matplotlib.pyplot as plt # import matplotlib

flower\_orders = ['W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','B/Y','B/Y','B/Y','B/Y','B/Y','R/B/Y','R/B/Y','R/B/Y','R/B/Y','R/B/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/G','W/G','W/G','W/G','R/Y','R/Y','R/Y','R/Y','N/R/V/Y','N/R/V/Y','N/R/V/Y','N/R/V/Y','W/R/B/V','W/R/B/V','W/R/B/V','W/R/B/V','W/N/R/V/Y','W/N/R/V/Y','W/N/R/V/Y','W/N/R/V/Y','N/R/Y','N/R/Y','N/R/Y','W/V/O','W/V/O','W/V/O','W/N/R/Y','W/N/R/Y','W/N/R/Y','R/B/V/Y','R/B/V/Y','R/B/V/Y','W/R/V/Y','W/R/V/Y','W/R/V/Y','W/R/B/V/Y','W/R/B/V/Y','W/R/B/V/Y','W/N/R/B/Y','W/N/R/B/Y','W/N/R/B/Y','R/G','R/G','B/V/Y','B/V/Y','N/B/Y','N/B/Y','W/B/Y','W/B/Y','W/N/B','W/N/B','W/N/R','W/N/R','W/N/B/Y','W/N/B/Y','W/B/V/Y','W/B/V/Y','W/N/R/B/V/Y/G/M','W/N/R/B/V/Y/G/M','B/R','N/R','V/Y','V','N/R/V','N/V/Y','R/B/O','W/B/V','W/V/Y','W/N/R/B','W/N/R/O','W/N/R/G','W/N/V/Y','W/N/Y/M','N/R/B/Y','N/B/V/Y','R/V/Y/O','W/B/V/M','W/B/V/O','N/R/B/Y/M','N/R/V/O/M','W/N/R/Y/G','N/R/B/V/Y','W/R/B/V/Y/P','W/N/R/B/Y/G','W/N/R/B/V/O/M','W/N/R/B/V/Y/M','W/N/B/V/Y/G/M','W/N/B/V/V/Y/P']

orders\_string = '/'.join([elem for elem in flower\_orders]) # create a string use'/' to join to elements

#print(orders\_string)

colors\_list = orders\_string.split('/') # seperate each color. Use '/' as seperator

colors\_dict = dict(Counter(colors\_list)) # Use counter to count each elemnet and add to dictonary.

print(colors\_dict)

plt.bar(colors\_dict.keys(), colors\_dict.values()) # create bar chart using metplotlib

plt.show() # show bar chart in pop window

# 4. Rank the pairs of colors in each order regardless of how many colors are in an order.

#we will use combinations from itertools for this; for each order, we will

#remove slash, get unique colors, sort them, and then get combinations

flower\_orders = ['W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','B/Y','B/Y','B/Y','B/Y','B/Y','R/B/Y','R/B/Y','R/B/Y','R/B/Y','R/B/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/G','W/G','W/G','W/G','R/Y','R/Y','R/Y','R/Y','N/R/V/Y','N/R/V/Y','N/R/V/Y','N/R/V/Y','W/R/B/V','W/R/B/V','W/R/B/V','W/R/B/V','W/N/R/V/Y','W/N/R/V/Y','W/N/R/V/Y','W/N/R/V/Y','N/R/Y','N/R/Y','N/R/Y','W/V/O','W/V/O','W/V/O','W/N/R/Y','W/N/R/Y','W/N/R/Y','R/B/V/Y','R/B/V/Y','R/B/V/Y','W/R/V/Y','W/R/V/Y','W/R/V/Y','W/R/B/V/Y','W/R/B/V/Y','W/R/B/V/Y','W/N/R/B/Y','W/N/R/B/Y','W/N/R/B/Y','R/G','R/G','B/V/Y','B/V/Y','N/B/Y','N/B/Y','W/B/Y','W/B/Y','W/N/B','W/N/B','W/N/R','W/N/R','W/N/B/Y','W/N/B/Y','W/B/V/Y','W/B/V/Y','W/N/R/B/V/Y/G/M','W/N/R/B/V/Y/G/M','B/R','N/R','V/Y','V','N/R/V','N/V/Y','R/B/O','W/B/V','W/V/Y','W/N/R/B','W/N/R/O','W/N/R/G','W/N/V/Y','W/N/Y/M','N/R/B/Y','N/B/V/Y','R/V/Y/O','W/B/V/M','W/B/V/O','N/R/B/Y/M','N/R/V/O/M','W/N/R/Y/G','N/R/B/V/Y','W/R/B/V/Y/P','W/N/R/B/Y/G','W/N/R/B/V/O/M','W/N/R/B/V/Y/M','W/N/B/V/Y/G/M','W/N/B/V/V/Y/P']

from itertools import \* #impot itertools

from collections import Counter  # import counter module

tuple\_pairs2 = [x for y in flower\_orders for x in combinations(''.join(sorted(list(set(y.replace("/",""))))), 2)] # creat tuple with pair of two

#print(tuple\_pairs2)

tuple\_pairs2\_counts = Counter(tuple\_pairs2)

print("Rank the pairs of colors")

print(tuple\_pairs2\_counts)

# 5 Rank the triplets of colors in each order regardless of how many colors are in an order.

#we will use combinations from itertools for this; for each order, we will

#remove slash, get unique colors, sort them, and then get combinations

flower\_orders = ['W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','B/Y','B/Y','B/Y','B/Y','B/Y','R/B/Y','R/B/Y','R/B/Y','R/B/Y','R/B/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/G','W/G','W/G','W/G','R/Y','R/Y','R/Y','R/Y','N/R/V/Y','N/R/V/Y','N/R/V/Y','N/R/V/Y','W/R/B/V','W/R/B/V','W/R/B/V','W/R/B/V','W/N/R/V/Y','W/N/R/V/Y','W/N/R/V/Y','W/N/R/V/Y','N/R/Y','N/R/Y','N/R/Y','W/V/O','W/V/O','W/V/O','W/N/R/Y','W/N/R/Y','W/N/R/Y','R/B/V/Y','R/B/V/Y','R/B/V/Y','W/R/V/Y','W/R/V/Y','W/R/V/Y','W/R/B/V/Y','W/R/B/V/Y','W/R/B/V/Y','W/N/R/B/Y','W/N/R/B/Y','W/N/R/B/Y','R/G','R/G','B/V/Y','B/V/Y','N/B/Y','N/B/Y','W/B/Y','W/B/Y','W/N/B','W/N/B','W/N/R','W/N/R','W/N/B/Y','W/N/B/Y','W/B/V/Y','W/B/V/Y','W/N/R/B/V/Y/G/M','W/N/R/B/V/Y/G/M','B/R','N/R','V/Y','V','N/R/V','N/V/Y','R/B/O','W/B/V','W/V/Y','W/N/R/B','W/N/R/O','W/N/R/G','W/N/V/Y','W/N/Y/M','N/R/B/Y','N/B/V/Y','R/V/Y/O','W/B/V/M','W/B/V/O','N/R/B/Y/M','N/R/V/O/M','W/N/R/Y/G','N/R/B/V/Y','W/R/B/V/Y/P','W/N/R/B/Y/G','W/N/R/B/V/O/M','W/N/R/B/V/Y/M','W/N/B/V/Y/G/M','W/N/B/V/V/Y/P']

from itertools import \* #impot itertools

triplets\_pairs3 = [x for y in flower\_orders for x in combinations(''.join(sorted(list(set(y.replace("/",""))))), 3)] # create triplets

#print(triplets\_pairs3)

triplets\_pairs3\_counts = Counter(triplets\_pairs3)

print("Rank the triplets of colors.")

print(triplets\_pairs3\_counts)

#6. Make dictionary color for keys and values are what other colors it is ordered with.

#1st Dct:

flower\_orders = ['W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','B/Y','B/Y','B/Y','B/Y','B/Y','R/B/Y','R/B/Y','R/B/Y','R/B/Y','R/B/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/G','W/G','W/G','W/G','R/Y','R/Y','R/Y','R/Y','N/R/V/Y','N/R/V/Y','N/R/V/Y','N/R/V/Y','W/R/B/V','W/R/B/V','W/R/B/V','W/R/B/V','W/N/R/V/Y','W/N/R/V/Y','W/N/R/V/Y','W/N/R/V/Y','N/R/Y','N/R/Y','N/R/Y','W/V/O','W/V/O','W/V/O','W/N/R/Y','W/N/R/Y','W/N/R/Y','R/B/V/Y','R/B/V/Y','R/B/V/Y','W/R/V/Y','W/R/V/Y','W/R/V/Y','W/R/B/V/Y','W/R/B/V/Y','W/R/B/V/Y','W/N/R/B/Y','W/N/R/B/Y','W/N/R/B/Y','R/G','R/G','B/V/Y','B/V/Y','N/B/Y','N/B/Y','W/B/Y','W/B/Y','W/N/B','W/N/B','W/N/R','W/N/R','W/N/B/Y','W/N/B/Y','W/B/V/Y','W/B/V/Y','W/N/R/B/V/Y/G/M','W/N/R/B/V/Y/G/M','B/R','N/R','V/Y','V','N/R/V','N/V/Y','R/B/O','W/B/V','W/V/Y','W/N/R/B','W/N/R/O','W/N/R/G','W/N/V/Y','W/N/Y/M','N/R/B/Y','N/B/V/Y','R/V/Y/O','W/B/V/M','W/B/V/O','N/R/B/Y/M','N/R/V/O/M','W/N/R/Y/G','N/R/B/V/Y','W/R/B/V/Y/P','W/N/R/B/Y/G','W/N/R/B/V/O/M','W/N/R/B/V/Y/M','W/N/B/V/Y/G/M','W/N/B/V/V/Y/P']

# build two pair tupel

tuple\_pair2 = [x for y in flower\_orders for x in permutations(''.join(sorted(list(set(y.replace("/",""))))), 2)]

#print(tuple\_pair2)

#Create Counter to get unique pairs

tuple\_pair2\_c = Counter(tuple\_pair2)

#print(tuple\_pair2\_c)

#Create dictionary from keys of the counter

tuple\_pair\_dct = {}

for k,v in list(tuple\_pair2\_c.keys()):

    tuple\_pair\_dct.setdefault(k, []).append(v)

print("\n Dictionary Colors:")

print(tuple\_pair\_dct)

#2nd Dct:

flower\_orders = ['W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','B/Y','B/Y','B/Y','B/Y','B/Y','R/B/Y','R/B/Y','R/B/Y','R/B/Y','R/B/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/G','W/G','W/G','W/G','R/Y','R/Y','R/Y','R/Y','N/R/V/Y','N/R/V/Y','N/R/V/Y','N/R/V/Y','W/R/B/V','W/R/B/V','W/R/B/V','W/R/B/V','W/N/R/V/Y','W/N/R/V/Y','W/N/R/V/Y','W/N/R/V/Y','N/R/Y','N/R/Y','N/R/Y','W/V/O','W/V/O','W/V/O','W/N/R/Y','W/N/R/Y','W/N/R/Y','R/B/V/Y','R/B/V/Y','R/B/V/Y','W/R/V/Y','W/R/V/Y','W/R/V/Y','W/R/B/V/Y','W/R/B/V/Y','W/R/B/V/Y','W/N/R/B/Y','W/N/R/B/Y','W/N/R/B/Y','R/G','R/G','B/V/Y','B/V/Y','N/B/Y','N/B/Y','W/B/Y','W/B/Y','W/N/B','W/N/B','W/N/R','W/N/R','W/N/B/Y','W/N/B/Y','W/B/V/Y','W/B/V/Y','W/N/R/B/V/Y/G/M','W/N/R/B/V/Y/G/M','B/R','N/R','V/Y','V','N/R/V','N/V/Y','R/B/O','W/B/V','W/V/Y','W/N/R/B','W/N/R/O','W/N/R/G','W/N/V/Y','W/N/Y/M','N/R/B/Y','N/B/V/Y','R/V/Y/O','W/B/V/M','W/B/V/O','N/R/B/Y/M','N/R/V/O/M','W/N/R/Y/G','N/R/B/V/Y','W/R/B/V/Y/P','W/N/R/B/Y/G','W/N/R/B/V/O/M','W/N/R/B/V/Y/M','W/N/B/V/Y/G/M','W/N/B/V/V/Y/P']

lst\_fo= []

for x in flower\_orders:

    foo= x.split("/")

    lst\_fo.append(foo)

print(lst\_fo)

res = dict()

for sub in lst\_fo:

    dct = {sub[0]: sub[1:] for sub in lst\_fo}

# print result

print(dct)

#7. Make a graph showing the probability of having an edge between two colors based on how often they co-occur.  (a numpy square matrix)

from itertools import \* #impot itertools

from collections import Counter

import numpy as np

from tabulate import tabulate #to print arrays

flower\_orders = ['W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R/B','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','W/R','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','R/V/Y','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/N/R/V','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','W/R/B/Y','B/Y','B/Y','B/Y','B/Y','B/Y','R/B/Y','R/B/Y','R/B/Y','R/B/Y','R/B/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/N/R/B/V/Y','W/G','W/G','W/G','W/G','R/Y','R/Y','R/Y','R/Y','N/R/V/Y','N/R/V/Y','N/R/V/Y','N/R/V/Y','W/R/B/V','W/R/B/V','W/R/B/V','W/R/B/V','W/N/R/V/Y','W/N/R/V/Y','W/N/R/V/Y','W/N/R/V/Y','N/R/Y','N/R/Y','N/R/Y','W/V/O','W/V/O','W/V/O','W/N/R/Y','W/N/R/Y','W/N/R/Y','R/B/V/Y','R/B/V/Y','R/B/V/Y','W/R/V/Y','W/R/V/Y','W/R/V/Y','W/R/B/V/Y','W/R/B/V/Y','W/R/B/V/Y','W/N/R/B/Y','W/N/R/B/Y','W/N/R/B/Y','R/G','R/G','B/V/Y','B/V/Y','N/B/Y','N/B/Y','W/B/Y','W/B/Y','W/N/B','W/N/B','W/N/R','W/N/R','W/N/B/Y','W/N/B/Y','W/B/V/Y','W/B/V/Y','W/N/R/B/V/Y/G/M','W/N/R/B/V/Y/G/M','B/R','N/R','V/Y','V','N/R/V','N/V/Y','R/B/O','W/B/V','W/V/Y','W/N/R/B','W/N/R/O','W/N/R/G','W/N/V/Y','W/N/Y/M','N/R/B/Y','N/B/V/Y','R/V/Y/O','W/B/V/M','W/B/V/O','N/R/B/Y/M','N/R/V/O/M','W/N/R/Y/G','N/R/B/V/Y','W/R/B/V/Y/P','W/N/R/B/Y/G','W/N/R/B/V/O/M','W/N/R/B/V/Y/M','W/N/B/V/Y/G/M','W/N/B/V/V/Y/P']

# build two pair tupel

tuple\_pair2 = [x for y in flower\_orders for x in permutations(''.join(sorted(list(set(y.replace("/",""))))), 2)]

print(tuple\_pair2)

#Create Counter to get unique pairs

tuple\_pair2\_c = Counter(tuple\_pair2)

print(tuple\_pair2\_c)

#define function for numpy array; creates n\*n matrix and populates it from the given dictionary

def tuple\_dict\_to\_square\_matrix(d):

    d = dict(d)

    #get distinct first values from tuples and create index

    keys = sorted(set([y for x in d.keys() for y in x]))

    #we have keys list; initialize a square matrix

    out\_matrix = np.zeros((len(keys), len(keys)))

    #now read dict keys as matrix indices and values as values

    for k, v in d.items():

        out\_matrix[keys.index(k[0])][keys.index(k[1])] = np.round(v, 5)

    #return the array as well as the keys index to help with print

    return [out\_matrix, keys]

#define function to display array using tabulate; takes floatformat as optional parameter

#takes output from tuple\_dict\_to\_square\_matrix as the input

def format\_print\_array(a, ff=""):

    table = tabulate(a[0], a[1], showindex=a[1], tablefmt="fancy\_grid", numalign="right", floatfmt=ff)

    print(table)

tuple\_pair\_perms\_prob = {k: v / (sum(tuple\_pair2\_c.values())/2) for k, v in tuple\_pair2\_c.items()}

#print(tuple\_pair\_perms\_prob)

dict\_to\_matrix\_out = tuple\_dict\_to\_square\_matrix(dict(tuple\_pair\_perms\_prob))

format\_print\_array(dict\_to\_matrix\_out)

#8. Make 10 business questions related to the questions we asked above.

    # i) Determine the color combinations ordered?

    # ii) Which combination is sold most?

    # iii) What consumption were dolded "W" color?

    # iv) What is consumption of each color?

    # v) How consumption of comparedwith each other?

    # vi) What combinations are most probable?

    # vii) List bottem 5 color consumed (may be look for 80/20 kind of split)?

    # viii) List five most popular pairs of color combination?

    # iX) List five top most popular triplet?

    # x) What are possible combination with each color?

# dead\_men\_tell\_taies

# 1. Join everything

dead\_men\_tell\_taies = ['Four score and seven years ago our fathers brought forth on this','continent a new nation, conceived in liberty and dedicated to the','proposition that all men are created equal. Now we are engaged in','a great civil war, testing whether that nation or any nation so','conceived and so dedicated can long endure. We are met on a great','battlefield of that war. We have come to dedicate a portion of','that field as a final resting-place for those who here gave their','lives that that nation might live. It is altogether fitting and','proper that we should do this. But in a larger sense, we cannot','dedicate, we cannot consecrate, we cannot hallow this ground.','The brave men, living and dead who struggled here have consecrated','it far above our poor power to add or detract. The world will','little note nor long remember what we say here, but it can never','forget what they did here. It is for us the living rather to be','dedicated here to the unfinished work which they who fought here','have thus far so nobly advanced. It is rather for us to be here','dedicated to the great task remaining before us--that from these','honored dead we take increased devotion to that cause for which','they gave the last full measure of devotion--that we here highly','resolve that these dead shall not have died in vain, that this','nation under God shall have a new birth of freedom, and that','government of the people, by the people, for the people shall','not perish from the earth.']

story\_join = "".join(dead\_men\_tell\_taies) #create string

print(story\_join)

foo\_len=len(story\_join)

print(foo\_len)

# 2. Remove spaces

dead\_men\_tell\_taies1 = ['Four score and seven years ago our fathers brought forth on this','continent a new nation, conceived in liberty and dedicated to the','proposition that all men are created equal. Now we are engaged in','a great civil war, testing whether that nation or any nation so','conceived and so dedicated can long endure. We are met on a great','battlefield of that war. We have come to dedicate a portion of','that field as a final resting-place for those who here gave their','lives that that nation might live. It is altogether fitting and','proper that we should do this. But in a larger sense, we cannot','dedicate, we cannot consecrate, we cannot hallow this ground.','The brave men, living and dead who struggled here have consecrated','it far above our poor power to add or detract. The world will','little note nor long remember what we say here, but it can never','forget what they did here. It is for us the living rather to be','dedicated here to the unfinished work which they who fought here','have thus far so nobly advanced. It is rather for us to be here','dedicated to the great task remaining before us--that from these','honored dead we take increased devotion to that cause for which','they gave the last full measure of devotion--that we here highly','resolve that these dead shall not have died in vain, that this','nation under God shall have a new birth of freedom, and that','government of the people, by the people, for the people shall','not perish from the earth.']

story\_join = "".join(dead\_men\_tell\_taies)

story\_space = story\_join.replace(" ", "") # replace space

print(story\_space)

print(len(story\_space))

# 3. Occurrence probabilities for letters

from itertools import \* #impot itertools

from collections import Counter

import numpy as np

from tabulate import tabulate

dead\_men\_tell\_taies = ['Four score and seven years ago our fathers brought forth on this','continent a new nation, conceived in liberty and dedicated to the','proposition that all men are created equal. Now we are engaged in','a great civil war, testing whether that nation or any nation so','conceived and so dedicated can long endure. We are met on a great','battlefield of that war. We have come to dedicate a portion of','that field as a final resting-place for those who here gave their','lives that that nation might live. It is altogether fitting and','proper that we should do this. But in a larger sense, we cannot','dedicate, we cannot consecrate, we cannot hallow this ground.','The brave men, living and dead who struggled here have consecrated','it far above our poor power to add or detract. The world will','little note nor long remember what we say here, but it can never','forget what they did here. It is for us the living rather to be','dedicated here to the unfinished work which they who fought here','have thus far so nobly advanced. It is rather for us to be here','dedicated to the great task remaining before us--that from these','honored dead we take increased devotion to that cause for which','they gave the last full measure of devotion--that we here highly','resolve that these dead shall not have died in vain, that this','nation under God shall have a new birth of freedom, and that','government of the people, by the people, for the people shall','not perish from the earth.']

story\_join = "".join(dead\_men\_tell\_taies)

story\_space = story\_join.replace(" ", "") # remove space

story\_let = story\_space.replace(",", "") # remove ","

story\_let = story\_let.replace(".", "") # remove "."

story\_let = story\_let.replace("-", "") # remove "-"

#print(story\_let)

#print(len(story\_let))

story\_let\_l = story\_let.lower() #convert all letters to lower case

letter\_count = dict(Counter(story\_let\_l)) # create a directory of letters and respective count

#print(letter\_count)

letter\_prob = {k: np.round(v / len(story\_let\_l), 5) for k, v in letter\_count.items()} # Calculate probility for each letter

print(letter\_prob) # calculate probility of each letter

letter\_prob\_s = dict(sorted(letter\_prob.items(), key=lambda item: item[0])) # sort the letter

print('Occurrence probabilities for letters:',letter\_prob\_s) # sort the probility

# 4. Tell me transition probabilities for every letter pairs

from collections import Counter

story\_list =[]

story\_let\_l = story\_let.lower() #convert all letters to lower case

story\_list[:0] = story\_let\_l

#story\_list = list[story\_let\_l]

letter\_pairs = [(story\_list[i], story\_list[i+1]) for i in range(0, len(story\_list) - 1)]

print(letter\_pairs)

#Create Counter to get unique pairs

letter\_pair\_cnt = dict(Counter(letter\_pairs)) # create a directory of letter pairs and respective count

print(letter\_pair\_cnt)

letter\_pair\_prob = {k: np.round(v / (sum(letter\_pair\_cnt.values())/2),5) for k, v in letter\_pair\_cnt.items()}

print(letter\_pair\_prob)

# 5. Make a 26x26 graph of 4. in numpy

letter\_pair\_prob = {k: np.round(v / (sum(letter\_pair\_cnt.values())/2),5) for k, v in letter\_pair\_cnt.items()}

print(letter\_pair\_prob) # calculate probility of each pair of letter

#define function for numpy array; creates n\*n matrix and populates it from the given dictionary

def tuple\_dict\_to\_square\_matrix(d):

    d = dict(d)

    #get distinct first values from tuples and create index

    keys = sorted(set([y for x in d.keys() for y in x]))

    #we have keys list; initialize a square matrix

    out\_matrix = np.zeros((len(keys), len(keys)))

    #now read dict keys as matrix indices and values as values

    for k, v in d.items():

        out\_matrix[keys.index(k[0])][keys.index(k[1])] = np.round(v, 5)

    #return the array as well as the keys index to help with print

    return [out\_matrix, keys]

#define function to display array using tabulate; takes floatformat as optional parameter

#takes output from tuple\_dict\_to\_square\_matrix as the input

def format\_print\_array(a, ff=""):

    table = tabulate(a[0], a[1], showindex=a[1], tablefmt="fancy\_grid", numalign="right", floatfmt=ff)

    print(table)

dict\_to\_matrix\_out = tuple\_dict\_to\_square\_matrix(dict(letter\_pair\_prob))

format\_print\_array(dict\_to\_matrix\_out) # create the matrix

# 6. plot graph of transition probabilities from letter to letter

# I am having challnge with metplotlib. but prints in jupiter note

# discussed on form looks like config issue but could not fix it.

import matplotlib.pyplot as plt

import numpy as np

letter\_pair\_prob = {k: np.round(v / (sum(letter\_pair\_cnt.values())/2),5) for k, v in letter\_pair\_cnt.items()}

print(letter\_pair\_prob)

plt.bar(letter\_pair\_prob.keys(), letter\_pair\_prob.values())

plt.show() # graph is enclosed on word doc for your referance

#Unrelated:

#7. Flatten a nested list

#Cool intro python resources:

#https://thomas-cokelaer.info/tutorials/python/index.html

#Create Counter to get unique pairs

plt.imshow(dict\_to\_matrix\_out[0], cmap='Blues')

plt.xticks(np.arange(len(dict\_to\_matrix\_out[1])), dict\_to\_matrix\_out[1])

plt.yticks(np.arange(len(dict\_to\_matrix\_out[1])), dict\_to\_matrix\_out[1])

a\_nested\_list = [[1, 2, 3], [4, 5, 6]]

flat\_list = [item for sublist in a\_nested\_list for item in sublist]

print(flat\_list)

from itertools import \* #impot itertools

from collections import Counter

import itertools

from psutil.\_compat import b

import itertools

import itertools

from tqdm.contrib import itertools

lst = ['a', 'b', 'c', 'd']

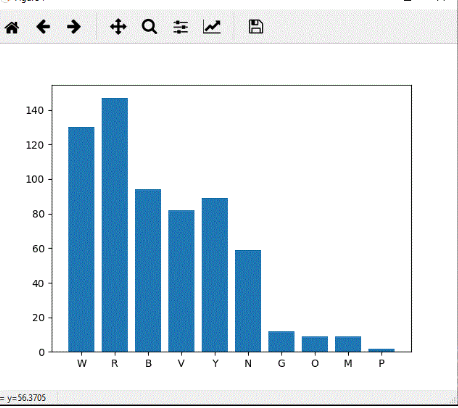
print("The original list : ", lst)

res = list(combinations(lst, 2))

print("All possible pairs : ", res)

flower\_orders

Q3 . Make histogram of colors



dead\_mem\_tell\_taies

Q5 Make a 26x26 graph of 4. in numpy

