import random as rn

import numpy as np

"""

Problem 1

"""

def unique\_words(xstring):

    """Problem 1. Find the unique words in a string

        Args: String of text

        Returns: list of unique words in the input text

    """

    # The text might contain both upper case and lower case, so doing simple cleaning to ensure

    # input contain same case letters and words only.

    input\_text = xstring.lower().split()

    word\_list = [word.strip('.,!;()[]') for word in input\_text] # Removing punctuation.

    # passing word\_list into set, because set will hold only unique values.

    unique\_word\_list = set(word\_list)

    # Converting set into list and return it.

    return list(unique\_word\_list)

def get\_transition\_matrix(xtr):

    """Problem 1. Generate the transition matrix

    Args: String of text

    Returns: list of lists which contain transition matrix

    """

    # Getting unique value for the matrix

    unique\_words\_list = unique\_words(xtr)

    # convert unique value list into dict for lookup

    unique\_words\_dict = {k: v for v, k in enumerate(unique\_words\_list)}

    # Word list

    input\_text = xtr.lower().split()

    word\_list = [word.strip('.,!;()[]') for word in input\_text]

    # Initialize matrix

    result\_matrix = [ [ 0 for i in range(len(unique\_words\_list)) ] for j in range(len(unique\_words\_list)) ]

    for i in range(len(word\_list)-1):

        row\_value = unique\_words\_dict[word\_list[i]] # Get row index of the word

        col\_value = unique\_words\_dict[word\_list[i+1]] # Get column index of next word

        result\_matrix[row\_value][col\_value] += 1 # Update result matrix

    return result\_matrix

""" Problem 2 """

def running\_average(xlist,per):

    """

        Compute thr running average

        Args: xlist: contain list of randomly generated number

              per : Period which need to be calculeted

        Returns: list contains moving average

    """

    # np.convolve returns discret, linear convolution of two 1-d sequence

    '''

        Input list convolving with a sequence of np.ones. the np.ones length is equal to the period we want.

        so, the input sequence took x list and period and creates sequence of ones lenght of period.

        we choose mode is valid. so that the convolution product will give only for the values the sequence overlapping.

    '''

    result = np.convolve(xlist, np.ones(per), 'valid') / per

    # Rounding result to two digit and return as list

    return list(np.around(result, decimals = 2))

if \_\_name\_\_ == "\_\_main\_\_":

    text = '''The cat is in the house. The dog is outside playing with the kids.

              Both the dog and the cat need a bath. The kids need to come in and eat dinner.'''

    uniwords = unique\_words(text)

    print(uniwords)

    print(f"There are {len(uniwords)} unique words in the text.")

    print("The transition matrix is below:")

    print(get\_transition\_matrix(text))

#Generate random data sequence

    data = [rn.randint(1, 100) for i in range(10)]

    print(data)

    period = 3 # time period for running avg (3 day average)

    run\_avg = running\_average(data,period)

    print(f"The {period}-day running average is: {run\_avg}")