**Questions :** rite a function so that the columns of the output matrix are powers of the input

vector.

The order of the powers is determined by the increasing boolean argument. Specifically, when

increasing is False, the i-th output column is the input vector raised element-wise to the power

of N - i - 1.

HINT: Such a matrix with a geometric progression in each row is named for Alexandre-

Theophile Vandermonde.

Problem Statement 2:

Given a sequence of n values x1, x2, ..., xn and a window size k>0, the k-th moving average of

the given sequence is defined as follows:

The moving average sequence has n-k+1 elements as shown below.

The moving averages with k=4 of a ten-value sequence (n=10) is shown below

i 1 2 3 4 5 6 7 8 9 10

===== == == == == == == == == == ==

Input 10 20 30 40 50 60 70 80 90 100

y1 25 = (10+20+30+40)/4

y2 35 = (20+30+40+50)/4

y3 45 = (30+40+50+60)/4

y4 55 = (40+50+60+70)/4

y5 65 = (50+60+70+80)/4

y6 75 = (60+70+80+90)/4

y7 85 = (70+80+90+100)/4

Thus, the moving average sequence has n-k+1=10-4+1=7 values.

Question: Write a function to find moving average in an array over a window:

Test it over [3, 5, 7, 2, 8, 10, 11, 65, 72, 81, 99, 100, 150] and window of 3.

**Answers:**

**Problem Statement 1: Vandermonde Matrix**

Here's the function to create the Vandermonde matrix:

**Python Code**

import numpy as np

def vandermonde\_matrix(vector, N=None, increasing=False):

if N is None:

N = len(vector)

if increasing:

matrix = np.array([vector\*\*i for i in range(N)]).T

else:

matrix = np.array([vector\*\*(N-1-i) for i in range(N)]).T

return matrix

# Example usage

vector = np.array([1, 2, 3, 4, 5])

print("Vandermonde matrix (increasing):")

print(vandermonde\_matrix(vector, increasing=True))

print("\nVandermonde matrix (decreasing):")

print(vandermonde\_matrix(vector, increasing=False))

**Problem Statement 2: Moving Average**

To compute the moving average of an array over a given window, we can create a function that uses a sliding window approach. Here's the function:

**Python Code**

def moving\_average(sequence, k):

n = len(sequence)

if k > n:

return []

moving\_averages = []

for i in range(n - k + 1):

window = sequence[i:i + k]

window\_average = sum(window) / k

moving\_averages.append(window\_average)

return moving\_averages

# Example usage

sequence = [3, 5, 7, 2, 8, 10, 11, 65, 72, 81, 99, 100, 150]

window\_size = 3

print("Moving averages with window size 3:")

print(moving\_average(sequence, window\_size))