Write 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 AlexandreTheophile Vandermonde.

In [18]:

```
import numpy as np
def gen_vandermonde_matrix_user_defined(inputvector, increasing=True):
    if(increasing):
        return np.array([i**j for i in inputvector for j in range(len(inputvecto
    return np.array([i**(len(inputvector)-j-1) for i in inputvector for j in ran



inputvector = [1, 2, 3, 4]
print('----------Increasing Order--------------')
print(gen_vandermonde_matrix_user_defined(inputvector, True))
print('----------Decreasing Order--------------')
print(gen_vandermonde_matrix_user_defined(inputvector, False))
```

```
----------Increasing Order--------------
[[ 1  1  1  1]
 [ 1  2  4  8]
 [ 1  3  9 27]
 [ 1  4 16 64]]
----------Decreasing Order--------------
[[ 1  1  1  1]
 [ 8  4  2  1]
 [27  9  3  1]
 [64 16  4  1]]
```

In [21]:

```
import numpy as np
def gen_vandermone_matrix(vector, increasing = True):
    return np.vander(a, increasing=increasing)

vector = np.array([1, 2, 3, 4])
print('----------Increasing Order--------------')
print(gen_vandermone_matrix(vector, True))
print('----------Decreasing Order--------------')
print(gen_vandermone_matrix(vector, False))
```

```
----------Increasing Order--------------
[[ 1  1  1  1]
 [ 1  2  4  8]
 [ 1  3  9 27]
 [ 1  4 16 64]]
----------Decreasing Order--------------
[[ 1  1  1  1]
 [ 8  4  2  1]
 [27  9  3  1]
 [64 16  4  1]]
```

In [26]:

```
np.vander(a)
```

Out[26]:

```
array([[ 1,  1,  1,  1],
       [ 8,  4,  2,  1],
       [27,  9,  3,  1],
       [64, 16,  4,  1]])
```

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.

In [7]:

```
import numpy as np
def moving_average_user_defined(data, window=3):
    l=[]
    for i in range(len(data) - window + 1):
        l.append(np.average(data[i : window + i]))
    print(l)

input_data =  [3, 5, 7, 2, 8, 10, 11, 65, 72, 81, 99, 100, 150]
window = 3
moving_average_user_defined(input_data, window)
```

```
[5.0, 4.666666666666667, 5.666666666666667, 6.666666666666667, 9.66666666666666
6, 28.666666666666668, 49.333333333333336, 72.66666666666667, 84.0, 93.33333333
333333, 116.33333333333333]
```

In [35]:

```
import numpy as np
def moving_average(data, window = 3):
    return np.convolve(data, np.ones(window)/window, mode='valid')

input_data =  [3, 5, 7, 2, 8, 10, 11, 65, 72, 81, 99, 100, 150]
window = 3
print(list(moving_average(input_data, window)))
```

```
[5.0, 4.666666666666666, 5.666666666666666, 6.666666666666666, 9.66666666666666
6, 28.666666666666664, 49.33333333333333, 72.66666666666666, 84.0, 93.333333333
33333, 116.33333333333333]
```

In [ ]: