Problem 1

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.

In [1]:
```
import numpy as np

def gen_vander_matrix(ipvector, n, increasing=False):

    if not increasing:
        op_matx = np.array([x**(n-1-i) for x in ipvector for i in range(n)]).reshape(ipvecto
r.size,n)
    elif increasing:
        op_matx = np.array([x**i for x in ipvector for i in range(n)]).reshape(ipvector.size
,n)

    return op_matx

print("------------OUTPUT-------------\n")

inputvector = np.array([1,2,3,4,5])
no_col_opmat = 5
op_matx_dec_order = gen_vander_matrix(inputvector,no_col_opmat,False)
op_matx_inc_order = gen_vander_matrix(inputvector,no_col_opmat,True)

print("The input array is:",inputvector,"\n")
print("Number of columns in output matrix should be:",no_col_opmat,"\n")
print("Vander matrix of the input array in decreasing order of powers:\n\n",op_matx_dec_orde
r,"\n")
print("Vander matrix of the input array in increasing order of powers:\n\n",op_matx_inc_orde
r)
```

```
------------OUTPUT-------------

The input array is: [1 2 3 4 5]

Number of columns in output matrix should be: 5

Vander matrix of the input array in decreasing order of powers:

 [[  1   1   1   1   1]
 [ 16   8   4   2   1]
 [ 81  27   9   3   1]
 [256  64  16   4   1]
 [625 125  25   5   1]]

Vander matrix of the input array in increasing order of powers:

 [[  1   1   1   1   1]
 [  1   2   4   8  16]
 [  1   3   9  27  81]
 [  1   4  16  64 256]
 [  1   5  25 125 625]]
```

Problem 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 [3]:
```
import numpy as np

def moving_average(a, n=3) :
    x = np.cumsum(a, dtype=float)
    x[n:] = x[n:] - x[:-n]
    return x[n - 1:] / n

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

```
[  5.          4.66666667   5.66666667   6.66666667   9.66666667
  28.66666667  49.33333333  72.66666667  84.          93.33333333
 116.33333333]
```