Name: Ruturajsinh Udaysinh Vaghela

Reg No: 24-27-07

Programme: MTech Data Science

Assignment Number: 2

Q1

In [1]:
```
# a)

import numpy as np
var1 = np.arange(0, 31)
print(var1)
var1.shape
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30]
```

Out[1]:
```
(31,)
```

In [2]:
```
# b)
# Since var1 has 31 elements and 31 can't be converted into 2D. So removing the firs

var1 = np.delete(var1, 0)

var2 = var1.reshape(5, 6)
print(var2)
var2.shape
```

```
[[ 1  2  3  4  5  6]
 [ 7  8  9 10 11 12]
 [13 14 15 16 17 18]
 [19 20 21 22 23 24]
 [25 26 27 28 29 30]]
```

Out[2]:
```
(5, 6)
```

In [3]:
```
# c)

var3 = var1.reshape(2, 5, 3)
print(var3)
var3.shape
```

```
[[[ 1  2  3]
  [ 4  5  6]
  [ 7  8  9]
  [10 11 12]
  [13 14 15]]

 [[16 17 18]
  [19 20 21]
  [22 23 24]
  [25 26 27]
  [28 29 30]]]
```

Out[3]:
```
(2, 5, 3)
```

In [4]:
```
# d)

var2[1][0] = -1
print(f"Var 2: \n{var2} \n\n Var 1: \n{var1} \n\n Var 3: \n{var3}")

print('''
Since we used reshape on var1 to get var2 and var3 it creates a copy or the numpy ar
Hence if the elements of 1 array is modified... The other 2 arrays elements will als
Therefor when we changed var2 elements as -1 it also reflected in var1 and var3
''')
```

```
Var 2:
[[ 1  2  3  4  5  6]
 [-1  8  9 10 11 12]
 [13 14 15 16 17 18]
 [19 20 21 22 23 24]
 [25 26 27 28 29 30]]

 Var 1:
[ 1  2  3  4  5  6 -1  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
 25 26 27 28 29 30]

 Var 3:
[[[ 1  2  3]
  [ 4  5  6]
  [-1  8  9]
  [10 11 12]
  [13 14 15]]

 [[16 17 18]
  [19 20 21]
  [22 23 24]
  [25 26 27]
  [28 29 30]]]

Since we used reshape on var1 to get var2 and var3 it creates a copy or the numpy arr
ay
Hence if the elements of 1 array is modified... The other 2 arrays elements will also
be modified
Therefor when we changed var2 elements as -1 it also reflected in var1 and var3
```

In [6]:
```
# e)

# i)
res1 = np.sum(var3, axis = 1)
print("Sum over 2nd Dimension: \n", res1)

# ii)
res2 = np.sum(var3, axis = 2)
print("\nSum over 3rd Dimension: \n", res2)

# iii)
res3 = np.sum(var3, axis = (0, 2))
print("\nSum over 1st and 3rd Dimensions: \n", res3)
```

```
Sum over 2nd Dimension:
 [[ 27  40  45]
 [110 115 120]]
```

```
Sum over 3rd Dimension:
 [[ 6 15 16 33 42]
 [51 60 69 78 87]]

Sum over 1st and 3rd Dimensions:
 [ 57  75  85 111 129]
```

In [10]:

```python
# f)

print("Var2: \n", var2)
# i)
res1 = var2[1]
print("\n2nd Row of var2: \n", res1)

# ii)
res2 = var2[:, -1]
print("\nLast column of var2: \n", res2)

# iii)
res3 = var2[:2, -2:]
print("\nTop Right 2x2 Submatrix of var2: \n", res3)
```

```
Var2:
 [[ 1  2  3  4  5  6]
 [-1  8  9 10 11 12]
 [13 14 15 16 17 18]
 [19 20 21 22 23 24]
 [25 26 27 28 29 30]]

2nd Row of var2:
 [-1  8  9 10 11 12]

Last column of var2:
 [ 6 12 18 24 30]

Top Right 2x2 Submatrix of var2:
 [[ 5  6]
 [11 12]]
```

Q2

In [12]:

```python
# a)

vector = np.arange(10) + 1
print(vector)
```

```
[ 1  2  3  4  5  6  7  8  9 10]
```

In [13]:

```python
# b)

A = vector.reshape(10, 1) + vector
print(A)
```

```
[[ 2  3  4  5  6  7  8  9 10 11]
 [ 3  4  5  6  7  8  9 10 11 12]
 [ 4  5  6  7  8  9 10 11 12 13]
 [ 5  6  7  8  9 10 11 12 13 14]
 [ 6  7  8  9 10 11 12 13 14 15]
 [ 7  8  9 10 11 12 13 14 15 16]
 [ 8  9 10 11 12 13 14 15 16 17]
 [ 9 10 11 12 13 14 15 16 17 18]
 [10 11 12 13 14 15 16 17 18 19]
 [11 12 13 14 15 16 17 18 19 20]]
```

In [17]:
```
# c)

import numpy.random as npr
data = np.exp(npr.randn(50, 5))
```

In [18]:
```
# d)

print(data)
```

```
[[ 1.03540376  0.32884487  0.2041805   0.16389041  3.17432344]
 [ 0.65233132  0.17048763  3.68866082  0.18364127  2.39669719]
 [ 1.9434254   1.88433181  0.62566655  0.30496564  0.37400965]
 [ 0.31637298  0.29774947  1.13593444  1.3723215   7.30320315]
 [ 0.15974331  0.44263702  1.4181086   0.1656185   0.36385167]
 [ 1.59628369  3.51131038  1.33953283  0.77714533  0.88223619]
 [ 2.04747583  0.44492639  0.30468473  0.68722602  0.88669336]
 [ 3.47633574  0.92106767  0.64819553  0.20584494  0.33625396]
 [ 2.12048548  1.43327094  0.61994934  1.24343938  0.40237315]
 [ 0.32395723  4.11808269  1.81554419  0.42540764  6.01456447]
 [ 2.06500103  2.42584004  2.78394267  0.27204254  0.93305398]
 [ 2.43545405  2.54995427  0.43742659  0.71210955  1.54000187]
 [ 3.45851146  0.68550593  2.57525154  1.32884076  6.11538769]
 [ 1.39116844  2.33679793  0.59740371  2.19370203  1.00664067]
 [ 0.34809204  0.55288192  0.65633958  1.67873178  0.77780403]
 [ 0.16653839  0.06216871  2.90310372  0.80917034  2.31282778]
 [ 1.96083323  1.03694797  1.15133131  0.43801161  2.55432381]
 [ 1.06289088  0.15019003  0.30091183 11.82422007  4.15148286]
 [ 0.90875601  2.42227352  0.06676948  7.29777999  2.25848654]
 [ 2.81757182  0.74633122  0.81912607  4.14714423  0.34810124]
 [ 1.0803938   0.31597945  1.06202388  1.09561413  1.11338983]
 [ 2.34883685  4.27242299  0.18938682  0.69451032  2.37774493]
 [ 1.5854319   2.33172615  0.46884362  0.37278844  1.9409641 ]
 [ 1.59414834  0.53155462  0.35380838  0.40749922  0.69277184]
 [ 0.40729139  2.06151069  0.95550358  0.19268741  4.08065151]
 [ 0.38197093  1.21553045  1.20587396  1.78572015  1.77807371]
 [ 0.60757039  1.6631604   0.40151845  0.96825145  0.57416618]
 [ 5.51289961  0.53794806  2.53057997  4.85513193  1.62011431]
 [ 4.71039449  0.55179957  1.54993817  0.56815461  1.78101476]
 [ 6.99414687 15.56219753  0.24066803  0.25939194  0.61521684]
 [ 1.58911142  0.4231363   0.31331522  2.77830788  5.64477673]
 [ 1.07200218  0.53130138  0.40669175  0.42597589  0.48330686]
 [ 3.18258393  7.39574693  1.51134733  0.40291764  4.03870159]
 [ 6.95202993  2.85034287  2.60149095  2.57643669  1.07287481]
 [ 0.1796863   1.60960446  0.36245657  0.1097793   1.83865691]
 [ 1.90178314  6.45416597  2.36822904  0.76385274  1.27050148]
 [ 1.35199111  1.99484792  8.28585708  0.25622437  1.60730882]
 [ 2.81768221  2.90715103  1.31333721  2.70620475  0.33868731]
 [ 0.54164594  1.9992426   0.74388008  0.99725938  2.36779158]
 [ 1.39385974  1.81466476  5.85367832  0.35733437  0.58046304]
 [ 0.257645    0.11976272  1.565816    0.71318411  2.98606164]
 [ 0.28026363  2.7991754   0.66023325  1.25813744  3.04776514]
 [ 1.47337937  2.01172756  0.89075888  1.40485025  1.31662864]
 [ 0.23879954  3.13109628  0.27305486  0.70933289  0.58237389]
 [ 1.69680722  0.73537066  0.71086759  0.76438023  0.39584341]
 [ 0.38554015  0.22444764  4.71102339  1.4305002   1.80013246]
 [ 0.54980885  0.48420732  0.85478117  0.9655112   3.20680018]
 [ 0.5100933   2.21649288  0.10731429  0.68344947  0.94670864]
 [ 0.32018325  1.065143    0.23089757  0.65115669  1.51458713]
 [ 1.46340815  0.24154595  0.82548266  4.83335876  1.20299169]]
```

In [22]:
```
# e)

mean = np.mean(data, axis = 0)
s_dev = np.std(data, axis = 0)
```

```
print(f"Mean: {mean}\nStandard Deviation: {s_dev}")
```

```
Mean: [1.67336042 1.93149208 1.35281444 1.44438315 1.93898773]
Standard Deviation: [1.59193028 2.47871806 1.54137272 2.03958503 1.64499246]
```

In [23]:
```
# f)

norm = (data - mean)/s_dev

mean = np.mean(norm, axis = 0)
s_dev = np.std(norm, axis = 0)

print(f"Mean: {mean}\nStandard Deviation: {s_dev}")
```

```
Mean: [-1.75415238e-16 -1.93178806e-16 -5.21804822e-17 -1.64313008e-16
 -2.73114864e-16]
Standard Deviation: [1. 1. 1. 1. 1.]
```

Q3

In [54]:
```
# a)

def vandermonde(N):
    vec = np.arange(N) + 1
    return vec[:, np.newaxis] ** (vec - 1)

res = vandermonde(12)
print(res)
```

```
[[          1           1           1           1           1           1
            1           1           1           1           1           1]
 [          1           2           4           8          16          32
           64         128         256         512        1024        2048]
 [          1           3           9          27          81         243
          729        2187        6561       19683       59049      177147]
 [          1           4          16          64         256        1024
         4096       16384       65536      262144     1048576     4194304]
 [          1           5          25         125         625        3125
        15625       78125      390625     1953125     9765625    48828125]
 [          1           6          36         216        1296        7776
        46656      279936     1679616    10077696    60466176   362797056]
 [          1           7          49         343        2401       16807
       117649      823543     5764801    40353607   282475249  1977326743]
 [          1           8          64         512        4096       32768
       262144     2097152    16777216   134217728  1073741824           0]
 [          1           9          81         729        6561       59049
       531441     4782969    43046721   387420489  -808182895  1316288537]
 [          1          10         100        1000       10000      100000
      1000000    10000000   100000000  1000000000  1410065408  1215752192]
 [          1          11         121        1331       14641      161051
      1771561    19487171   214358881 -1937019605   167620825  1843829075]
 [          1          12         144        1728       20736      248832
      2985984    35831808   429981696   864813056  1787822080   -20971520]]
```

In [55]:
```
# b)

x = np.ones(12)
b = np.dot(res, x)
print(b)
```

```
[1.20000000e+01 4.09500000e+03 2.65720000e+05 5.59240500e+06
 6.10351560e+07 4.35356467e+08 2.30688120e+09 1.22713351e+09
 9.43953692e+08 3.73692871e+09 3.10225064e+08 3.10073456e+09]
```

In [61]:
```python
# c)

from numpy.linalg import inv
x2 = inv(res).dot(b)
print(x2)
```

```
[1.00537872 0.99484253 0.99961472 1.00018311 0.99998856 1.00000012
 1.         1.         1.         1.         1.         1.        ]
```

In [66]:
```python
# d)

x3 = np.linalg.solve(res, b)
print(x3)
```

```
[0.99999915 1.00000115 0.99999995 0.99999962 1.00000017 0.99999997
 1.         1.         1.         1.         1.         1.        ]
```

https://github.com/Ruturaj18/Ruturaj_Vaghela_27-07.git