Assn2_Tirthankar_Datta_24-52-28

October 10, 2024

[ ]:
```
#Name: Tirthankar Datta
#Registration Number: 24-52-28
#Programme: PhD
#Assignment Number: 2
```

[11]:
```
#Question 1

import numpy as np
var1=np.arange(0,31,1)
print(var1)
print("Shape :"+str(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]
Shape :(31,)
```

[13]:
```
var2=var1[:-1].reshape(3,10)
print(var2)
```

```
[[ 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]]
```

[15]:
```
var3=var1[:-1].reshape(3,5,2)
print(var3)
```

```
[[[ 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]]]
```

[17]:
```
var2[1,0]=-1
print(var2)
```

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

[19]:
```
print(var1)
print(var3)
```

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

 [[-1 11]
  [12 13]
  [14 15]
  [16 17]
  [18 19]]

 [[20 21]
  [22 23]
  [24 25]
  [26 27]
  [28 29]]]
```

[21]:
```
sum_over_second_dim=np.sum(var3,axis=1)
print(sum_over_second_dim)
```

```
[[ 20  25]
 [ 59  75]
 [120 125]]
```

[23]:
```
sum_over_third_dim=np.sum(var3,axis=2)
print(sum_over_third_dim)
```

```
[[ 1  5  9 13 17]
 [10 25 29 33 37]
 [41 45 49 53 57]]
```

[25]:
```
sum_over_first_second=np.sum(var3,axis=(0,2))
print(sum_over_first_second)
```

```
[ 52  75  87  99 111]
```

[27]:
```
print(var2[1:2])
print(var2[0:,-1])
print(var2[:2,-2:])
```

```
[[-1 11 12 13 14 15 16 17 18 19]]
[ 9 19 29]
[[ 8  9]
 [18 19]]
```

[29]:
```
#Question 2

arr=np.arange(0,10,1)
arr_b=arr+1
print(arr_b)
```

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

[31]:
```
arr2=arr+arr.reshape(10,1)
print(arr2)
```

```
[[ 0  1  2  3  4  5  6  7  8  9]
 [ 1  2  3  4  5  6  7  8  9 10]
 [ 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]]
```

[35]:
```
dataset=np.exp(np.random.randn(50,5))
print(dataset)
```

```
[[ 0.61105159  0.44300341  0.14898338  0.74302806  2.35541348]
 [ 0.70786584  9.06742371  1.42792013  0.47160835  0.31655374]
 [13.68768163  5.35762069  0.59214471  0.96822704  0.61917382]
 [ 1.56160384  0.63870212  3.06962323  0.79152161  1.55013182]
 [ 0.18242826  1.46992857  1.90473368  0.3963871   0.21271932]
 [ 0.42127946  0.32028772  3.44584785  0.68381577  0.6272904 ]
 [ 2.05803813  1.82559543 39.77221344  0.92237894  0.44234843]
 [ 2.29392588  3.32619634  0.80462363  0.51031767  1.91751238]
 [ 0.89567089  0.79977575  0.54207956  1.92060387  0.83351968]
 [ 0.17537369  1.87626186  0.92829949  0.94838375  4.19627839]
 [ 0.28552175  0.17959479  4.84267789  0.21810311  0.3965047 ]
```

```
[ 1.54061806  1.0569967   1.53679039  0.70233797  0.98493012]
[ 1.93759264  0.14594872  0.80564668  0.32654564  1.6253854 ]
[ 0.51527872  5.47221271  1.78095473  1.07919637  1.20077478]
[ 4.11159419  0.31389986  0.66852752  0.96269262  0.13609841]
[ 0.27287904  4.51410153  1.80000257  0.20862791  1.29426992]
[ 0.65367291  1.18851268  1.15797613  3.27495845  1.05284856]
[ 3.38412812  0.41072914  4.01289749  0.25911718  3.69550984]
[ 0.68931377  1.32250943  2.00398559  1.46813454  0.44866637]
[ 0.96581161  1.74320449  1.81585972  0.25543567  2.03780401]
[ 1.17554343  0.95212499  0.69209298  0.54130891  4.87795914]
[ 1.51816742  1.35654685  1.4675194   4.3127653   0.18755796]
[ 0.34474139  0.51850978  5.08129274  2.46970532  1.21885242]
[ 2.5254355   0.27256191  3.02786709  1.55561027  2.97158941]
[ 0.22941846  0.21698481  0.31025897  0.56249817  0.45985596]
[ 0.73418052  0.68611336  1.19869026  0.28799419  0.31889552]
[ 0.38577004  0.55729289  0.85185557  1.23165704  0.38263447]
[ 2.67089106  0.22305669  3.95599976  0.95631383  2.45573589]
[ 2.35808728  0.41438679  0.41593343  1.9960048   3.33438244]
[ 2.74501853  1.5688265   0.4732559   0.97662428  2.56641209]
[ 0.1598534   3.89629223  0.64958983 13.36785247  2.32780912]
[ 0.42698566  0.80514485  0.56665373  1.09587308  1.67347956]
[ 0.2838948   2.202237    1.37869827  4.60410715  1.71086105]
[ 0.78035166  1.60512953  4.57135142  0.29171497  1.68407789]
[ 0.51160321  4.40677454  0.56002416  0.60944511  2.35799916]
[ 1.75199153  4.77919792  2.27583646  2.34233463  4.15496087]
[ 0.29967375  0.52809239  1.15450426  0.44169007  0.28228097]
[ 4.96685956  0.33737322  0.89263101  0.28816276  1.56475979]
[ 1.55325581  1.54476397  0.34744119  1.72409657  0.30188723]
[ 1.34017127  3.21846689  2.84999097  0.52515398  3.70340494]
[ 0.37576968  0.32866619  0.30707583  0.37035277  0.48733443]
[ 2.92802902  3.15575998  0.78718648  0.73876489  0.72759469]
[ 1.2590207   0.71366085  0.38803326  0.59298956  0.35645918]
[ 0.6495907   0.20334378  0.93422194  1.24715397  0.90125516]
[ 0.32571082  2.77182156  1.57412946  0.36535186  0.36428596]
[ 0.11170005  0.82413611  1.40644447  3.08199424  0.78438564]
[ 1.72755615  0.52471046  0.49741086  0.71134379  4.72556869]
[ 2.98299067  1.05187251  5.24037619  0.94350829  0.87013363]
[ 0.35366485  1.0509003   0.62967831  6.15545064  1.0692153 ]
[ 0.590431    1.65053088  6.1524032   1.27460141  0.52389243]]
```

[39]:
```
mean=np.mean(dataset,axis=0)
sd=np.std(dataset,axis=0)
```

[41]:
```
print(mean)
print(sd)
```

```
[1.50035376 1.67675571 2.4740047  1.45547692 1.50578509]
[2.06369877 1.7842416  5.53262232 2.08527582 1.28175562]
```

[45]:
```
standardise=(dataset-mean)/sd
new_mean=np.mean(standardise,axis=0)
normalized=np.std(standardise,axis=0)
print(new_mean)
print(normalized)
```

```
[ 6.88338275e-17  2.63261635e-16 -1.11022302e-17 -9.15933995e-18
 -7.10542736e-17]
[1. 1. 1. 1. 1.]
```

[49]:
```
#Question 3

def vandermonde(N):
    vec=np.arange(N)+1
    vander=vec.reshape(N,1)
    vander=vander**np.arange(N)
    return vander
vander=vandermonde(12)
print(vander)
```

```
[[          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]]
```

[51]:
```
x=np.ones(12)
b=np.dot(vander,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]
```

[53]:
```
inverse=np.linalg.inv(vander)
res=np.dot(inverse,b)
print(res)
```

```
[0.99620819 1.00462723 0.99909973 1.00006104 0.99999666 1.00000048
 0.99999995 1.         1.         1.         1.         1.        ]
```

[55]:
```
#The expected result was that all the elements of the matrix would be exactly 1.␣
↪However, not all results come out to be 1.
```

[57]:
```
res_new=np.linalg.solve(vander,b)
print(res_new)
```

```
[0.99998827 1.00002951 0.99997139 1.00001427 0.99999595 1.00000068
 0.99999993 1.         1.         1.         1.         1.        ]
```

[59]:
```
#The result is closer to the expected value.
```

[63]:
```
#Github Link
```

[ ]:
```
https://github.com/TirthankarDatta/TirthankarDatta_24-52-28
```

[ ]:
```

```