2. (15 points) [Matrix Operations] [Ge Gao] Write code in Python to perform the following tasks. Please report your output and relevant code in the document file, and also include your code file (ends with extension .py) in the .zip file.

(a) (1 point) Generate a 10×10 identity matrix A.

Ans:

```
[[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

Code:

```
# a.)Generate a 10×10 identity matrix A.
print("a.) Generate a 10×10 identity matrix A.")
matrixA = numpy.identity(10)
print(matrixA)
```

(b) (1 point) Change all elements in the 7th column of A to 8.

Ans:

```
[[1. 0. 0. 0. 0. 0. 8. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 8. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 8. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 8. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 8. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 8. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 8. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 8. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 8. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 8. 0. 0. 1.]]
```

Code:

```
# b.)Change all elements in the 7th column of A to 8
print("\n" + "b.)Change all elements in the 7th column of A to 8")
matrixA[:, 6] = 8
print(matrixA)
```

(c) (1 point) Sum of all elements in the matrix (use a ”for/while loop”).

Ans: 89

Code:

```
# c.)Sum of all elements in the matrix (use a ”for/while loop
print("\n" + "c.)  Sum of all elements in the matrix (use a ”for/while loop”)")
sum = 0
for arr in matrixA:
    for elem in arr:
        sum = sum + elem
print(sum)
```

(d) (1 point) Transpose the matrix A (A = AT).

Ans:

```
[[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [8. 8. 8. 8. 8. 8. 8. 8. 8. 8.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

Code:

```
# d.)Transpose the matrix A (A = AT).
print("\n" + "d.)Transpose the matrix A (A = AT).")
transposeMatrixA = matrixA.transpose()
print(transposeMatrixA)
```

(e) (2 points) Calculate sum of the 1st row, the diagonal and the 2nd column in the matrix A.

Ans:

1st Row Sum: **9**

Diagonal Sum: **17**

2nd Column Sum : **1**

Code:

```
# e.)Calculate sum of the 1st row, the diagonal and the 2nd column in the matrix A
print(
    "\n"
    + "e.) Calculate sum of the 1st row, the diagonal and the 2nd column in the matrix A."
)
sumDiagonal = numpy.sum(numpy.diag(matrixA))
sumRow = numpy.sum(matrixA[0])
sumCol = numpy.sum(matrixA[:, 1])
print("1st Row Sum: " + str(sumRow))
print("Diagonal Sum: " + str(sumDiagonal))
print("2nd Column Sum: " + str(sumCol))
```

(f) (1 point) Generate a 10×10 matrix B following Gaussian Distribution with mean 12 and variance π.

Ans:

```
[[12.842767    4.837831   11.05785947  8.4517867  12.66693731 12.51113062
  11.56008017 12.3916969  12.51655958  9.56908921]
 [10.68217016 13.45266263 15.49608594 14.87046578 10.19315083 13.81743135
   8.33701739 13.17713867  8.76627384 12.42448212]
 [10.75514534 15.82690052 12.9076604  13.49368064  8.79017053 16.4307185
  16.17073061  9.58525565  5.61891158 13.05784879]
 [12.49357534  7.0045621   8.7846516  13.70313759  7.23563707  9.64531113
  13.81816363  7.20943913 12.02757004 11.40884302]
 [17.86069604 15.67559626  7.1505866  16.03595852  6.92794047 10.12213096
  12.11309526  5.0648832  11.4310811   9.42108136]
 [ 6.59118616 11.90059777 10.61277816 12.51126106 12.48006104 11.07871141
   7.42199904 10.10666212 13.06189217  9.50916922]
 [12.81300582 10.64380949 12.02652994  9.89771606  9.06945678 16.41258858
   9.75283728 19.45751985 12.0847348  11.77580942]
 [ 5.50045163 13.65286234 10.50113517  7.03867442  9.46186172  9.13844477
  12.6948596   9.11465961  6.18791989  9.39663556]
 [15.66686141 12.96819329  9.96646418  9.44255409 16.13103468 14.20564376
  11.48021815  7.16112656 13.06874886 10.12598987]
 [15.4153089  13.47576898 14.86889502 13.7786195   6.53323631 10.58421614
  14.28037581 10.75138001 10.84052987 17.06118596]]
```

Code:

```
# f.)Generate a 10×10 matrix B following Gaussian Distribution with mean 12 and variance π.
print(
    "\n"
    + "f.) Generate a 10×10 matrix B following Gaussian Distribution with mean 12 and variance pi."
)
matrixB = numpy.random.normal(12, numpy.pi, size=(10, 10))
print(matrixB)
```

(g) (2 points) From A and B, use matrix operations to get a new 3×10 matrix C such that, the first row of C is equal to the 1st row of B times the 8th row of A minus the 3rd column of B, the second row of C is equal to the 4th row of A minus the 7th column of B, and the third row of C is equal to the sum of the 10th column of A and 1st row of B.

Ans:

```
[[-11.05785947 -15.49608594 -12.9076604   -8.7846516   -7.1505866
  -10.61277816  80.45411145   1.89056173  -9.96646418 -14.86889502]
 [-11.56008017  -8.33701739 -16.17073061 -12.81816363 -12.11309526
   -7.42199904  -1.75283728 -12.6948596  -11.48021815 -14.28037581]
 [ 12.842767     4.837831    11.05785947   8.4517867   12.66693731
   12.51113062  11.56008017  12.3916969   12.51655958  10.56908921]]
```

Code:

```
# g.)From A and B, use matrix operations to get a new 3×10 matrix C such
#   that, the first row of C is equal to the 1st row of B times the 8th row of A minus
#   the 3rd column of B, the second row of C is equal to the 4th row of A minus the 7th
#   column of B, and the third row of C is equal to the sum of the 10th column of A
#   and 1st row of B
print(
    "\ng.) From A and B, use matrix operations to get a new 3×10 matrix C such"
    + "that, the first row of C is equal to the 1st row of B times the 8th row of A minus"
    + "the 3rd column of B, the second row of C is equal to the 4th row of A minus the 7th"
    + "column of B, and the third row of C is equal to the sum of the 10th column of A"
    + "and 1st row of B "
)
row1 = (matrixB[0] * matrixA[7]) - matrixB[:, 2]
row2 = matrixA[3] - matrixB[:, 6]
row3 = matrixA[:, 9] + matrixB[0]
matrixC = numpy.array([row1, row2, row3])
print(matrixC)
```

(h) (2 points) From C, using one matrix operation to get a new matrix D such that, the first column of D is equal to the first column of C times 3, the second column of D is equal to the second column of C times 4 and so on.

Ans:

```
[[ -33.17357841  -61.98434377  -64.538302    -52.70790959  -50.05410618
   -84.90222528  724.08700306   18.9056173  -109.63110599 -178.42674028]
 [ -34.68024052  -33.34806955  -80.85365307  -76.90898176  -84.79166685
   -59.37599229  -15.77553548 -126.94859595 -126.28239964 -171.36450967]
 [  38.528301     19.35132399   55.28929735   50.71072021   88.66856115
   100.08904498  104.04072157  123.91696904  137.68215539  126.82907047]]
```

Code:

```
# h.) From C, using one matrix operation to get a new matrix D such that,
#   the first column of D is equal to the first column of C times 3, the second column
#   of D is equal to the second column of C times 4 and so on.
print(
    "\nh.) From C, using one matrix operation to get a new matrix D such that,"
    + "the first column of D is equal to the first column of C times 3, the second column"
    + "of D is equal to the second column of C times 4 and so on."
)
matrixD = []
for i in range(0, len(matrixC)):
    count = 3
    matrixD.append(["", "", "", "", "", "", "", "", "", ""])
    for j in range(0, 10):
        matrixD[i][j] = matrixC[i][j] * count
        count = count + 1
matrixD = numpy.array(matrixD)
print(matrixD)
```

(i) (2 points) X = [2, 0, 2, 0]T , Y = [9, 8, 4, 3]T , Z = [7, 1, 7, 9]T , M = [6, 4, 8, 2]T , N = [5, 4, 9, 3]T , P = [1, 5, 2, 4]T . Compute the covariance matrix of X, Y ,Z, M, N and P. Then compute the Pearson correlation coefficients between X and Y.

Ans:

```
[[ 1.33333333  0.66666667  1.33333333  2.66666667  2.33333333 -2.        ]
 [ 0.66666667  8.66666667 -6.          1.33333333 -1.33333333 -1.        ]
 [ 1.33333333 -6.         12.          0.         1.         -3.33333333]
 [ 2.66666667  1.33333333  0.          6.66666667  6.33333333 -3.33333333]
 [ 2.33333333 -1.33333333  1.          6.33333333  6.91666667 -2.66666667]
 [-2.         -1.         -3.33333333 -3.33333333 -2.66666667  3.33333333]]
0.19611613513818402
```

Correlation Coefficient: .19611613513818402

Code:

```
# i.)Compute the covariance matrix of X, Y ,Z, M, N
#    and P. Then compute the Pearson correlation coefficients between X and Y.
print(
    "\ni.)Compute the covariance matrix of X, Y ,Z, M, N"
    + "and P. Then compute the Pearson correlation coefficients between X and Y."
)

X = numpy.array([2, 0, 2, 0])
Y = numpy.array([9, 8, 4, 3])
Z = numpy.array([7, 1, 7, 9])
M = numpy.array([6, 4, 8, 2])
N = numpy.array([5, 4, 9, 3])
P = numpy.array([1, 5, 2, 4])
testCovMatrix = numpy.cov([X, Y, Z, M, N, P])
print(testCovMatrix)
r = numpy.corrcoef(X, Y)
print(r[0][1])
```

(j) (2 points) Given the equation: mean(x^2) = (mean(x)^2 2+sigma^2(x)) where x = [19, 12, 16, 6, 11, 27, 1, 29]T . Please determine whether the equation holds when:

i. σ(x) is the population standard deviation. Show your work.

Ans:

```
mean(x^2): 311.125
mean(x)^2: 228.765625
Population Standard Deviation: 82.359375

mean(x^2) = (mean(x)^2+populationStd^2(x))
311.125 = (228.765625 + 82.359375)
311.125 = 311.125
-The above equation holds for the population standard deviation
```

Code:

```
#   i. σ(x) is the population standard deviation. Show your work.
print(
    "\nj.Given the equation: mean(x^2) = (mean(x)^2+std^2(x)) x = [19, 12, 16, 6, 11, 27, 1, 29]^T."
    + "Please determine whether the equation holds when:"
)
print("i. std(x) is the population standard deviation. Show your work.")
x = numpy.array([19, 12, 16, 6, 11, 27, 1, 29])
# gives x^2
xSquare = numpy.multiply(x, x)
# gives mean(x^2)
xSquareMean = numpy.mean(xSquare)
print("mean(x^2): " + str(xSquareMean))
# gives mean(x)^2
xMeanSquare = numpy.mean(x) ** 2
print("mean(x)^2: " + str(xMeanSquare))
# gives population standard deviation
populationStd = numpy.std(x) ** 2
print("Population Standard Deviation: " + str(populationStd))
print("\nmean(x^2) = (mean(x)^2+populationStd^2(x))")
print(
    str(xSquareMean) + " = (" + str(xMeanSquare) + " + " + str(populationStd) + ")"
)
popVal = xMeanSquare + populationStd
print(str(xSquareMean) + " = " + str(popVal))
print("-The above equation holds for the population standard deviation")
```

ii. σ(x) is the sample standard deviation. Show your work.

Ans:

```
mean(x^2): 311.125
mean(x)^2: 228.765625
Sample Standard Deviation: 94.125

mean(x^2) = (mean(x)^2+sampleStd^2(x))
311.125 = 322.890625
-The above equation doesn't hold for the sample standard deviation
```

Code:

```
#   ii. σ(x) is the sample standard deviation. Show your work.
print("\nii. std(x) is the sample standard deviation. Show your work.")
print("mean(x^2): " + str(xSquareMean))
print("mean(x)^2: " + str(xMeanSquare))
# gives the sample standard deviation
sampleStd = numpy.std(x, ddof=1) ** 2
print("Sample Standard Deviation: " + str(sampleStd))
print("\nmean(x^2) = (mean(x)^2+sampleStd^2(x))")
sampleVal = xMeanSquare + sampleStd
print(str(xSquareMean) + " = " + str(sampleVal))
print("-The above equation doesn't hold for the sample standard deviation")
```