Name: Prakriti Khadka

Student Id : 2431868

TO - DO - Task

Please complete all the problem listed below.

4.1 Warming Up Exercise: Basic Vector and Matrix Operation with Numpy.

Problem - 1: Array Creation:

Complete the following Tasks:

1. Initialize an empty array with size 2X2.

2. Initialize an all one array with size 4X2.

3. Return a new array of given shape and type, filled with fill value.{Hint: np.full}

4. Return a new array of zeros with same shape and type as a given array.{Hint: np.zeros like}

5. Return a new array of ones with same shape and type as a given array.{Hint: np.ones like}

6. For an existing list new\_list = [1,2,3,4] convert to an numpy array.{Hint: np.array()}

Code / Implementation:

import numpy as np

empty\_array = np.empty((2, 2))

ones\_array = np.ones((4, 2))

filled\_array = np.full((3, 3), fill\_value=7)

given\_array = np.array([[1, 2], [3, 4]])

zeros\_like\_array = np.zeros\_like(given\_array)

ones\_like\_array = np.ones\_like(given\_array)

new\_list = [1, 2, 3, 4]

numpy\_array = np.array(new\_list)

print("Empty Array:")

print(empty\_array)

print("All Ones Array:")

print(ones\_array)

print("Filled Array:")

print(filled\_array)

print("Zeros Like Array:")

print(zeros\_like\_array)

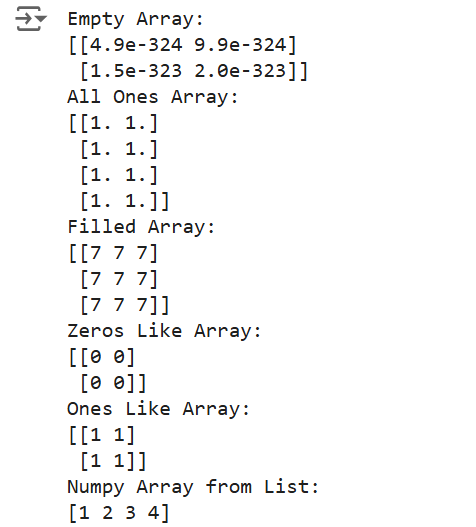
print("Ones Like Array:")

print(ones\_like\_array)

print("Numpy Array from List:")

print(numpy\_array)

Output:



Problem - 2: Array Manipulation: Numerical Ranges and Array indexing:

Complete the following tasks:

1. Create an array with values ranging from 10 to 49. {Hint:np.arrange()}.

2. Create a 3X3 matrix with values ranging from 0 to 8.

{Hint:look for np.reshape()}

3. Create a 3X3 identity matrix.{Hint:np.eye()}

4. Create a random array of size 30 and find the mean of the array.

{Hint:check for np.random.random() and array.mean() function}

5. Create a 10X10 array with random values and find the minimum and maximum values.

6. Create a zero array of size 10 and replace 5th element with 1.

7. Reverse an array arr = [1,2,0,0,4,0].

8. Create a 2d array with 1 on border and 0 inside.

9. Create a 8X8 matrix and fill it with a checkerboard pattern.

Code/Implementation:

import numpy as np

array\_range = np.arange(10, 50)

matrix\_3x3 = np.arange(9).reshape(3, 3)

identity\_matrix = np.eye(3)

random\_array = np.random.random(30)

mean\_value = random\_array.mean()

random\_10x10 = np.random.random((10, 10))

min\_value = random\_10x10.min()

max\_value = random\_10x10.max()

zero\_array = np.zeros(10)

zero\_array[4] = 1

arr = np.array([1, 2, 0, 0, 4, 0])

reversed\_arr = arr[::-1]

border\_array = np.ones((5, 5))

border\_array[1:-1, 1:-1] = 0

checkerboard = np.zeros((8, 8))

checkerboard[1::2, ::2] = 1

checkerboard[::2, 1::2] = 1

print("Array with values ranging from 10 to 49:")

print(array\_range)

print("\n3x3 matrix with values ranging from 0 to 8:")

print(matrix\_3x3)

print("\n3x3 identity matrix:")

print(identity\_matrix)

print("\nRandom array of size 30:")

print(random\_array)

print("\nMean of the random array:")

print(mean\_value)

print("\n10x10 random array:")

print(random\_10x10)

print("\nMinimum value in the 10x10 array:")

print(min\_value)

print("\nMaximum value in the 10x10 array:")

print(max\_value)

print("\nZero array of size 10 with the 5th element replaced by 1:")

print(zero\_array)

print("\nReversed array:")

print(reversed\_arr)

print("\n2D array with 1 on border and 0 inside:")

print(border\_array)

print("\n8x8 checkerboard pattern:")

print(checkerboard)

Output :  
Array with values ranging from 10 to 49:

[10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33

34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49]

3x3 matrix with values ranging from 0 to 8:

[[0 1 2]

[3 4 5]

[6 7 8]]

3x3 identity matrix:

[[1. 0. 0.]

[0. 1. 0.]

[0. 0. 1.]]

Random array of size 30:

[0.69270383 0.83843976 0.19678953 0.70388459 0.32705829 0.49361419

0.19647977 0.10552253 0.55195123 0.78134586 0.39866246 0.45962538

0.04302173 0.97431671 0.24624075 0.72151378 0.6927768 0.65213464

0.0600908 0.9552885 0.32225487 0.07947193 0.2738891 0.77231793

0.81484335 0.37170314 0.50620118 0.88151308 0.77465345 0.43431245]

Mean of the random array:

0.5107540542002101

10x10 random array:

[[0.4042317 0.21880536 0.49525905 0.07019069 0.96858589 0.75764876

0.14327276 0.09453595 0.3745453 0.3433275 ]

[0.79988864 0.60450526 0.72884778 0.82465231 0.75452862 0.44040675

0.12976562 0.08694415 0.58007453 0.84214857]

[0.10534285 0.44973045 0.32400586 0.78778144 0.1594611 0.57601344

0.91065159 0.05568839 0.0933502 0.86452926]

[0.82251917 0.45331485 0.70133617 0.94703879 0.99789422 0.34697726

0.36957759 0.59961769 0.17947544 0.58038337]

[0.58948633 0.81639291 0.99440981 0.9711369 0.37028184 0.63387918

0.09534471 0.91805834 0.88985765 0.32510792]

[0.27492142 0.32916602 0.25022671 0.49391878 0.05428996 0.4436753

0.4652434 0.38685779 0.75929415 0.28973774]

[0.37902317 0.755488 0.03324678 0.67494775 0.28488267 0.79916634

0.42100527 0.3769761 0.62745 0.09784302]

[0.46094559 0.75425333 0.81597539 0.12950216 0.45765966 0.11234639

0.98786572 0.83447724 0.38220028 0.63337293]

[0.55778682 0.3964391 0.40128187 0.19327996 0.24105141 0.53511818

0.82787218 0.83587131 0.04302918 0.58931367]

[0.97927588 0.46470296 0.92462478 0.12275561 0.17160663 0.89562331

0.82117118 0.84909211 0.66603722 0.42166737]]

Minimum value in the 10x10 array:

0.03324677716848112

Maximum value in the 10x10 array:

0.9978942181523891

Zero array of size 10 with the 5th element replaced by 1:

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

Reversed array:

[0 4 0 0 2 1]

2D array with 1 on border and 0 inside:

[[1. 1. 1. 1. 1.]

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

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

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

[1. 1. 1. 1. 1.]]

8x8 checkerboard pattern:

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

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

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

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

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

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

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

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

Problem - 3: Array Operations:

For the following arrays:

x = np.array([[1,2],[3,5]]) and y = np.array([[5,6],[7,8]]);

v = np.array([9,10]) and w = np.array([11,12]);

Complete all the task using numpy:

1. Add the two array.

2. Subtract the two array.

3. Multiply the array with any integers of your choice.

4. Find the square of each element of the array.

5. Find the dot product between: v(and)w ; x(and)v ; x(and)y.

6. Concatenate x(and)y along row and Concatenate v(and)w along column.

{Hint:try np.concatenate() or np.vstack() functions.

7. Concatenate x(and)v; if you get an error, observe and explain why did you get the error?

Code/Implementation:

import numpy as np

x = np.array([[1, 2], [3, 5]])

y = np.array([[5, 6], [7, 8]])

v = np.array([9, 10])

w = np.array([11, 12])

add\_result = np.add(x, y)

print("Addition of x and y:")

print(add\_result)

subtract\_result = np.subtract(x, y)

print("\nSubtraction of x and y:")

print(subtract\_result)

multiply\_result = x \* 2

print("\nMultiplying x by 2:")

print(multiply\_result)

square\_result = np.square(x)

print("\nSquare of each element in x:")

print(square\_result)

dot\_vw = np.dot(v, w)

dot\_xv = np.dot(x, v)

dot\_xy = np.dot(x, y)

print("\nDot product of v and w:", dot\_vw)

print("Dot product of x and v:", dot\_xv)

print("Dot product of x and y:")

print(dot\_xy)

concat\_xy\_row = np.concatenate((x, y), axis=0)

concat\_vw\_column = np.concatenate((v.reshape(-1, 1), w.reshape(-1, 1)), axis=1)

print("\nConcatenating x and y along row:")

print(concat\_xy\_row)

print("\nConcatenating v and w along column:")

print(concat\_vw\_column)

try:

    concat\_xv = np.concatenate((x, v), axis=0)

    print("\nConcatenating x and v:")

    print(concat\_xv)

except Exception as e:

    print("\nError while concatenating x and v:", str(e))

Output:

Addition of x and y:

[[ 6 8]

[10 13]]

Subtraction of x and y:

[[-4 -4]

[-4 -3]]

Multiplying x by 2:

[[ 2 4]

[ 6 10]]

Square of each element in x:

[[ 1 4]

[ 9 25]]

Dot product of v and w: 219

Dot product of x and v: [29 77]

Dot product of x and y:

[[19 22]

[50 58]]

Concatenating x and y along row:

[[1 2]

[3 5]

[5 6]

[7 8]]

Concatenating v and w along column:

[[ 9 11]

[10 12]]

Error while concatenating x and v: all the input arrays must have same number of dimensions, but the array at index 0 has 2 dimension(s) and the array at index 1 has 1 dimension(s)

Problem - 4: Matrix Operations:

• For the following arrays:

A = np.array([[3,4],[7,8]]) and B = np.array([[5,3],[2,1]]);

Prove following with Numpy:

1. Prove A.A−1 = I.

2. Prove AB ̸= BA.

3. Prove (AB)

T = BTAT

.

• Solve the following system of Linear equation using Inverse Methods.

2x − 3y + z = −1

x − y + 2z = −3

3x + y − z = 9

{Hint: First use Numpy array to represent the equation in Matrix form. Then Solve for: AX = B}

• Now: solve the above equation using np.linalg.inv function.{Explore more about ”linalg” function

of Numpy}

Code/Implementation:

import numpy as np

A = np.array([[3, 4], [7, 8]])

B = np.array([[5, 3], [2, 1]])

A\_inv = np.linalg.inv(A)

I = np.eye(2)

result = np.dot(A, A\_inv)

is\_identity = np.allclose(result, I)

print("Is it identity ?  ",is\_identity)

AB = np.dot(A, B)

BA = np.dot(B, A)

print("AB:")

print(AB)

print("\nBA:")

print(BA)

is\_equal = np.allclose(AB, BA)

print("\nIs AB equal to BA?", is\_equal)

AB\_T = np.transpose(np.dot(A, B))

BT\_AT = np.dot(np.transpose(B), np.transpose(A))

print("Transpose of AB:")

print(AB\_T)

print("\nBT \* AT:")

print(BT\_AT)

is\_equal = np.allclose(AB\_T, BT\_AT)

print("\nIs (AB)^T equal to B^T \* A^T?", is\_equal)

A\_matrix = np.array([[2, -3, 1], [1, -1, 2], [3, 1, -1]])

B\_matrix = np.array([-1, -3, 9])

X = np.linalg.inv(A\_matrix).dot(B\_matrix)

x, y, z = X

print(f"Value of x is: {x}")

print(f"Value of y is: {y}")

print(f"Value of z is: {z}")

import numpy as np

A\_matrix = np.array([[2, -3, 1], [1, -1, 2], [3, 1, -1]])

B\_matrix = np.array([-1, -3, 9])

X = np.linalg.solve(A\_matrix, B\_matrix)

x, y, z = X

print(f"Value of x is: {x}")

print(f"Value of y is: {y}")

print(f"Value of z is: {z}")

Output:

Is it identity ? True

AB:

[[23 13]

[51 29]]

BA:

[[36 44]

[13 16]]

Is AB equal to BA? False

Transpose of AB:

[[23 51]

[13 29]]

BT \* AT:

[[23 51]

[13 29]]

Is (AB)^T equal to B^T \* A^T? True

Value of x is: 2.0

Value of y is: 1.0

Value of z is: -2.0

Value of x is: 2.0

Value of y is: 1.0000000000000002

Value of z is: -2.0