# **Question 1: Linear Algebra Fundamentals**

Here are the following is the python code for given operation in numpy.

import numpy as np

def Matrix\_Multiplication (a,b):

    return a@b

def E\_wise\_multiplication(a,b):

    return a\*b

def determinant(a):

    det = np.linalg.det(a)

    return det

def transpose(a):

    transposed = a.T

    return transposed

def main():

    a = np.array([[2,3],[1,4]])

    b = np.array([[5,2],[3,1]])

    print(f"Matrix Multiplication : ",Matrix\_Multiplication(a,b))

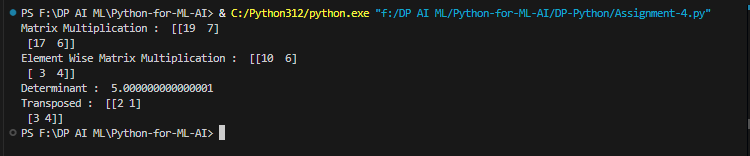
    print(f"Element Wise Matrix Multiplication : ",E\_wise\_multiplication(a,b))

    print(f"Determinant : ",determinant(a))

    print(f"Transposed : ",transpose(a))

main()

**Output:**

****

# **Question 2: NumPy Array Manipulation**

# Question 2

import numpy as np

array = np.random.randint(10,101,(6,6))

print(f"Random array : {array}")

diagonal\_elemtnts = np.diag(array)

print(f"Diagonal elements of an array : {diagonal\_elemtnts}")

new\_matrix = array.astype(float)

even\_pos = (new\_matrix %2 == 0)

new\_matrix[even\_pos] = np.sqrt(new\_matrix[even\_pos])

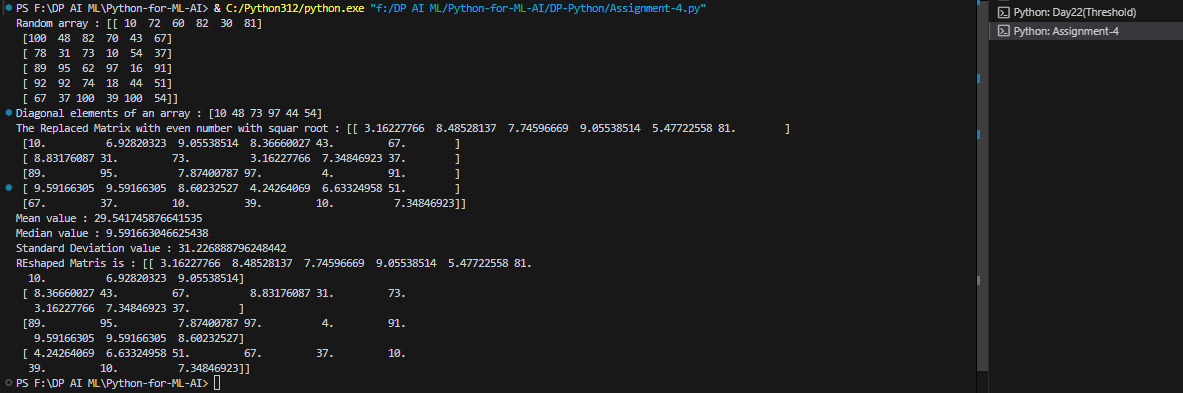
print(f"The Replaced Matrix with even number with squar root : {new\_matrix}")

print(f"Mean value : {np.mean(new\_matrix)}")

print(f"Median value : {np.median(new\_matrix)}")

print(f"Standard Deviation value : {np.std(new\_matrix)}")

**Output**:



# **Question 3: Image Loading and Basic Processing**

Here is the Open CV code in Python to do given operation with the image:

import cv2 as cv

image = cv.imread("img\image.png")

gray\_img = cv.cvtColor(image,cv.COLOR\_BGR2GRAY)

cv.imshow("Gray Image",gray\_img)

cv.imshow("Original Image",image)

cv.waitKey(0)

cv.destroyAllWindows()

cv.imwrite("img\copy\_img.png",gray\_img)

print(f"Image Height : ",image.shape[0])

print(f"Image Weight : ",image.shape[1])

print(f"Image Channel : ",image.shape[2])

**Output:**

**A black screen with blue and white text

AI-generated content may be incorrect.**

## Question No 4

# Question 4

import cv2 as cv

import os

def resize\_image\_operations(image\_path, output\_dir="resized\_outputs"):

    image = cv.imread(image\_path)

    if image is None:

        print("Error: Image not found.")

        return

    os.makedirs(output\_dir, exist\_ok=True)

    original\_height, original\_width = image.shape[:2]

    print(f"Original Dimensions: {original\_width}x{original\_height}")

    # Resize 50% smaller

    small = cv.resize(image, (0, 0), fx=0.5, fy=0.5)

    cv.imshow("50% Smaller Image", small)

    cv.imwrite(f"{output\_dir}/resized\_50\_percent\_smaller.jpg", small)

    # Resize 200% larger

    large = cv.resize(image, (0, 0), fx=2.0, fy=2.0)

    cv.imshow("200% Larger Image", large)

    cv.imwrite(f"{output\_dir}/resized\_200\_percent\_larger.jpg", large)

    #  300x300

    fixed = cv.resize(image, (300, 300))

    cv.imwrite(f"{output\_dir}/resized\_fixed\_300x300.jpg", fixed)

    # Maintain aspect ratio

    aspect\_ratio = original\_height / original\_width

    new\_height = int(300 \* aspect\_ratio)

    aspect\_resized = cv.resize(image, (300, new\_height))

    cv.imshow("Aspect Ratio Resized Image", aspect\_resized)

    cv.imwrite(f"{output\_dir}/resized\_300\_aspect\_ratio.jpg", aspect\_resized)

    cv.waitKey(0)

    cv.destroyAllWindows()

resize\_image\_operations("img\image0.jpg")