CSE 352 - ASSIGNMENT 3

Arnav Jain - 220002018

Objection Questions

Question 1

option A) [5, 7, 9]

Question 2

option D) Both A and B

Question 3

option A) np.sqrt()

Question 4

option A) Adding two arrays of different shapes automatically

Question 5

option A) Computes the sum of each column

Question 1

Code and Result:

q1.ipynb

q1.ipynb > # Replace the first column with all ones

Generate + Code + Markdown | Run All Restart Execute Group 1 Ex

```
import numpy as np

# Random 4x4 matrix
matrix = np.random.rand(4, 4)

print("Original Matrix:\n", matrix)
```

[1]

```
Original Matrix:
 [[0.48436866 0.41841968 0.32553889 0.653545  ]
 [0.76992958 0.11671264 0.88446617 0.18803465]
 [0.10674165 0.88022166 0.50484133 0.7016461 ]
 [0.68429226 0.11441269 0.80177283 0.92864889]]
```

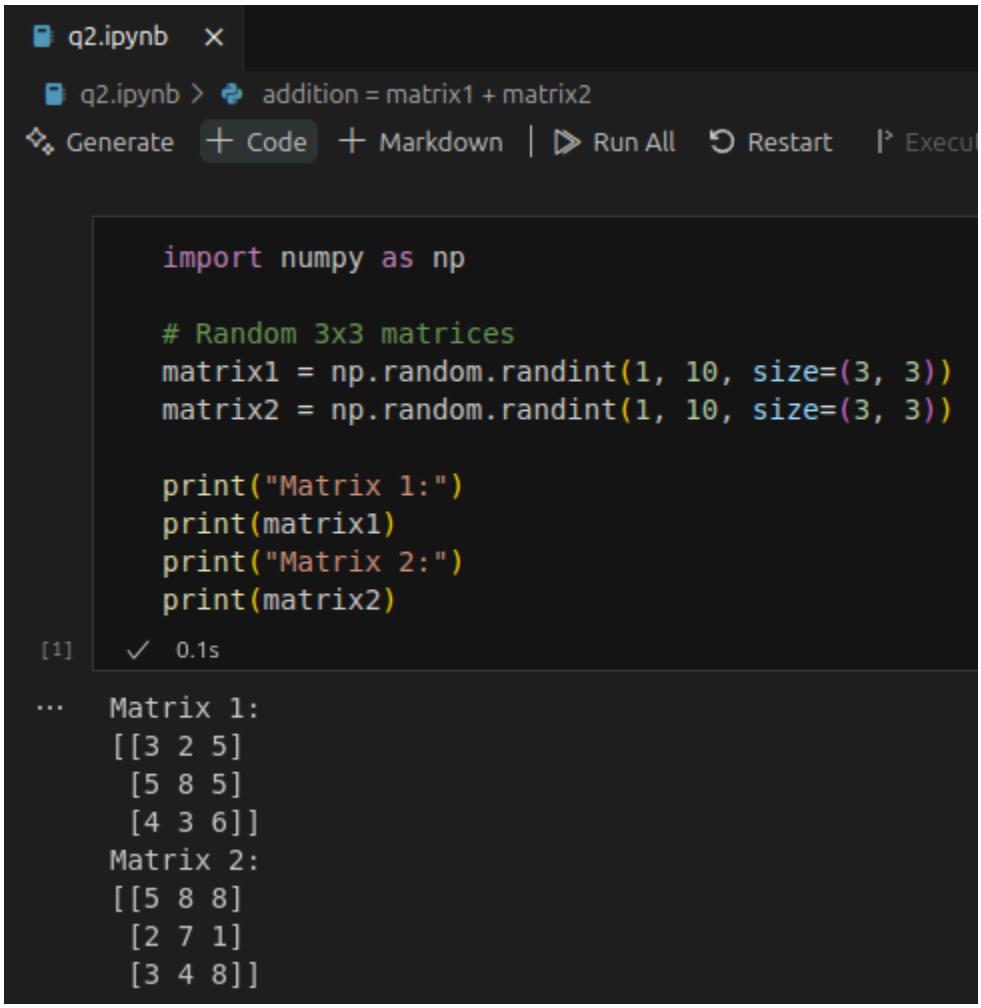
```
# Replace the first column with all ones
matrix[:, 0] = 1

print("Modified Matrix:\n", matrix)
```

[2]

```
Modified Matrix:
 [[1.         0.41841968 0.32553889 0.653545  ]
 [1.         0.11671264 0.88446617 0.18803465]
 [1.         0.88022166 0.50484133 0.7016461 ]
 [1.         0.11441269 0.80177283 0.92864889]]
```

Question 2

Code and Result:



```
addition = matrix1 + matrix2
subtraction = matrix1 - matrix2
multiplication = matrix1 * matrix2
division = matrix1 / matrix2

print("\nElement-wise Addition:\n", addition)
print("\nElement-wise Subtraction:\n", subtraction)
print("\nElement-wise Multiplication:\n", multiplication)
print("\nElement-wise Division:\n", division)
```

[2] ✓ 0.0s

```
...
    Element-wise Addition:
     [[ 8 10 13]
     [ 7 15  6]
     [ 7  7 14]]

    Element-wise Subtraction:
     [[-2 -6 -3]
     [ 3  1  4]
     [ 1 -1 -2]]

    Element-wise Multiplication:
     [[15 16 40]
     [10 56  5]
     [12 12 48]]

    Element-wise Division:
     [[0.6        0.25       0.625     ]
     [2.5        1.14285714 5.        ]
     [1.33333333 0.75       0.75      ]]
```

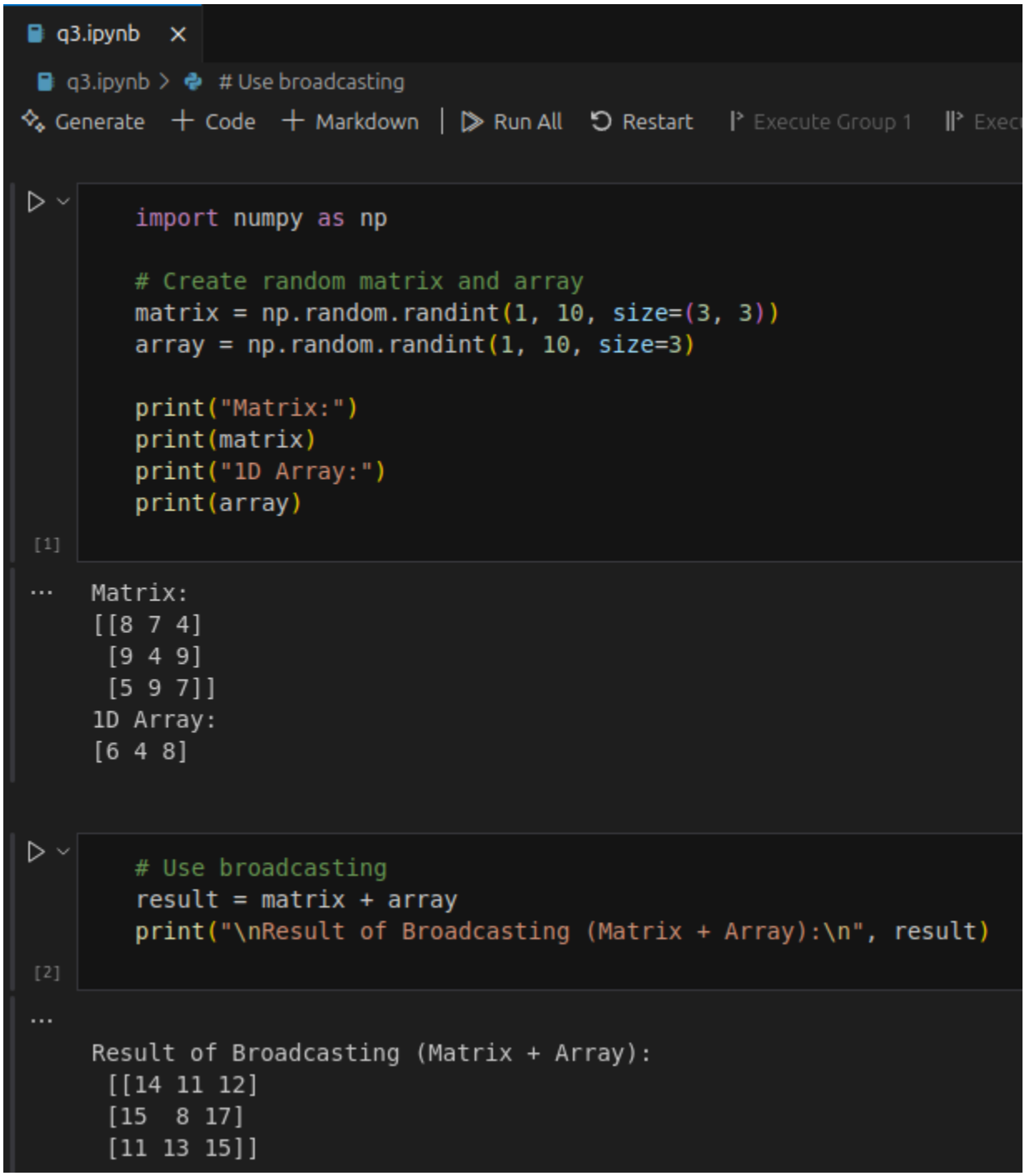
```
    # Calculate the determinant
    det_matrix1 = np.linalg.det(matrix1)
    det_matrix2 = np.linalg.det(matrix2)

    print(f"\nDeterminant of Matrix 1: {det_matrix1}")
    print(f"Determinant of Matrix 2: {det_matrix2}")
```

[3] ✓ 0.0s

```
...
    Determinant of Matrix 1: -5.99999999999999
    Determinant of Matrix 2: 51.999999999999964
```

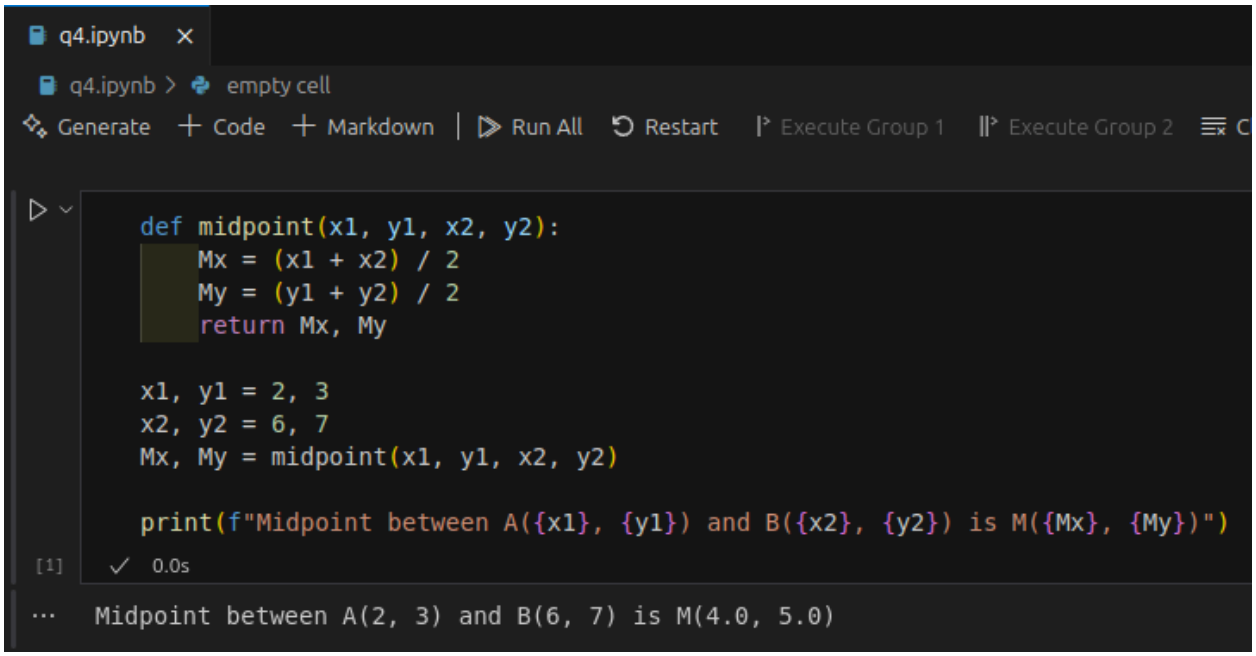
Question 3

Code and Result:



Question 4

Code and Result:



Question 5

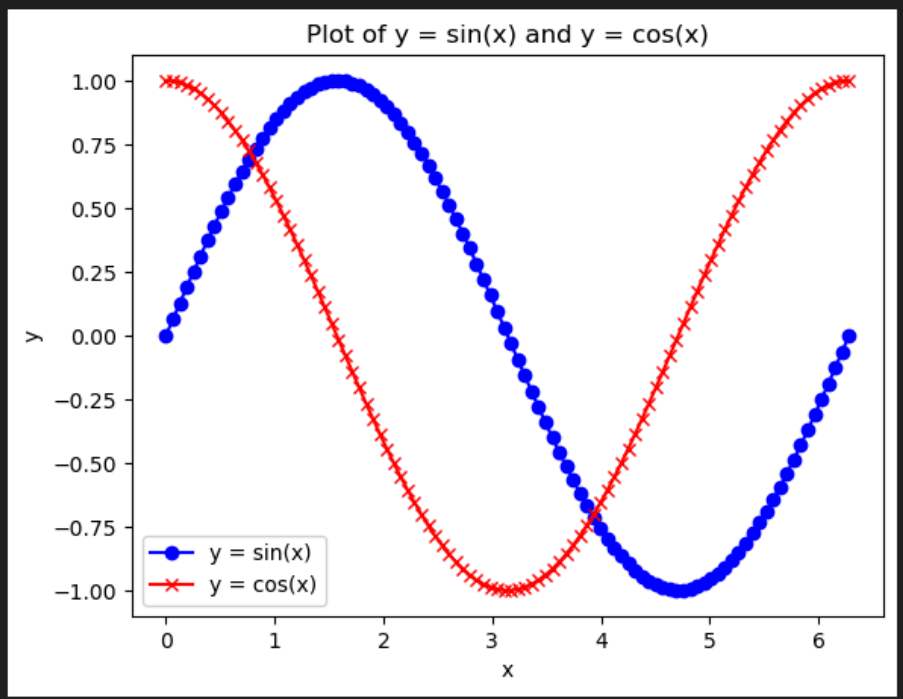
Code and Result:

q5.ipynb

Generate + Code + Markdown | Run All Restart Execute Group 1 Execute Group

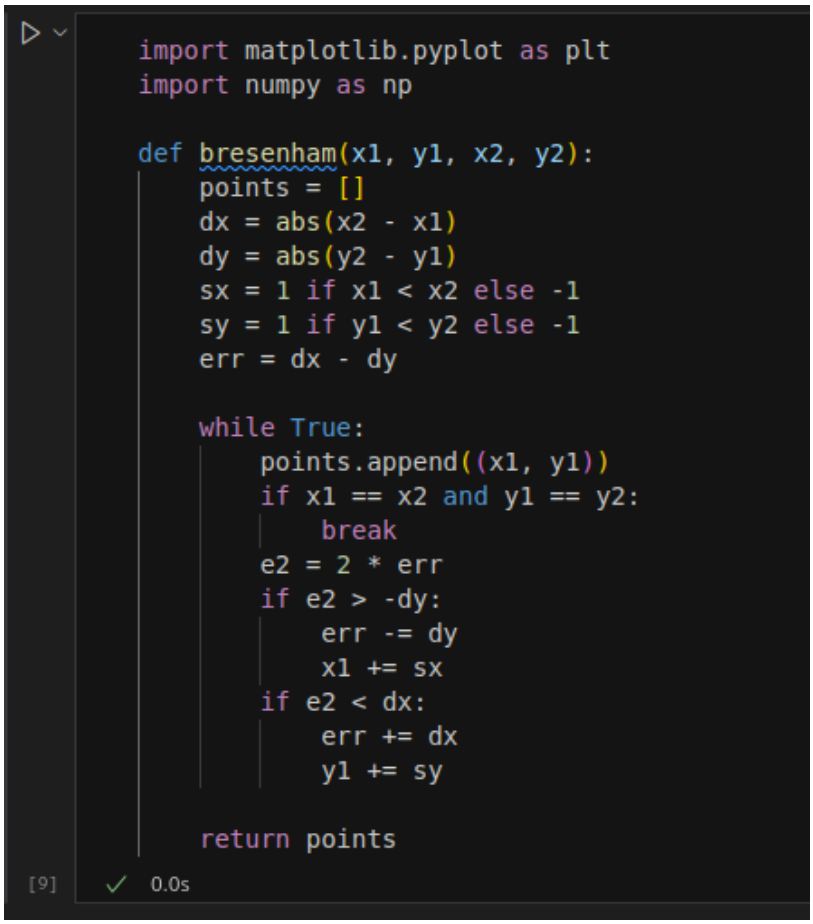
```
import numpy as np
import matplotlib.pyplot as plt

# Create array from 0 to 2π
x = np.linspace(0, 2 * np.pi, 100)

y_sin = np.sin(x)
y_cos = np.cos(x)

plt.plot(x, y_sin, label='y = sin(x)', color='blue', marker='o')
plt.plot(x, y_cos, label='y = cos(x)', color='red', marker='x')

plt.xlabel('x')
plt.ylabel('y')
plt.title('Plot of y = sin(x) and y = cos(x)')
plt.legend()

plt.show()
```

[1] ✓ 2.3s



Question 6

Code and Result:



```python
import matplotlib.pyplot as plt
import numpy as np

def bresenham(x1, y1, x2, y2):
    points = []
    dx = abs(x2 - x1)
    dy = abs(y2 - y1)
    sx = 1 if x1 < x2 else -1
    sy = 1 if y1 < y2 else -1
    err = dx - dy

    while True:
        points.append((x1, y1))
        if x1 == x2 and y1 == y2:
            break
        e2 = 2 * err
        if e2 > -dy:
            err -= dy
            x1 += sx
        if e2 < dx:
            err += dx
            y1 += sy

    return points
```

[9] ✓ 0.0s

```
x1, y1 = 2, 3
x2, y2 = 10, 8
line_points = bresenham(x1, y1, x2, y2)

x_vals, y_vals = zip(*line_points)

# Create the plot
fig, ax = plt.subplots()
ax.set_xticks(np.arange(min(x_vals)-1, max(x_vals)+2, 1))
ax.set_yticks(np.arange(min(y_vals)-1, max(y_vals)+2, 1))
ax.grid(True, which='both', linestyle='--', color='gray', alpha=0.5)
ax.set_xlim(min(x_vals)-1, max(x_vals)+1)
ax.set_ylim(min(y_vals)-1, max(y_vals)+1)
ax.set_aspect('equal')

# Highlight grid cells where the pixels are active
for x, y in line_points:
    ax.add_patch(plt.Rectangle((x, y), 1, 1, color='gray', alpha=0.3))

ax.scatter(x_vals, y_vals, color='gray', alpha=0.6, marker='o', label="Activated Pixels")
ax.plot(x_vals, y_vals, color='green', marker='o', label=f"Line from ({x1},{y1}) to ({x2},{y2})")

# Title, labels, and legend
ax.set_title(f"Bresenham Line: ({x1},{y1}) to ({x2},{y2})")
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.legend()

plt.show()
```

[10] ✓ 0.2s

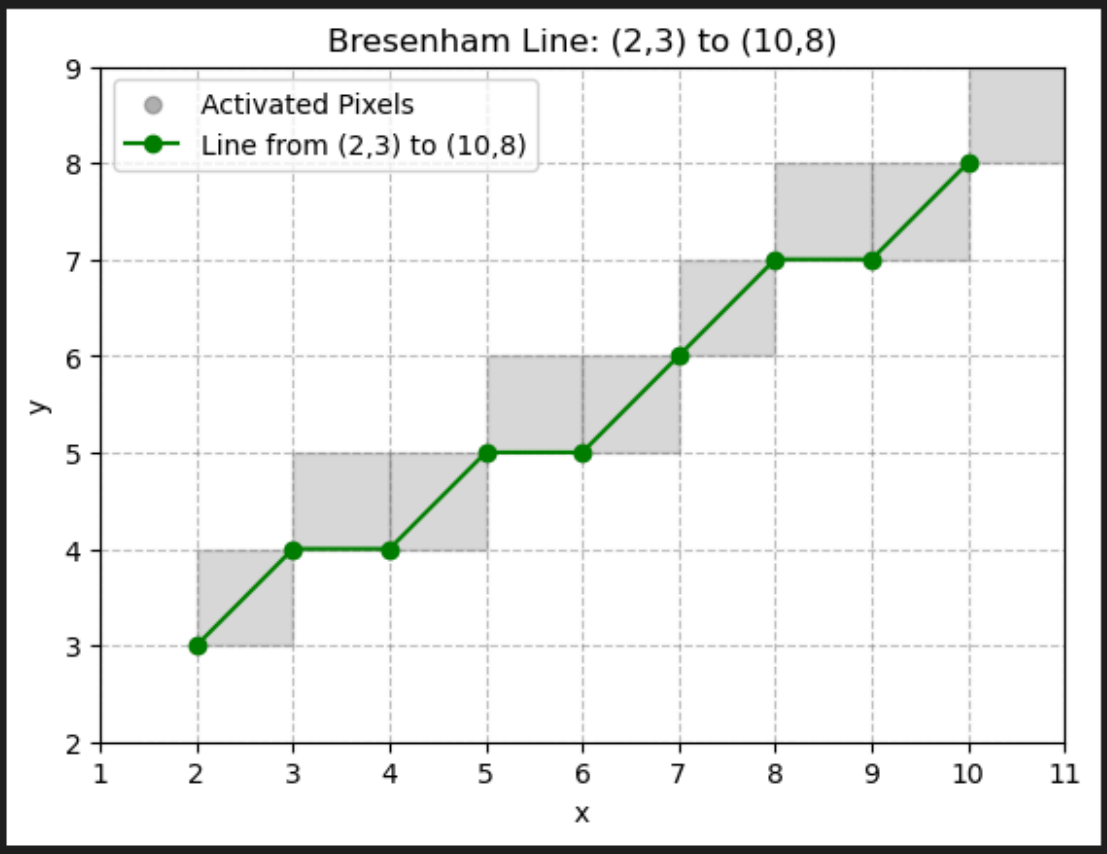
q6.ipynb

q6.ipynb

Generate | Code | Markdown | Run All | Restart | Execute Group 1 | Execute G

```
plt.show()
```

[10] ✓ 0.2s




```
# Output the points
print(f"Points on the line from ({x1}, {y1}) to ({x2}, {y2}):")
for point in line_points:
    print(point)
```

[11] ✓ 0.0s

```
Points on the line from (2, 3) to (10, 8):
(2, 3)
(3, 4)
(4, 4)
(5, 5)
(6, 5)
(7, 6)
(8, 7)
(9, 7)
(10, 8)
```

For code , refer GitHub

https://github.com/arnavjain2710/Computer-Graphics-Lab/tree/main/LAB%203