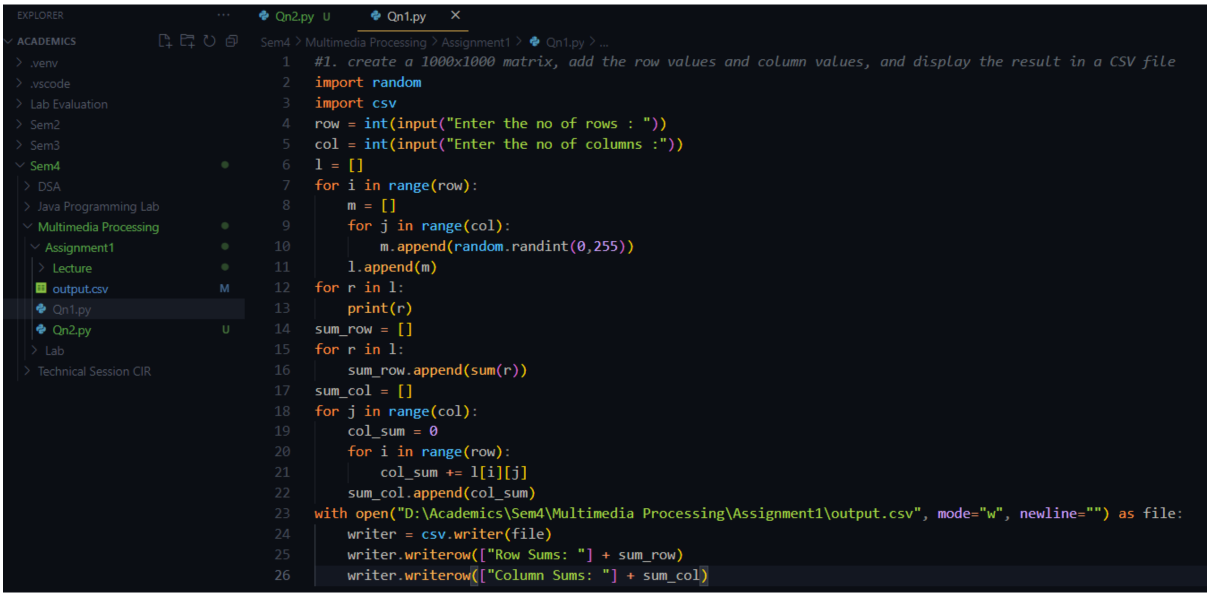
Multimedia Processing

Assignment – 1

Praneesh R V

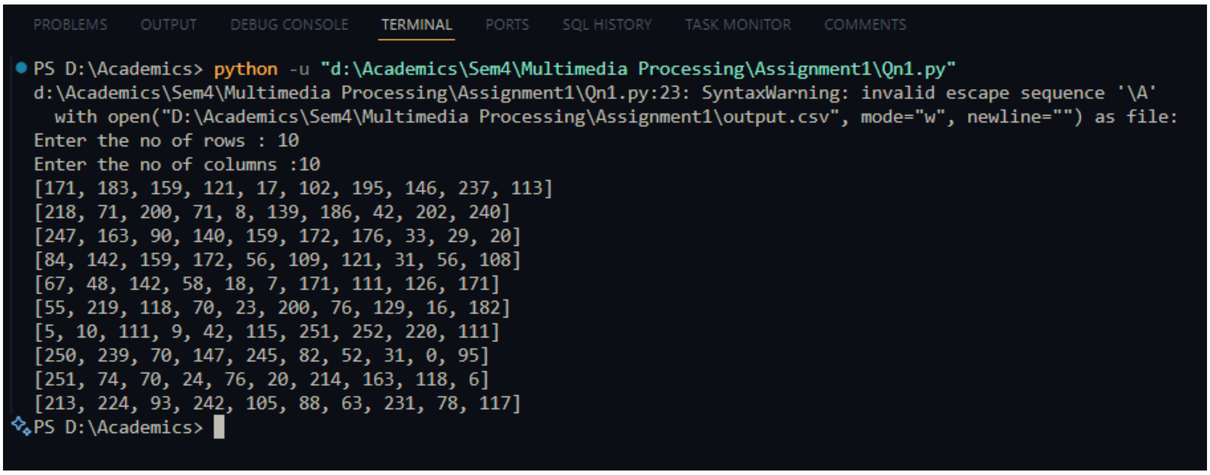
CB.SC.U4CYS23036

Qn1. Code:

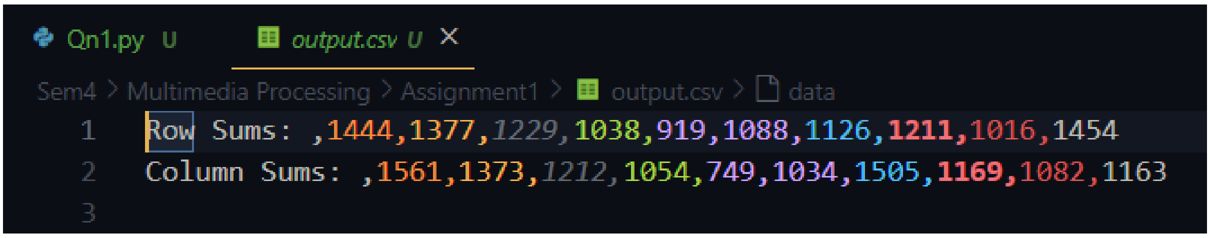


```python
#1. create a 1000x1000 matrix, add the row values and column values, and display the result in a CSV file
import random
import csv
row = int(input("Enter the no of rows : "))
col = int(input("Enter the no of columns :"))
l = []
for i in range(row):
    m = []
    for j in range(col):
        m.append(random.randint(0,255))
    l.append(m)
for r in l:
    print(r)
sum_row = []
for r in l:
    sum_row.append(sum(r))
sum_col = []
for j in range(col):
    col_sum = 0
    for i in range(row):
        col_sum += l[i][j]
    sum_col.append(col_sum)
with open("D:\Academics\Sem4\Multimedia Processing\Assignment1\output.csv", mode="w", newline="") as file:
    writer = csv.writer(file)
    writer.writerow(["Row Sums: "] + sum_row)
    writer.writerow(["Column Sums: "] + sum_col)
```

Output:



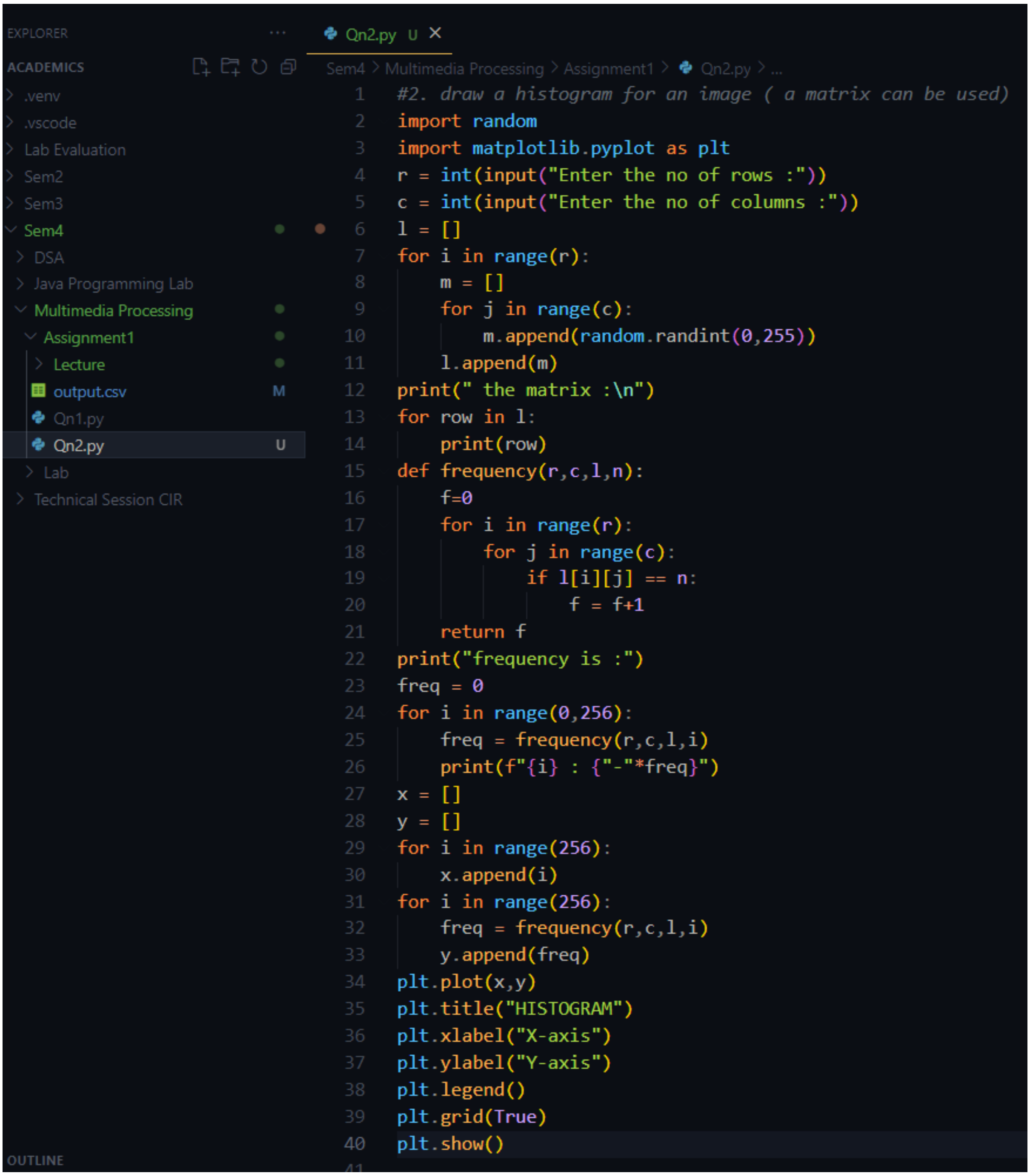
```
PS D:\Academics> python -u "d:\Academics\Sem4\Multimedia Processing\Assignment1\Qn1.py"
d:\Academics\Sem4\Multimedia Processing\Assignment1\Qn1.py:23: SyntaxWarning: invalid escape sequence '\A'
  with open("D:\Academics\Sem4\Multimedia Processing\Assignment1\output.csv", mode="w", newline="") as file:
Enter the no of rows : 10
Enter the no of columns :10
[171, 183, 159, 121, 17, 102, 195, 146, 237, 113]
[218, 71, 200, 71, 8, 139, 186, 42, 202, 240]
[247, 163, 90, 140, 159, 172, 176, 33, 29, 20]
[84, 142, 159, 172, 56, 109, 121, 31, 56, 108]
[67, 48, 142, 58, 18, 7, 171, 111, 126, 171]
[55, 219, 118, 70, 23, 200, 76, 129, 16, 182]
[5, 10, 111, 9, 42, 115, 251, 252, 220, 111]
[250, 239, 70, 147, 245, 82, 52, 31, 0, 95]
[251, 74, 70, 24, 76, 20, 214, 163, 118, 6]
[213, 224, 93, 242, 105, 88, 63, 231, 78, 117]
PS D:\Academics>
```


Qn1.py U
output.csv U
Sem4 > Multimedia Processing > Assignment1 > output.csv > data

```
Row Sums: ,1444,1377,1229,1038,919,1088,1126,1211,1016,1454
Column Sums: ,1561,1373,1212,1054,749,1034,1505,1169,1082,1163
```

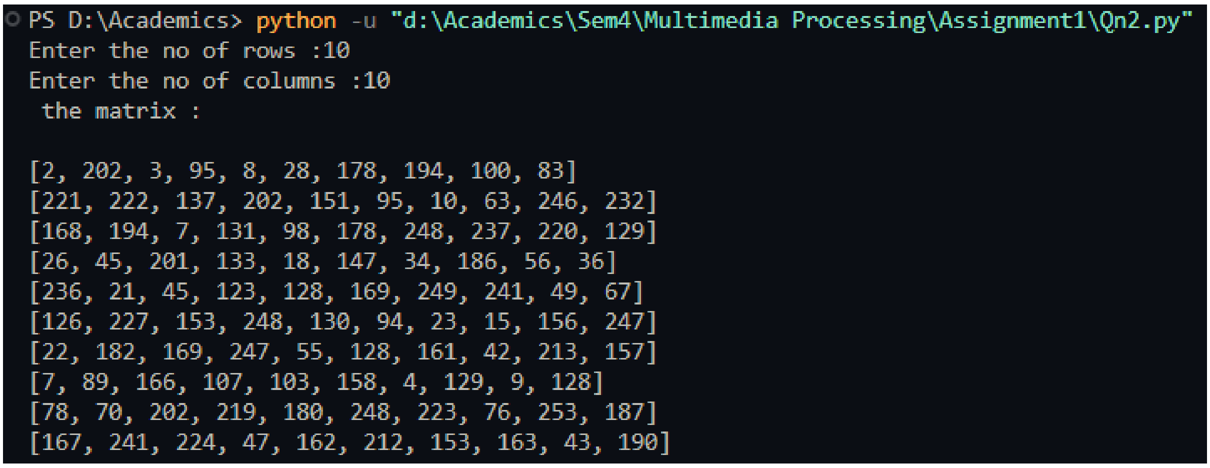
Qn2 - Histogram

Code:

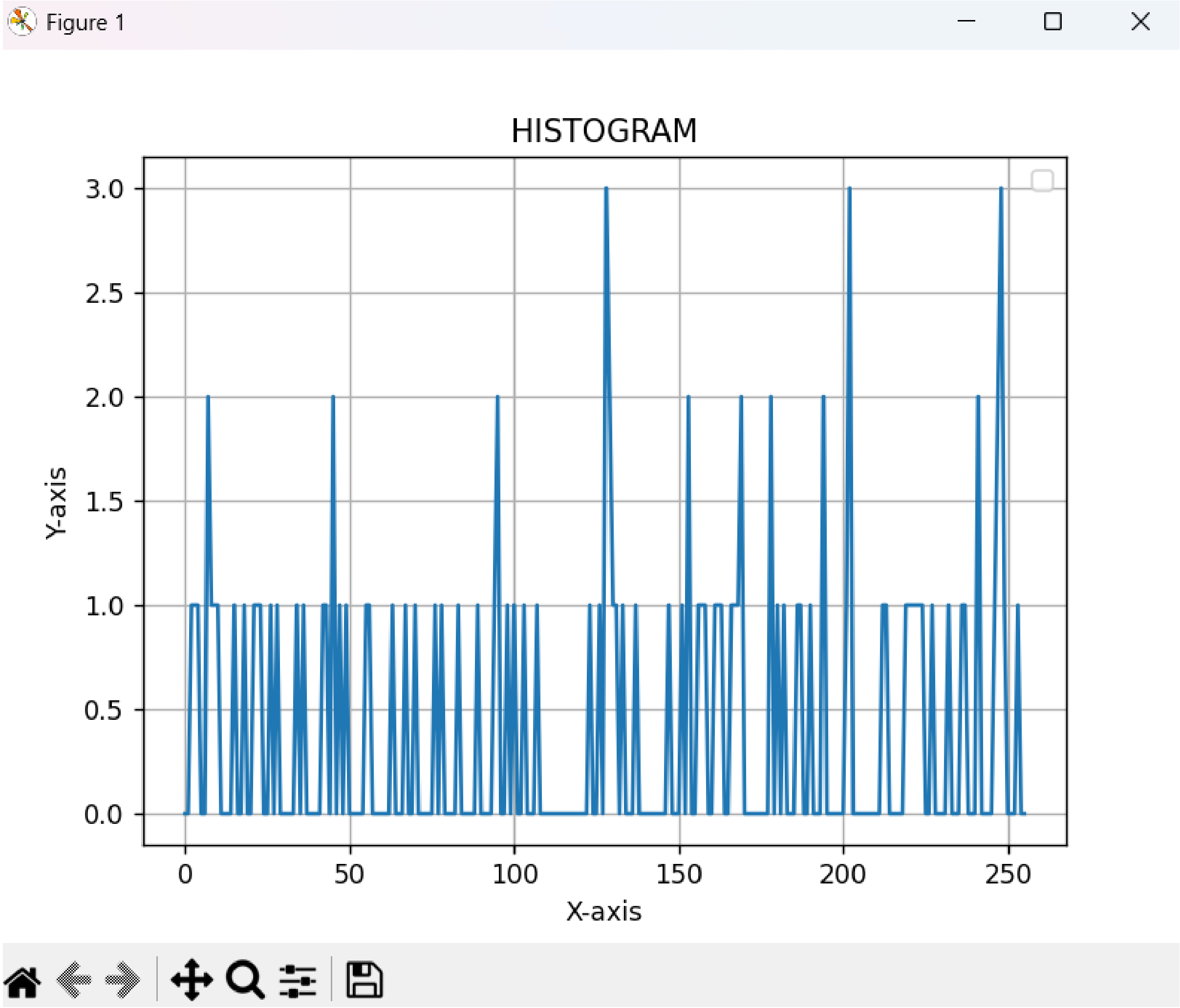


```python
#2. draw a histogram for an image ( a matrix can be used)
import random
import matplotlib.pyplot as plt
r = int(input("Enter the no of rows :"))
c = int(input("Enter the no of columns :"))
l = []
for i in range(r):
    m = []
    for j in range(c):
        m.append(random.randint(0,255))
    l.append(m)
print(" the matrix :\n")
for row in l:
    print(row)
def frequency(r,c,l,n):
    f=0
    for i in range(r):
        for j in range(c):
            if l[i][j] == n:
                f = f+1
    return f
print("frequency is :")
freq = 0
for i in range(0,256):
    freq = frequency(r,c,l,i)
    print(f"{i} : {"-"*freq}")
x = []
y = []
for i in range(256):
    x.append(i)
for i in range(256):
    freq = frequency(r,c,l,i)
    y.append(freq)
plt.plot(x,y)
plt.title("HISTOGRAM")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.legend()
plt.grid(True)
plt.show()
```

Output:

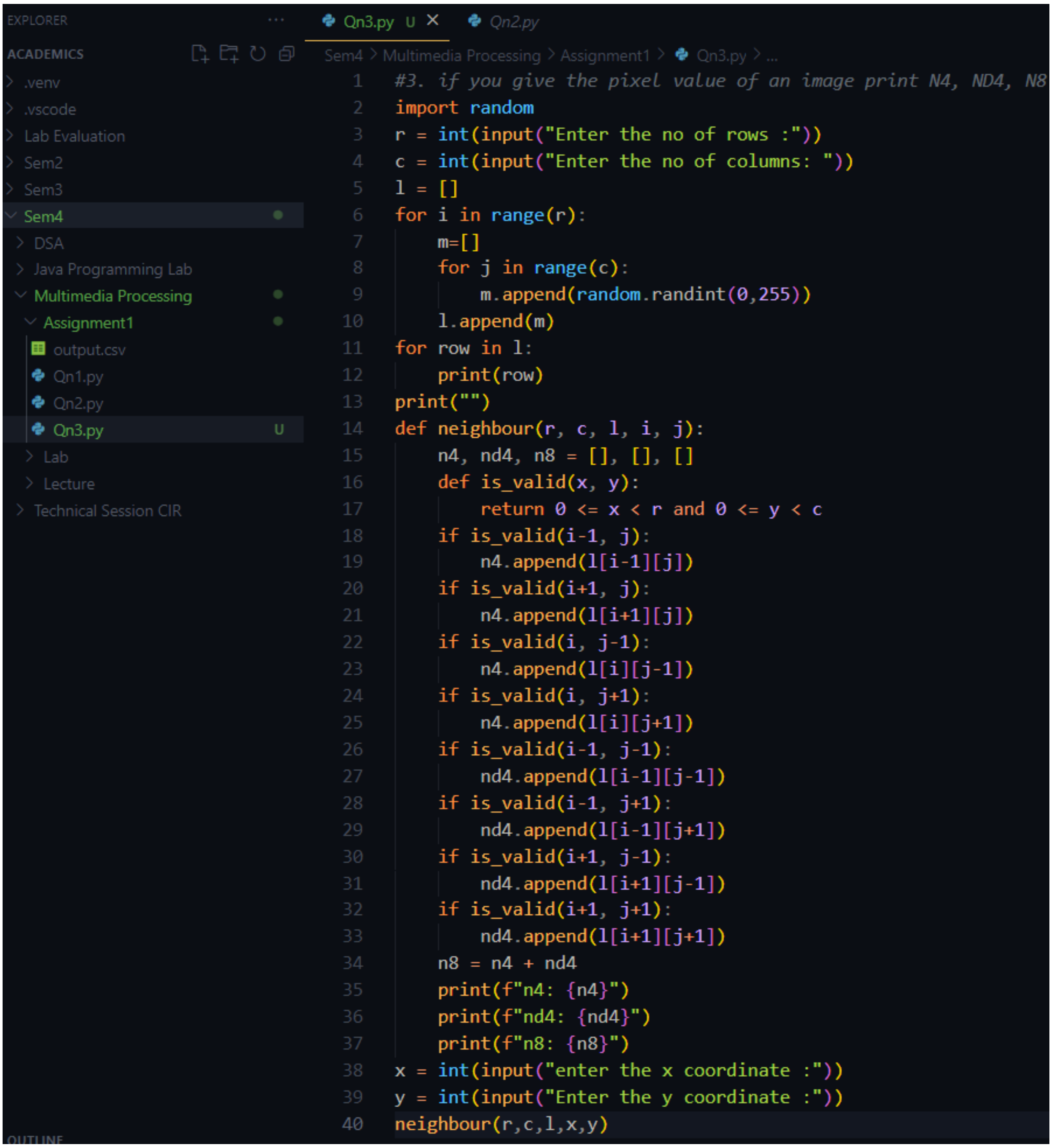


```
PS D:\Academics> python -u "d:\Academics\Sem4\Multimedia Processing\Assignment1\Qn2.py"
Enter the no of rows :10
Enter the no of columns :10
 the matrix :

[2, 202, 3, 95, 8, 28, 178, 194, 100, 83]
[221, 222, 137, 202, 151, 95, 10, 63, 246, 232]
[168, 194, 7, 131, 98, 178, 248, 237, 220, 129]
[26, 45, 201, 133, 18, 147, 34, 186, 56, 36]
[236, 21, 45, 123, 128, 169, 249, 241, 49, 67]
[126, 227, 153, 248, 130, 94, 23, 15, 156, 247]
[22, 182, 169, 247, 55, 128, 161, 42, 213, 157]
[7, 89, 166, 107, 103, 158, 4, 129, 9, 128]
[78, 70, 202, 219, 180, 248, 223, 76, 253, 187]
[167, 241, 224, 47, 162, 212, 153, 163, 43, 190]
```



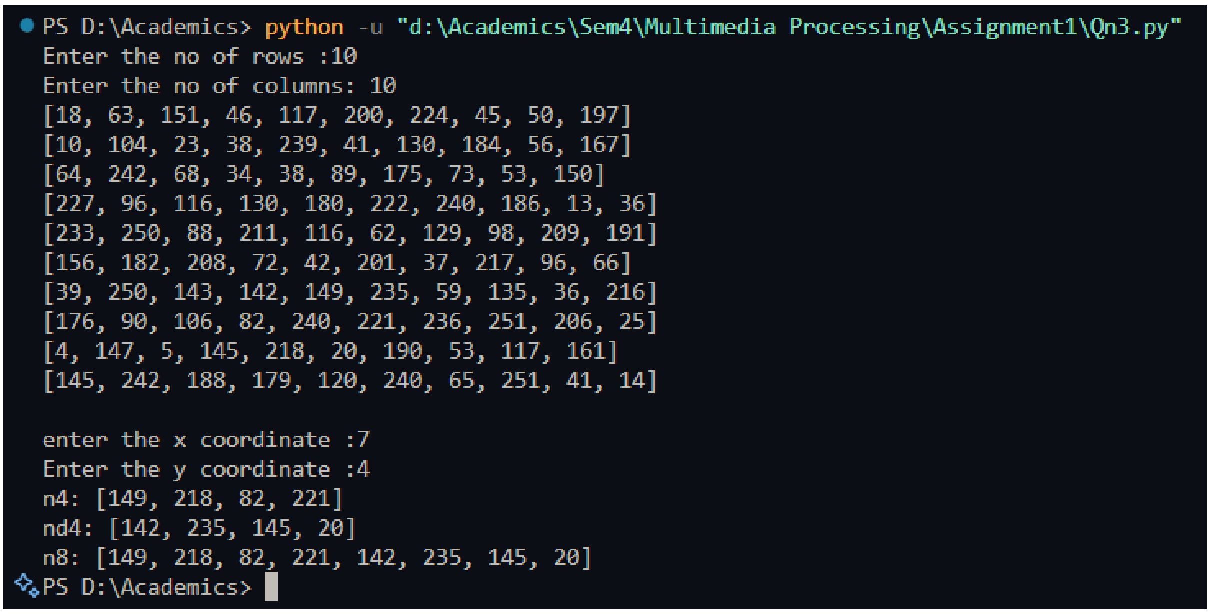
Qn3,

Code:



```python
#3. if you give the pixel value of an image print N4, ND4, N8
import random
r = int(input("Enter the no of rows :"))
c = int(input("Enter the no of columns: "))
l = []
for i in range(r):
    m=[]
    for j in range(c):
        m.append(random.randint(0,255))
    l.append(m)
for row in l:
    print(row)
print("")
def neighbour(r, c, l, i, j):
    n4, nd4, n8 = [], [], []
    def is_valid(x, y):
        return 0 <= x < r and 0 <= y < c
    if is_valid(i-1, j):
        n4.append(l[i-1][j])
    if is_valid(i+1, j):
        n4.append(l[i+1][j])
    if is_valid(i, j-1):
        n4.append(l[i][j-1])
    if is_valid(i, j+1):
        n4.append(l[i][j+1])
    if is_valid(i-1, j-1):
        nd4.append(l[i-1][j-1])
    if is_valid(i-1, j+1):
        nd4.append(l[i-1][j+1])
    if is_valid(i+1, j-1):
        nd4.append(l[i+1][j-1])
    if is_valid(i+1, j+1):
        nd4.append(l[i+1][j+1])
    n8 = n4 + nd4
    print(f"n4: {n4}")
    print(f"nd4: {nd4}")
    print(f"n8: {n8}")
x = int(input("enter the x coordinate :"))
y = int(input("Enter the y coordinate :"))
neighbour(r,c,l,x,y)
```

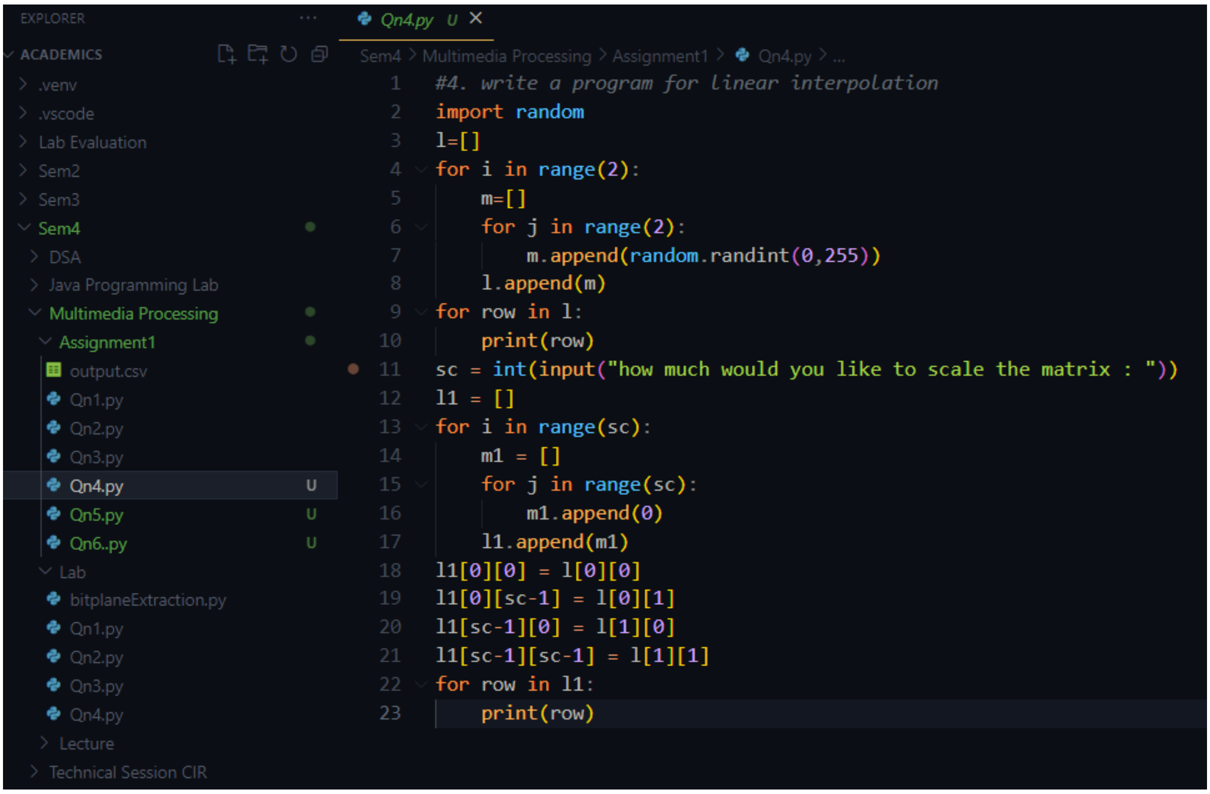
Output:



```
PS D:\Academics> python -u "d:\Academics\Sem4\Multimedia Processing\Assignment1\Qn3.py"
Enter the no of rows :10
Enter the no of columns: 10
[18, 63, 151, 46, 117, 200, 224, 45, 50, 197]
[10, 104, 23, 38, 239, 41, 130, 184, 56, 167]
[64, 242, 68, 34, 38, 89, 175, 73, 53, 150]
[227, 96, 116, 130, 180, 222, 240, 186, 13, 36]
[233, 250, 88, 211, 116, 62, 129, 98, 209, 191]
[156, 182, 208, 72, 42, 201, 37, 217, 96, 66]
[39, 250, 143, 142, 149, 235, 59, 135, 36, 216]
[176, 90, 106, 82, 240, 221, 236, 251, 206, 25]
[4, 147, 5, 145, 218, 20, 190, 53, 117, 161]
[145, 242, 188, 179, 120, 240, 65, 251, 41, 14]

enter the x coordinate :7
Enter the y coordinate :4
n4: [149, 218, 82, 221]
nd4: [142, 235, 145, 20]
n8: [149, 218, 82, 221, 142, 235, 145, 20]
PS D:\Academics> 
```

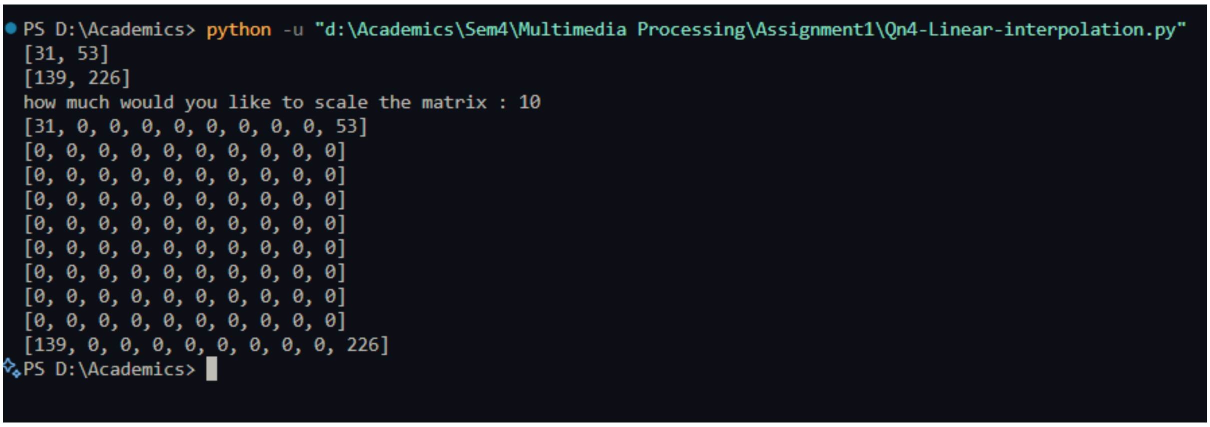
Qn4 - Linear Interpolation

Code:



```python
#4. write a program for linear interpolation
import random
l=[]
for i in range(2):
    m=[]
    for j in range(2):
        m.append(random.randint(0,255))
    l.append(m)
for row in l:
    print(row)
sc = int(input("how much would you like to scale the matrix : "))
l1 = []
for i in range(sc):
    m1 = []
    for j in range(sc):
        m1.append(0)
    l1.append(m1)
l1[0][0] = l[0][0]
l1[0][sc-1] = l[0][1]
l1[sc-1][0] = l[1][0]
l1[sc-1][sc-1] = l[1][1]
for row in l1:
    print(row)
```

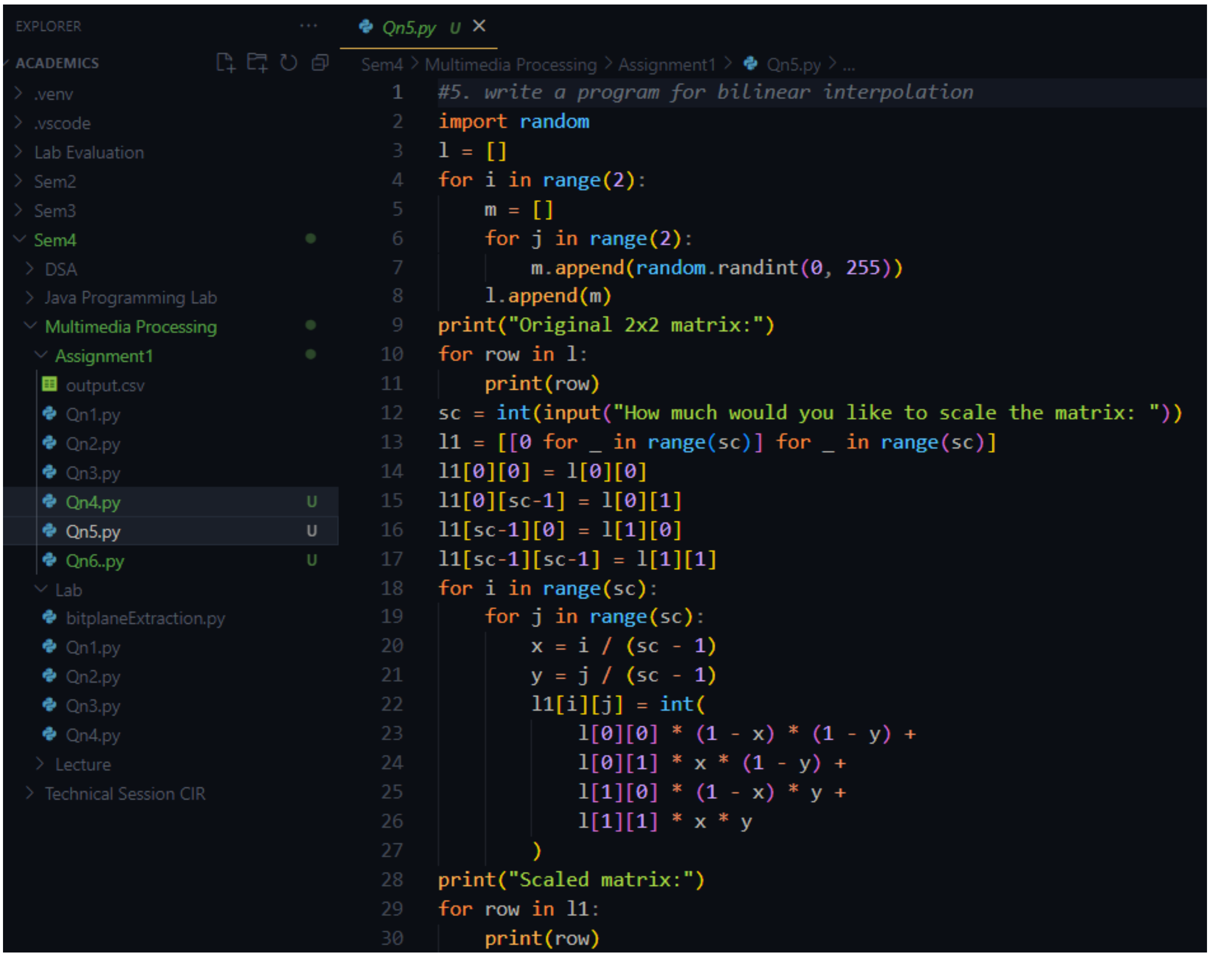
Output:



```
PS D:\Academics> python -u "d:\Academics\Sem4\Multimedia Processing\Assignment1\Qn4-Linear-interpolation.py"
[31, 53]
[139, 226]
how much would you like to scale the matrix : 10
[31, 0, 0, 0, 0, 0, 0, 0, 0, 53]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[139, 0, 0, 0, 0, 0, 0, 0, 0, 226]
PS D:\Academics> 
```

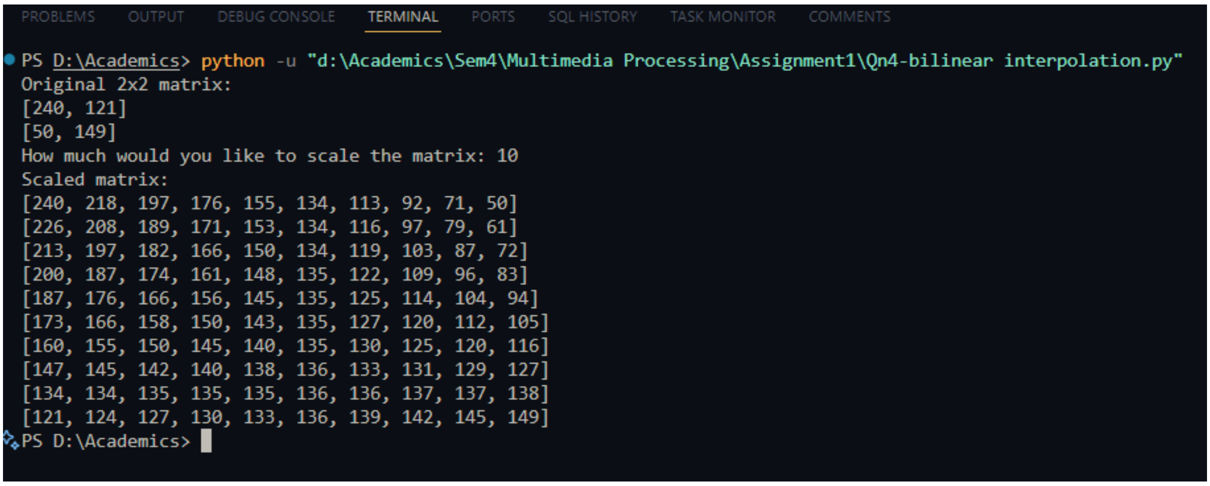
Qn5 - bilinear interpolation

Code:



```python
#5. write a program for bilinear interpolation
import random
l = []
for i in range(2):
    m = []
    for j in range(2):
        m.append(random.randint(0, 255))
    l.append(m)
print("Original 2x2 matrix:")
for row in l:
    print(row)
sc = int(input("How much would you like to scale the matrix: "))
l1 = [[0 for _ in range(sc)] for _ in range(sc)]
l1[0][0] = l[0][0]
l1[0][sc-1] = l[0][1]
l1[sc-1][0] = l[1][0]
l1[sc-1][sc-1] = l[1][1]
for i in range(sc):
    for j in range(sc):
        x = i / (sc - 1)
        y = j / (sc - 1)
        l1[i][j] = int(
            l[0][0] * (1 - x) * (1 - y) +
            l[0][1] * x * (1 - y) +
            l[1][0] * (1 - x) * y +
            l[1][1] * x * y
        )
print("Scaled matrix:")
for row in l1:
    print(row)
```

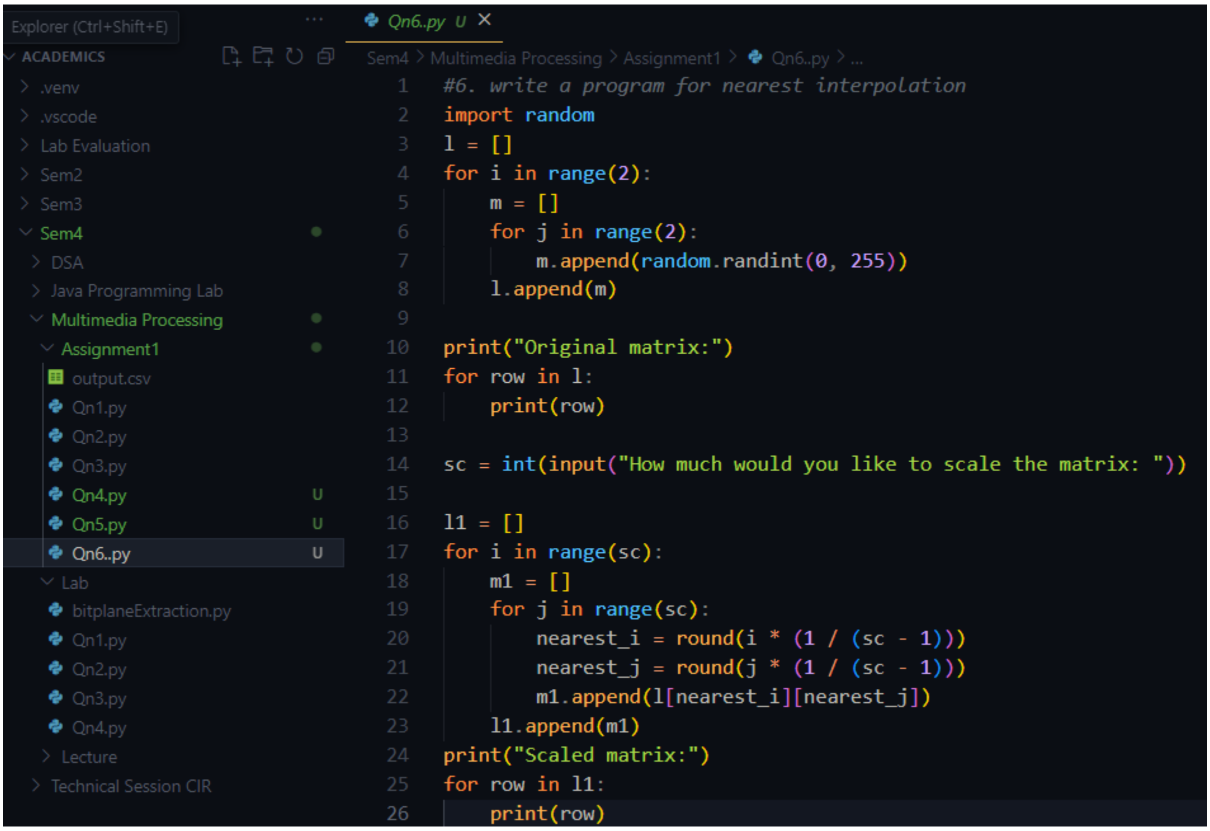
Output:



```
PS D:\Academics> python -u "d:\Academics\Sem4\Multimedia Processing\Assignment1\Qn4-bilinear interpolation.py"
Original 2x2 matrix:
[240, 121]
[50, 149]
How much would you like to scale the matrix: 10
Scaled matrix:
[240, 218, 197, 176, 155, 134, 113, 92, 71, 50]
[226, 208, 189, 171, 153, 134, 116, 97, 79, 61]
[213, 197, 182, 166, 150, 134, 119, 103, 87, 72]
[200, 187, 174, 161, 148, 135, 122, 109, 96, 83]
[187, 176, 166, 156, 145, 135, 125, 114, 104, 94]
[173, 166, 158, 150, 143, 135, 127, 120, 112, 105]
[160, 155, 150, 145, 140, 135, 130, 125, 120, 116]
[147, 145, 142, 140, 138, 136, 133, 131, 129, 127]
[134, 134, 135, 135, 135, 136, 136, 137, 137, 138]
[121, 124, 127, 130, 133, 136, 139, 142, 145, 149]
PS D:\Academics> 
```

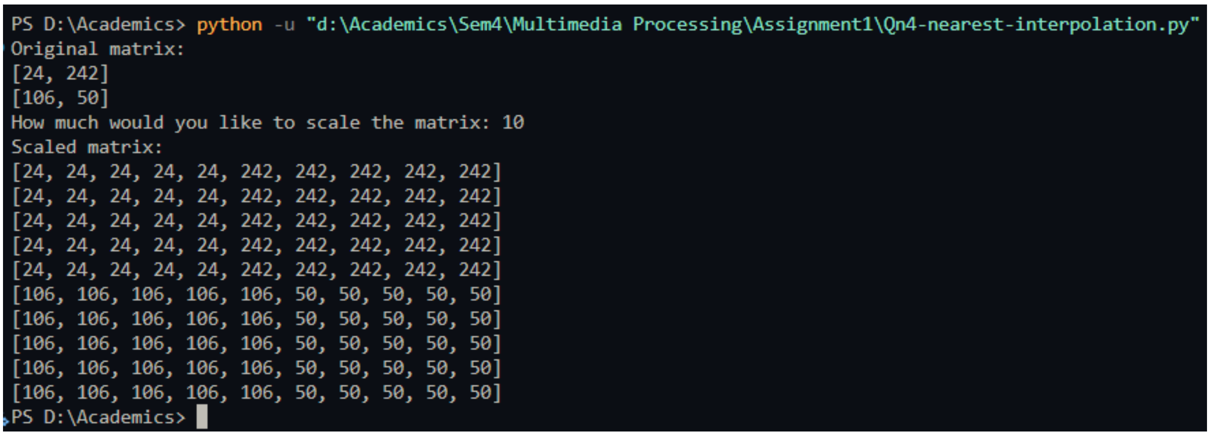
Qn6 - nearest interpolation

Code:



```python
#6. write a program for nearest interpolation
import random
l = []
for i in range(2):
    m = []
    for j in range(2):
        m.append(random.randint(0, 255))
    l.append(m)

print("Original matrix:")
for row in l:
    print(row)

sc = int(input("How much would you like to scale the matrix: "))

l1 = []
for i in range(sc):
    m1 = []
    for j in range(sc):
        nearest_i = round(i * (1 / (sc - 1)))
        nearest_j = round(j * (1 / (sc - 1)))
        m1.append(l[nearest_i][nearest_j])
    l1.append(m1)
print("Scaled matrix:")
for row in l1:
    print(row)
```

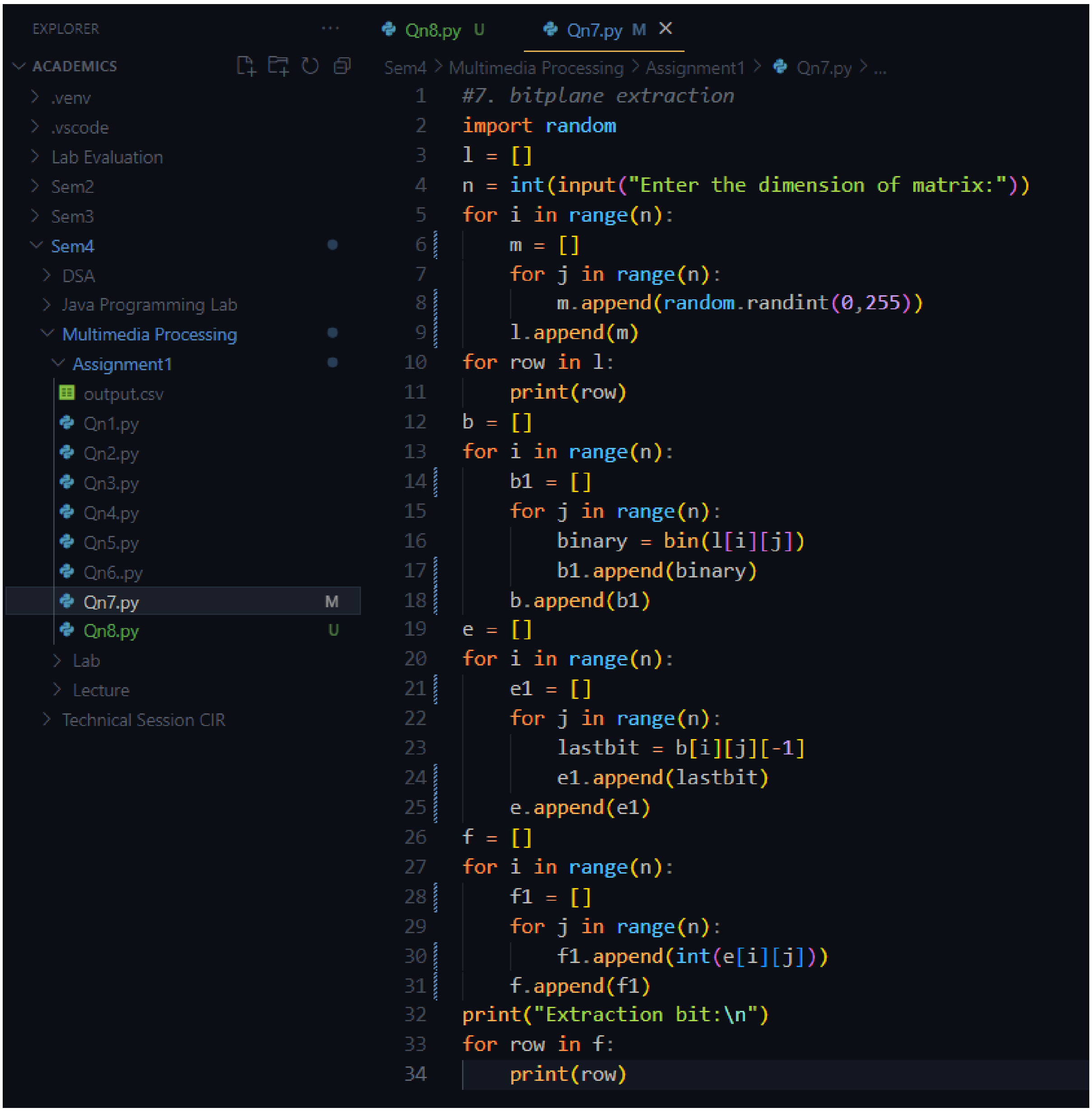
Output:



```
PS D:\Academics> python -u "d:\Academics\Sem4\Multimedia Processing\Assignment1\Qn4-nearest-interpolation.py"
Original matrix:
[24, 242]
[106, 50]
How much would you like to scale the matrix: 10
Scaled matrix:
[24, 24, 24, 24, 24, 242, 242, 242, 242, 242]
[24, 24, 24, 24, 24, 242, 242, 242, 242, 242]
[24, 24, 24, 24, 24, 242, 242, 242, 242, 242]
[24, 24, 24, 24, 24, 242, 242, 242, 242, 242]
[24, 24, 24, 24, 24, 242, 242, 242, 242, 242]
[106, 106, 106, 106, 106, 50, 50, 50, 50, 50]
[106, 106, 106, 106, 106, 50, 50, 50, 50, 50]
[106, 106, 106, 106, 106, 50, 50, 50, 50, 50]
[106, 106, 106, 106, 106, 50, 50, 50, 50, 50]
[106, 106, 106, 106, 106, 50, 50, 50, 50, 50]
PS D:\Academics>
```

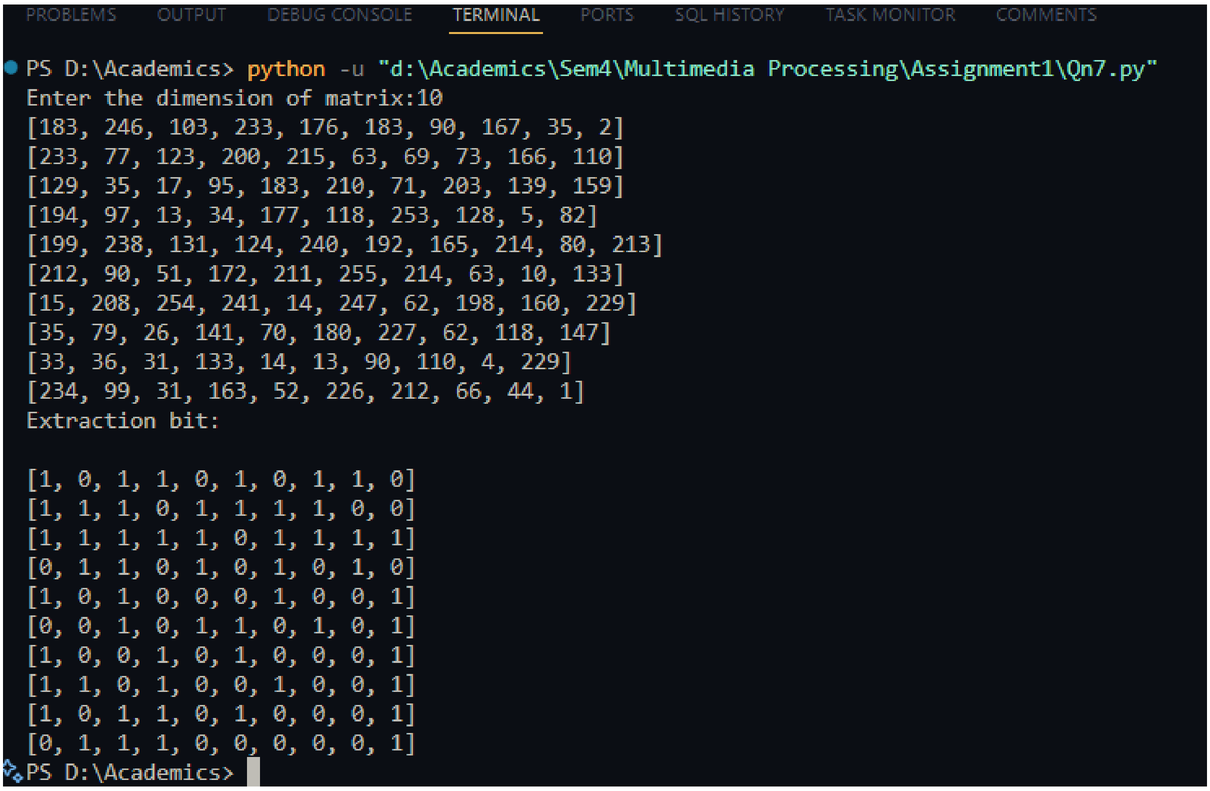
Qn7 – Bitplane extraction

Code:



```python
#7. bitplane extraction
import random
l = []
n = int(input("Enter the dimension of matrix:"))
for i in range(n):
    m = []
    for j in range(n):
        m.append(random.randint(0,255))
    l.append(m)
for row in l:
    print(row)
b = []
for i in range(n):
    b1 = []
    for j in range(n):
        binary = bin(l[i][j])
        b1.append(binary)
    b.append(b1)
e = []
for i in range(n):
    e1 = []
    for j in range(n):
        lastbit = b[i][j][-1]
        e1.append(lastbit)
    e.append(e1)
f = []
for i in range(n):
    f1 = []
    for j in range(n):
        f1.append(int(e[i][j]))
    f.append(f1)
print("Extraction bit:\n")
for row in f:
    print(row)
```

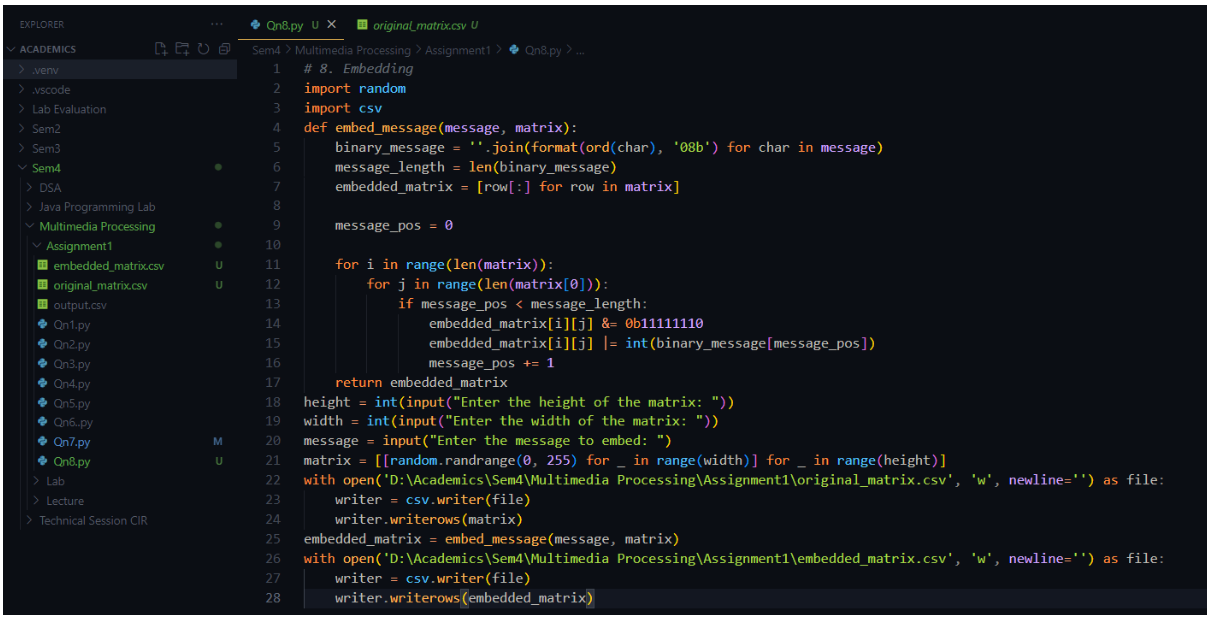
Output:



```
PS D:\Academics> python -u "d:\Academics\Sem4\Multimedia Processing\Assignment1\Qn7.py"
Enter the dimension of matrix:10
[183, 246, 103, 233, 176, 183, 90, 167, 35, 2]
[233, 77, 123, 200, 215, 63, 69, 73, 166, 110]
[129, 35, 17, 95, 183, 210, 71, 203, 139, 159]
[194, 97, 13, 34, 177, 118, 253, 128, 5, 82]
[199, 238, 131, 124, 240, 192, 165, 214, 80, 213]
[212, 90, 51, 172, 211, 255, 214, 63, 10, 133]
[15, 208, 254, 241, 14, 247, 62, 198, 160, 229]
[35, 79, 26, 141, 70, 180, 227, 62, 118, 147]
[33, 36, 31, 133, 14, 13, 90, 110, 4, 229]
[234, 99, 31, 163, 52, 226, 212, 66, 44, 1]
Extraction bit:

[1, 0, 1, 1, 0, 1, 0, 1, 1, 0]
[1, 1, 1, 0, 1, 1, 1, 1, 0, 0]
[1, 1, 1, 1, 1, 0, 1, 1, 1, 1]
[0, 1, 1, 0, 1, 0, 1, 0, 1, 0]
[1, 0, 1, 0, 0, 0, 1, 0, 0, 1]
[0, 0, 1, 0, 1, 1, 0, 1, 0, 1]
[1, 0, 0, 1, 0, 1, 0, 0, 0, 1]
[1, 1, 0, 1, 0, 0, 1, 0, 0, 1]
[1, 0, 1, 1, 0, 1, 0, 0, 0, 1]
[0, 1, 1, 1, 0, 0, 0, 0, 0, 1]
PS D:\Academics>
```

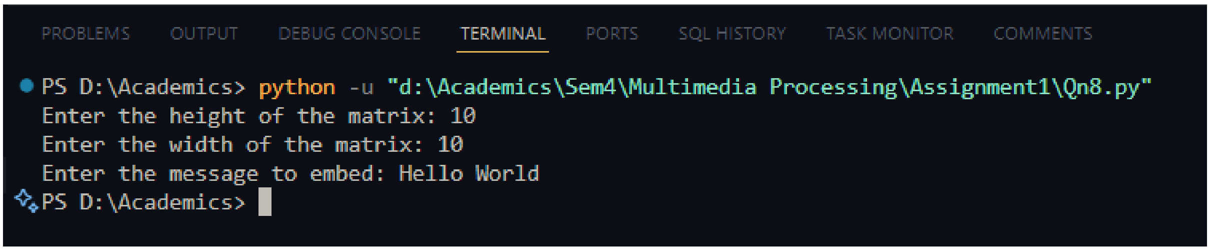
Qn 8 – Embedding

Code:



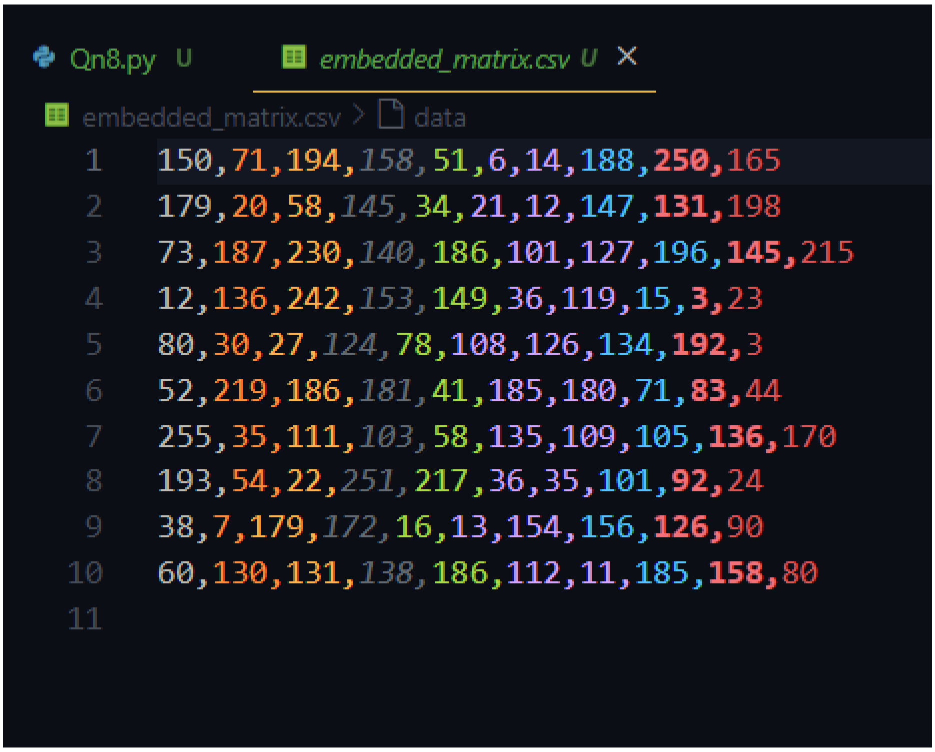
```python
# 8. Embedding
import random
import csv
def embed_message(message, matrix):
    binary_message = ''.join(format(ord(char), '08b') for char in message)
    message_length = len(binary_message)
    embedded_matrix = [row[:] for row in matrix]

    message_pos = 0

    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            if message_pos < message_length:
                embedded_matrix[i][j] &= 0b11111110
                embedded_matrix[i][j] |= int(binary_message[message_pos])
                message_pos += 1
    return embedded_matrix
height = int(input("Enter the height of the matrix: "))
width = int(input("Enter the width of the matrix: "))
message = input("Enter the message to embed: ")
matrix = [[random.randrange(0, 255) for _ in range(width)] for _ in range(height)]
with open('D:\Academics\Sem4\Multimedia Processing\Assignment1\original_matrix.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerows(matrix)
embedded_matrix = embed_message(message, matrix)
with open('D:\Academics\Sem4\Multimedia Processing\Assignment1\embedded_matrix.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerows(embedded_matrix)
```

Output:



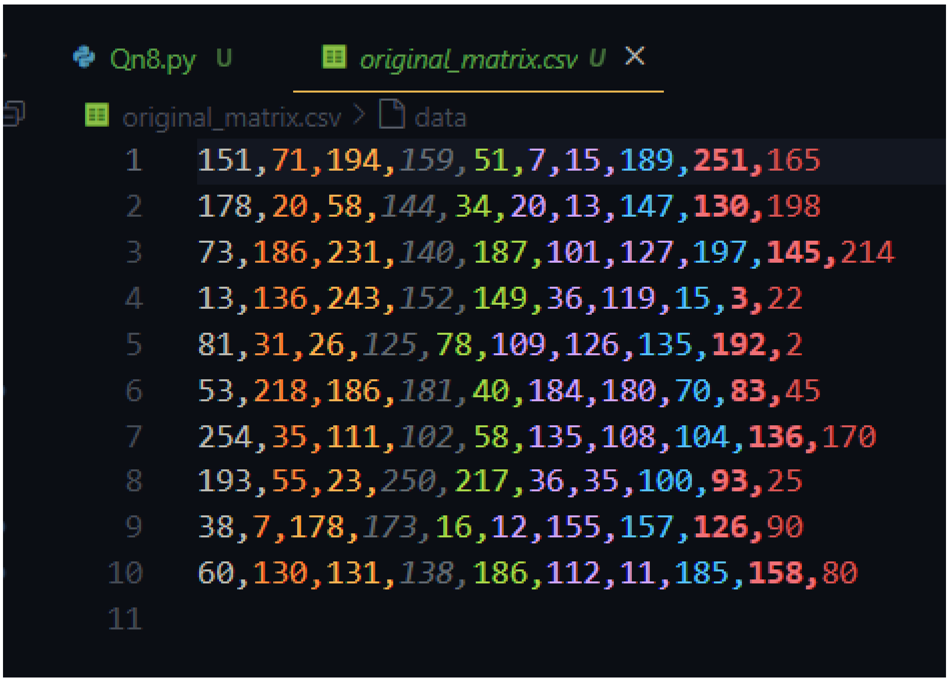
```
PS D:\Academics> python -u "d:\Academics\Sem4\Multimedia Processing\Assignment1\Qn8.py"
Enter the height of the matrix: 10
Enter the width of the matrix: 10
Enter the message to embed: Hello World
PS D:\Academics>
```

Embedded Matrix



```
150,71,194,158,51,6,14,188,250,165
179,20,58,145,34,21,12,147,131,198
73,187,230,140,186,101,127,196,145,215
12,136,242,153,149,36,119,15,3,23
80,30,27,124,78,108,126,134,192,3
52,219,186,181,41,185,180,71,83,44
255,35,111,103,58,135,109,105,136,170
193,54,22,251,217,36,35,101,92,24
38,7,179,172,16,13,154,156,126,90
60,130,131,138,186,112,11,185,158,80
```

Original Matrix:



```
151,71,194,159,51,7,15,189,251,165
178,20,58,144,34,20,13,147,130,198
73,186,231,140,187,101,127,197,145,214
13,136,243,152,149,36,119,15,3,22
81,31,26,125,78,109,126,135,192,2
53,218,186,181,40,184,180,70,83,45
254,35,111,102,58,135,108,104,136,170
193,55,23,250,217,36,35,100,93,25
38,7,178,173,16,12,155,157,126,90
60,130,131,138,186,112,11,185,158,80
```