Following is the python program to perform the data encoding and decoding of the 2D matrix.

1. For this program, the 2D matrix is predefined in 9 rows and 9 columns as follows:

```
matrix = np.array(
    [[0, 3, 2, 5, 4, 7, 6, 9, 8],

     [3, 0, 1, 2, 3, 4, 5, 6, 7],

     [2, 1, 0, 3, 2, 5, 4, 7, 6],

     [5, 2, 3, 0, 1, 2, 3, 4, 5],

     [4, 3, 2, 1, 0, 3, 2, 5, 4],

     [7, 4, 5, 2, 3, 0, 1, 2, 3],

     [6, 5, 4, 3, 2, 1, 0, 3, 2],

     [9, 6, 7, 4, 5, 2, 3, 0, 1],

     [8, 7, 6, 5, 4, 3, 2, 1, 0]])
```

2. For each element, the program will get its 3 columns and 3 rows of its neighbors. Then subtract that element of middle index from its 8 neighbors.

```
                matrixOfThree = [[matrix[i - 1][j - 1] - matrix[i][j],
                                  matrix[i - 1][j] - matrix[i][j],
                                  matrix[i - 1][j + 1] - matrix[i][j]],
                                 [matrix[i][j - 1] - matrix[i][j],
                                  matrix[i][j], matrix[i][j + 1] - matrix
  [i][j]],
                                 [matrix[i + 1][j - 1] - matrix[i][j],
                                  matrix[i + 1][j] - matrix[i][j],
                                  matrix[i + 1][j + 1] - matrix[i][j]]]
```

3. From the results it will then sign “1” to the values that are greater than or equal “0” and “0” for the values that are less than “0”.

```
                for k in range(0, 3):
                    for l in range(0, 3):
                        if k != 1 or l != 1:
                            if matrixOfThree[k][l] >= 0:
                                matrixOfThree[k][l] = 1
                            else:
                                matrixOfThree[k][l] = 0
```

4. Concatenate the 8-bit binary code as counterclockwise starting from the middle element which will be at the first bit from the right to bottom right element.

```
37                  # binary string
38                  binary_string = str(matrixOfThree[0][0]) \
39                                  + str(matrixOfThree[1][0]) \
40                                  + str(matrixOfThree[2][0]) \
41                                  + str(matrixOfThree[2][1]) \
42                                  + str(matrixOfThree[2][2]) \
43                                  + str(matrixOfThree[1][2]) \
44                                  + str(matrixOfThree[0][2]) \
45                                  + str(matrixOfThree[0][1])
46
47                  # getting the integer value
48                  integer_val = int(binary_string, 2)
```

5. Convert the binary code to decimal and substitute that decimal code instead of the element in the middle.

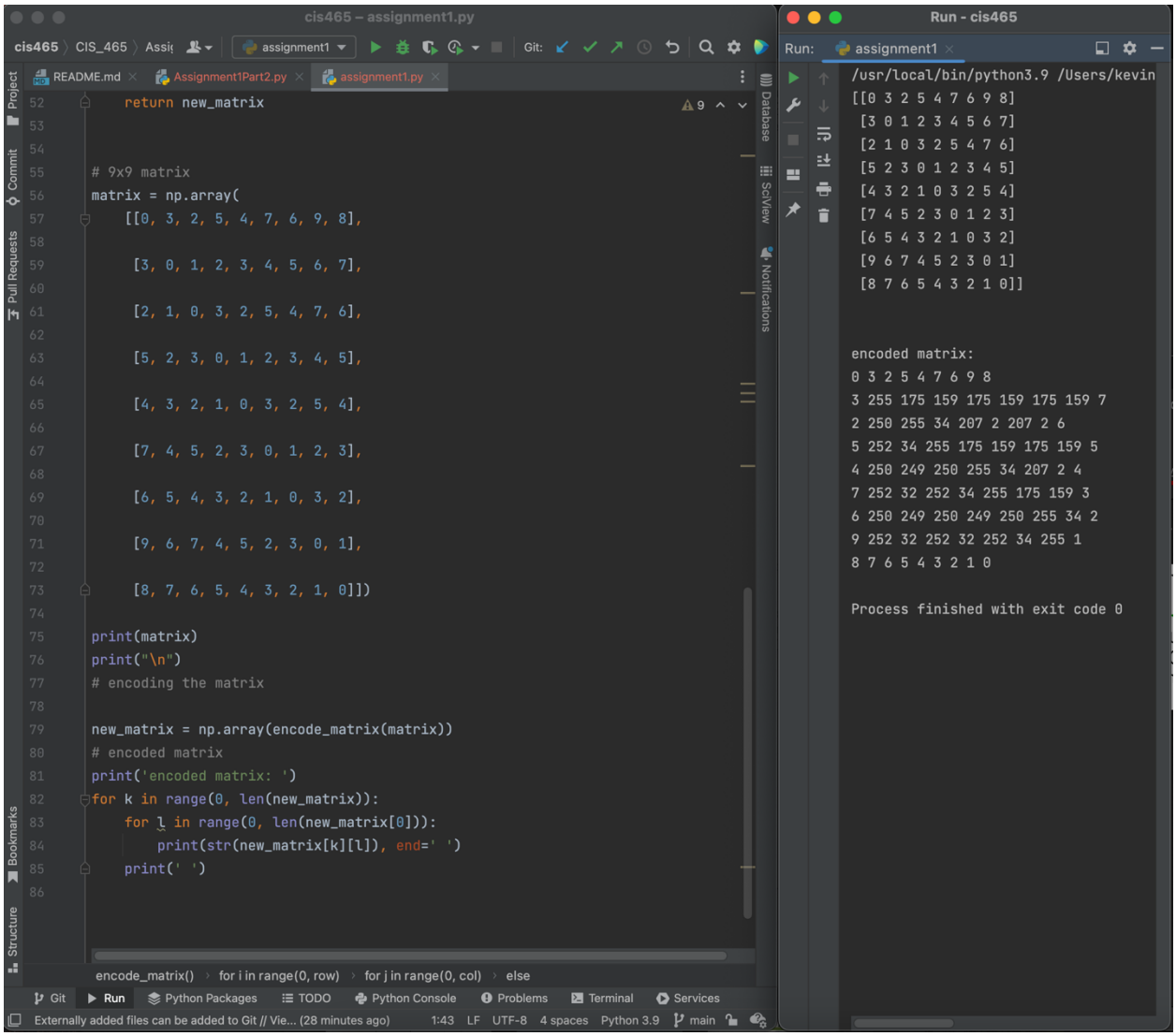
```
47                  # getting the integer value
48                  integer_val = int(binary_string, 2)
```

```
50                  new_array.insert(len(new_array), integer_val)
51              new_matrix.insert(len(new_matrix), new_array)
52          return new_matrix
53
54
```

6. The program will repeat for all the elements and display the encoded and decoded matrices.

```
79 new_matrix = np.array(encode_matrix(matrix))
80 # encoded matrix
81 print('encoded matrix: ')
82 for k in range(0, len(new_matrix)):
83     for l in range(0, len(new_matrix[0])):
84         print(str(new_matrix[k][l]), end=' ')
85     print(' ')
```

7. Output of the program:




```
    return new_matrix


# 9x9 matrix
matrix = np.array(
    [[0, 3, 2, 5, 4, 7, 6, 9, 8],

     [3, 0, 1, 2, 3, 4, 5, 6, 7],

     [2, 1, 0, 3, 2, 5, 4, 7, 6],

     [5, 2, 3, 0, 1, 2, 3, 4, 5],

     [4, 3, 2, 1, 0, 3, 2, 5, 4],

     [7, 4, 5, 2, 3, 0, 1, 2, 3],

     [6, 5, 4, 3, 2, 1, 0, 3, 2],

     [9, 6, 7, 4, 5, 2, 3, 0, 1],

     [8, 7, 6, 5, 4, 3, 2, 1, 0]])

print(matrix)
print("\n")
# encoding the matrix

new_matrix = np.array(encode_matrix(matrix))
# encoded matrix
print('encoded matrix: ')
for k in range(0, len(new_matrix)):
    for l in range(0, len(new_matrix[0])):
        print(str(new_matrix[k][l]), end=' ')
    print(' ')
```