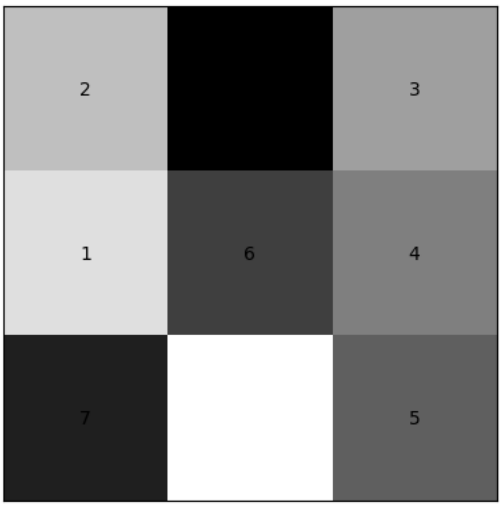
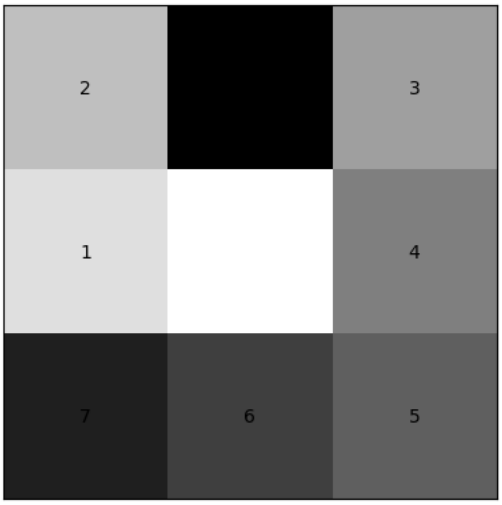
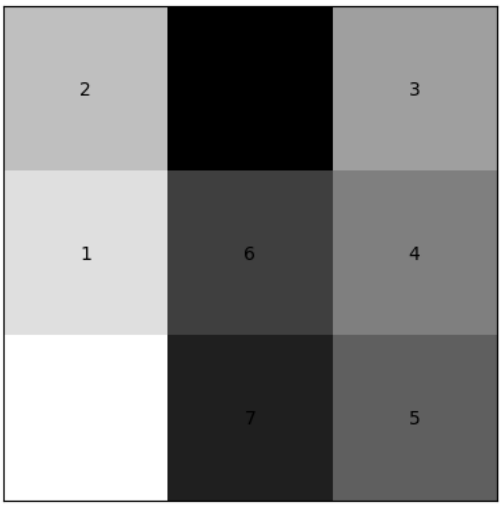
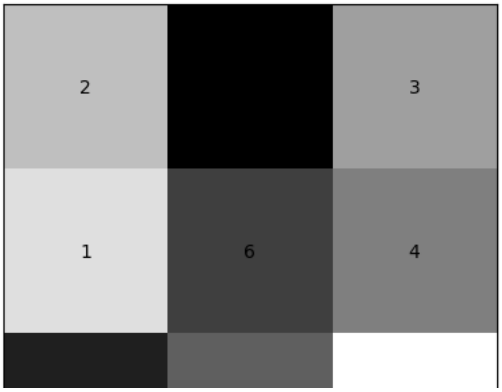
```
import numpy as np
import matplotlib.pyplot as plt
from collections import deque

class Puzzle:
    def __init__(self, inicio, objetivo):
        self.inicio = inicio
        self.objetivo = objetivo
        self.direcciones = ['arriba', 'abajo', 'izquierda', 'derecha']

    def mover(self, estado, direccion):
        estado = estado.copy()
        i, j = np.where(estado == 0)
        if direccion == 'arriba' and i > 0:
            estado[i, j], estado[i - 1, j] = estado[i - 1, j], estado[i, j]
        elif direccion == 'abajo' and i < 2:
            estado[i, j], estado[i + 1, j] = estado[i + 1, j], estado[i, j]
        elif direccion == 'izquierda' and j > 0:
            estado[i, j], estado[i, j - 1] = estado[i, j - 1], estado[i, j]
        elif direccion == 'derecha' and j < 2:
            estado[i, j], estado[i, j + 1] = estado[i, j + 1], estado[i, j]
        else:
            return None
        return estado

    def dibujar(self, estado):
        plt.imshow(estado, cmap='gray_r')
        plt.xticks([])
        plt.yticks([])
        for (j, i), label in np.ndenumerate(estado):
            if label > 0:
                plt.text(i, j, int(label), ha='center', va='center', color='black')
        plt.show(block=False)
        plt.pause(0.1)
        plt.clf()

    def resolver(self):
        cola = deque([self.inicio])
        visitados = set([self.inicio.tobytes()])
        caminos = {self.inicio.tobytes(): []}

        while cola:
            estado = cola.popleft()
            self.dibujar(estado)
            if np.array_equal(estado, self.objetivo):
                return caminos[estado.tobytes()]
            for direccion in self.direcciones:
                nuevo_estado = self.mover(estado, direccion)
                if nuevo_estado is not None and nuevo_estado.tobytes() not in visitados:
                    cola.append(nuevo_estado)
                    visitados.add(nuevo_estado.tobytes())
                    caminos[nuevo_estado.tobytes()] = caminos[estado.tobytes()] + [direccion]
        return None

inicio = np.array([[2,8,3], [1,6,4], [7,0,5]])
objetivo = np.array([[1,2,3], [8,0,4], [7,6,5]])

puzzle = Puzzle(inicio, objetivo)
solucion = puzzle.resolver()

if solucion is not None:
    print('Solución encontrada:')
    for paso in solucion:
        print(paso)
else:
    print('No se encontró una solución')
```

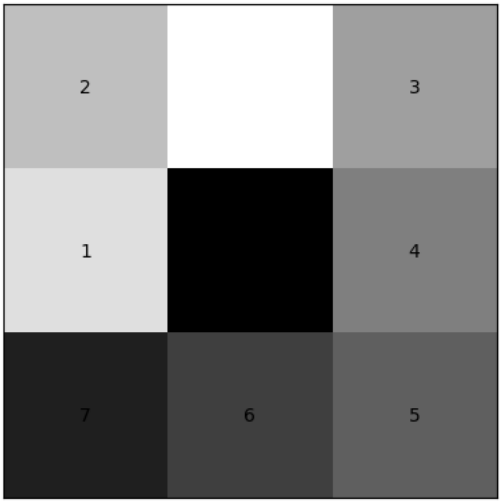

2
3
1
6
4
7
5

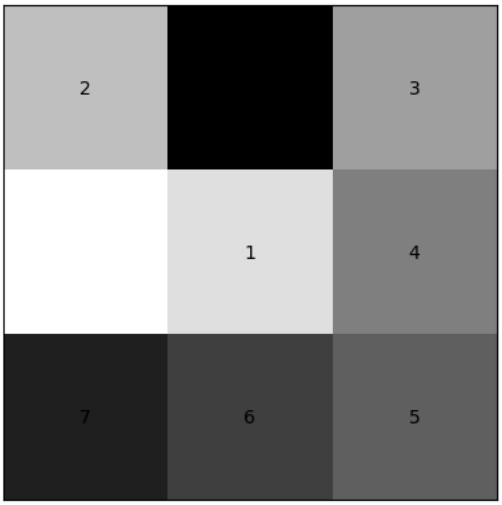

2
3
1
4
7
6
5

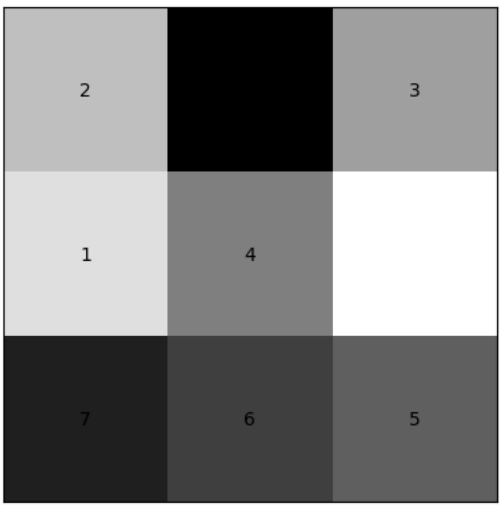

2
3
1
6
4
7
5

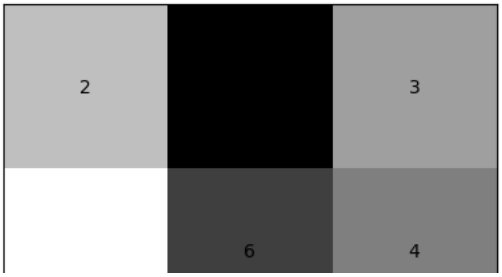

2
3
1
6
4

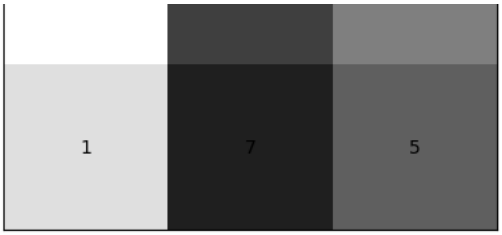

7
5

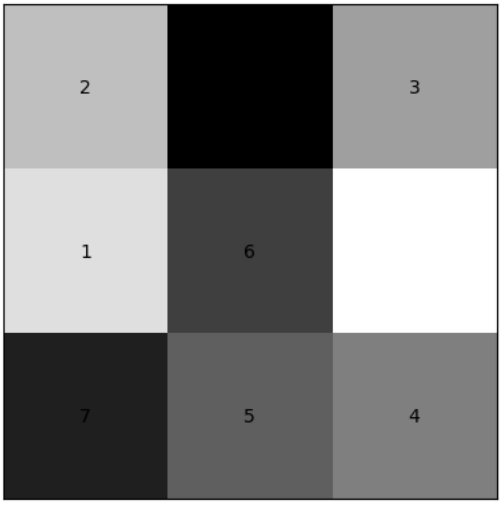

2
3
1
4
7
6
5

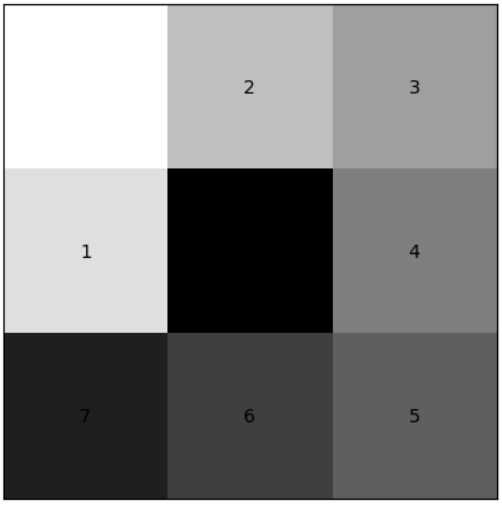

2
3
1
4
7
6
5

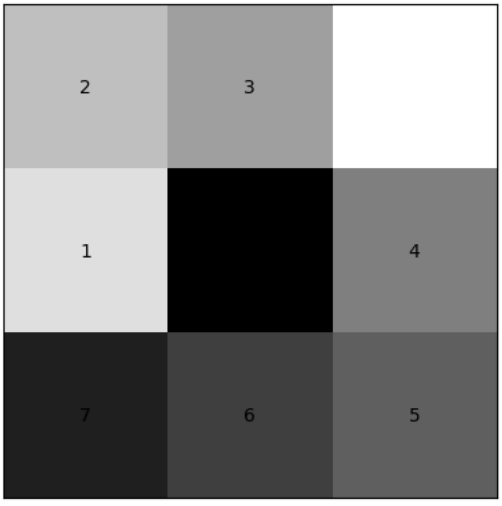

2
3
1
4
7
6
5


2
3
6
4

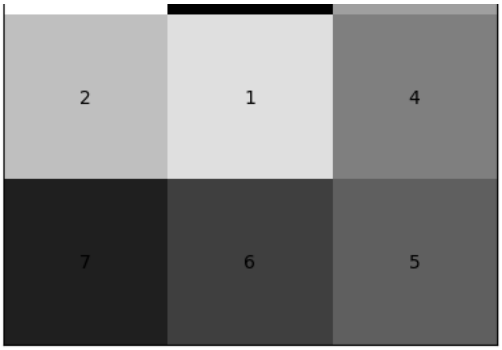

1
7
5

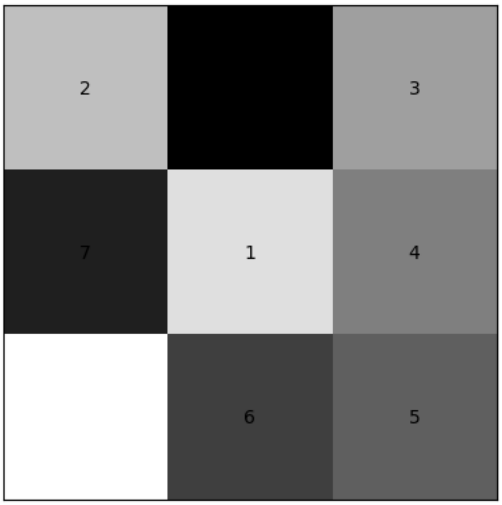

2
3
1
6
7
5
4

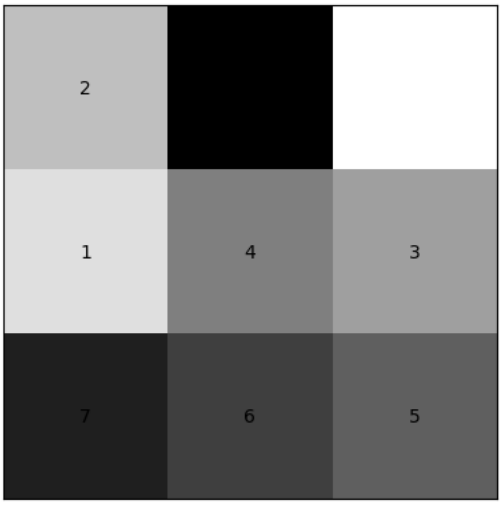

2
3
1
4
7
6
5

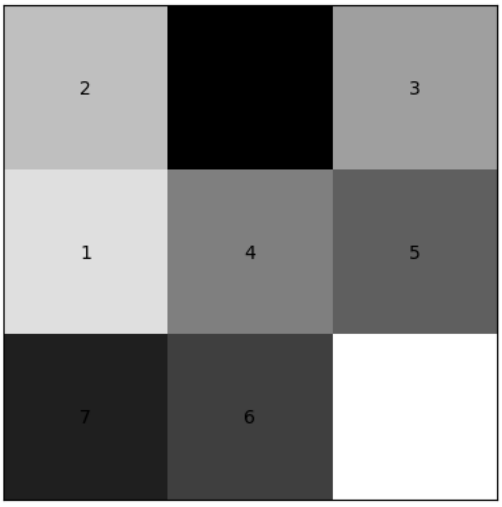

2
3
1
4
7
6
5

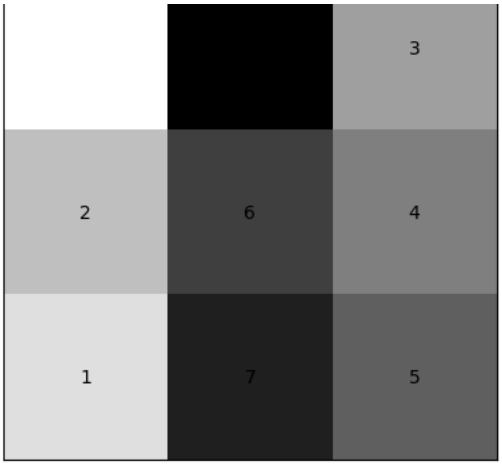

3

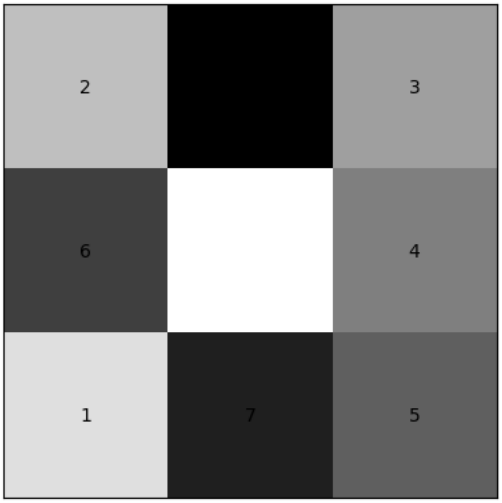

2
1
4
7
6
5

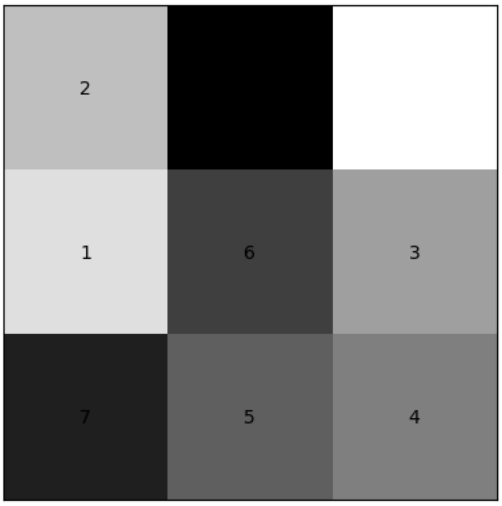

2
3
7
1
4
6
5

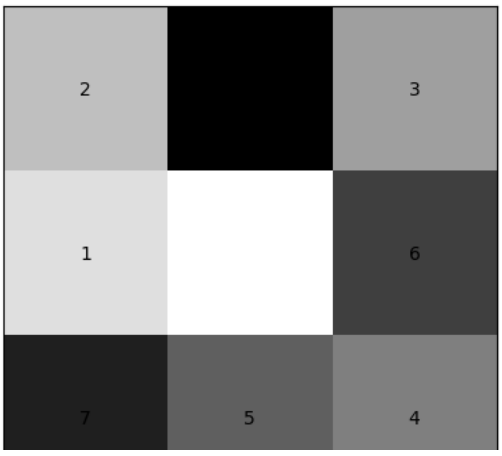

2
1
4
3
7
6
5

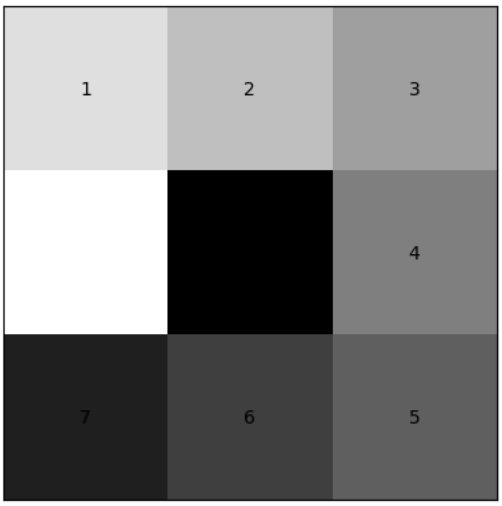

2
3
1
4
5
7
6

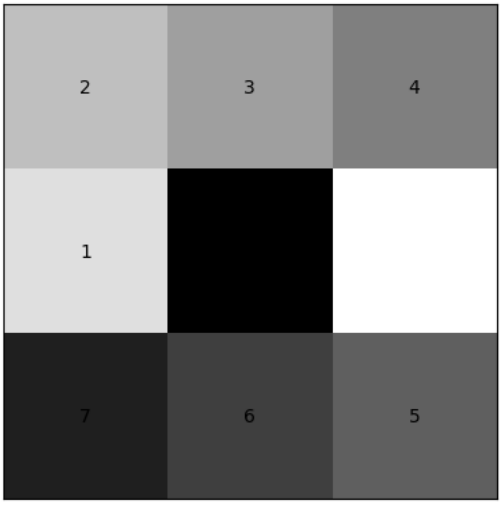

3
2 6 4
1 7 5

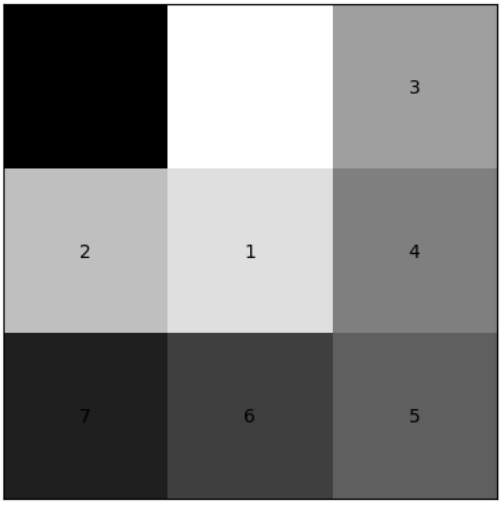

2 3
6 4
1 7 5

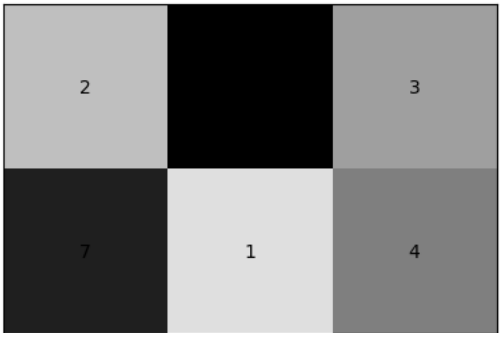

2
1 6 3
7 5 4


2 3
1 6
7 5 4

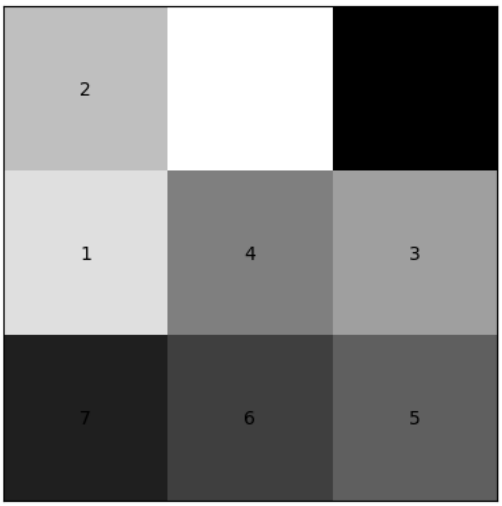

1
2
3
4
7
6
5

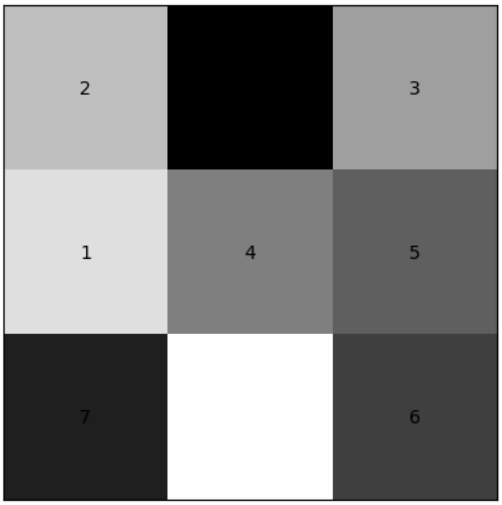

2
3
4
1
7
6
5

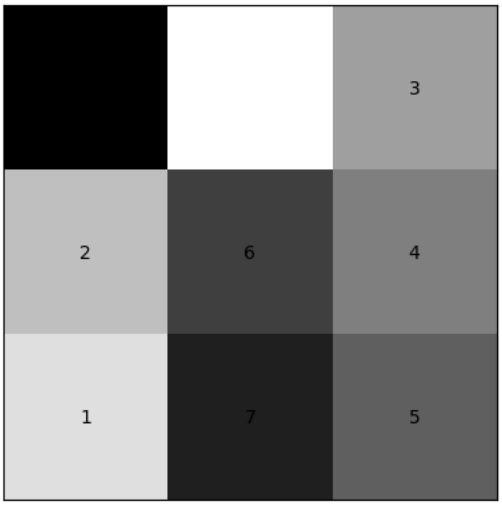

3
2
1
4
7
6
5

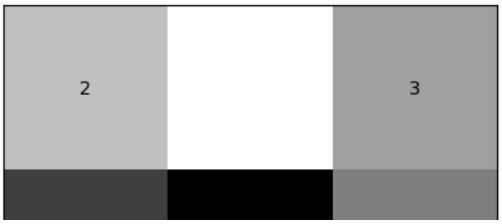

2
3
7
1
4


6
5

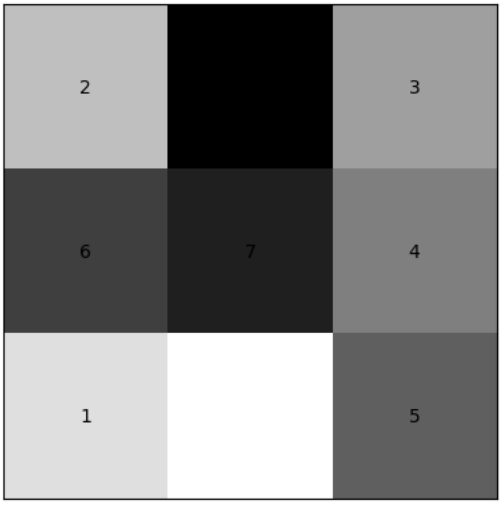

2
1
4
3
7
6
5

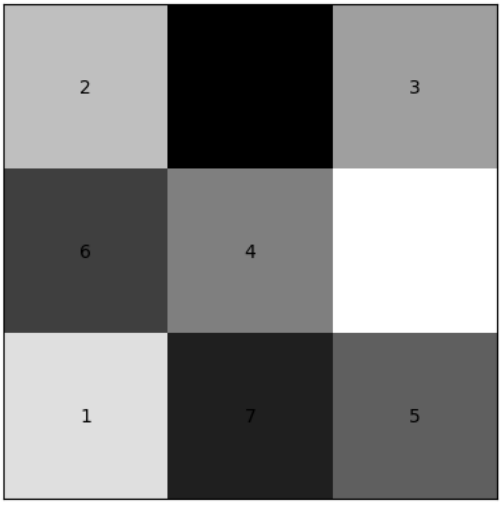

2
3
1
4
5
7
6

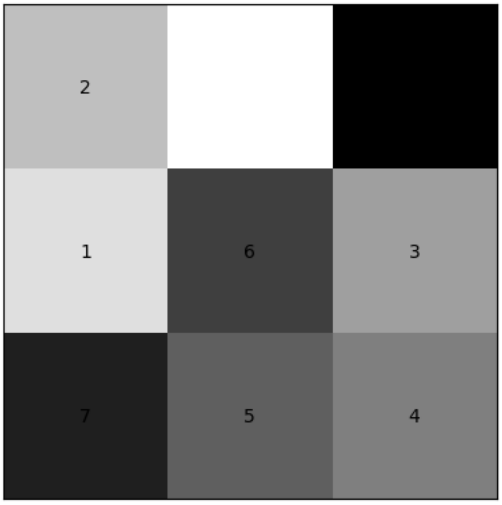

3
2
6
4
1
7
5


2
3

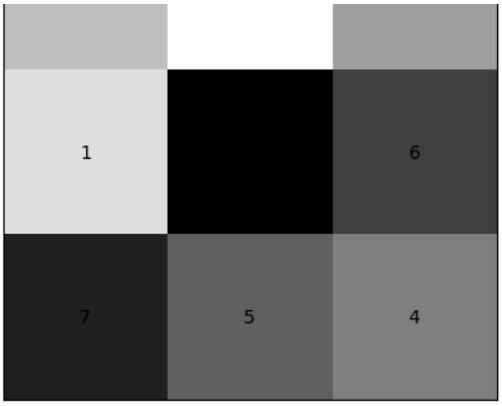

6
4
1
7
5

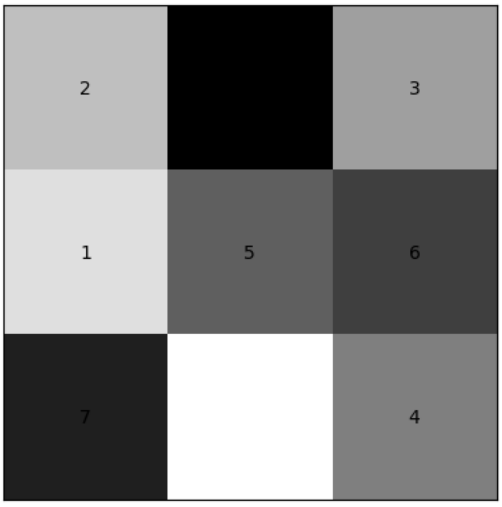

2
3
6
7
4
1
5

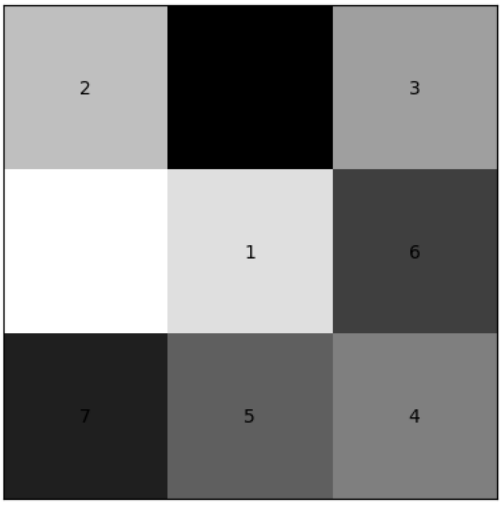

2
3
6
4
1
7
5

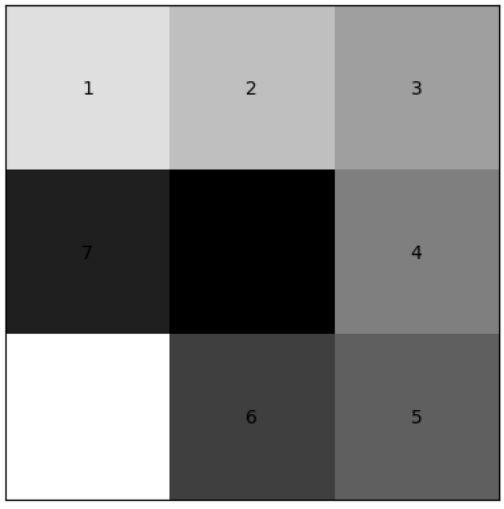

2
1
6
3
7
5
4

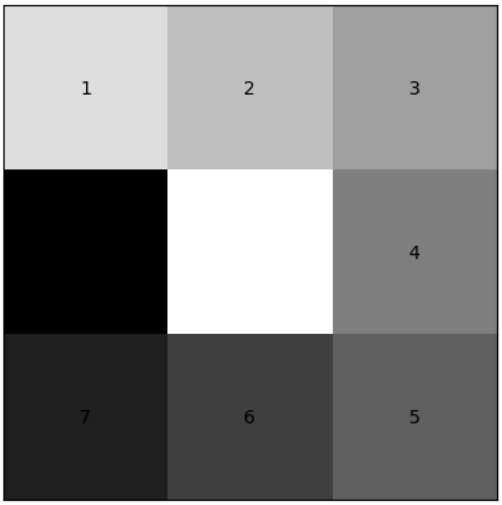

2
3


1
6
7
5
4


2
3
1
5
6
7
4


2
3
1
6
7
5
4


1
2
3
7
4
6
5




```
Solución encontrada:
arriba
arriba
izquierda
abajo
derecha
<Figure size 640x480 with 0 Axes>
```