**1.Write the python program to solve 8-Puzzle problem**

**Program:**

import heapq

goal = ((1,2,3),(4,5,6),(7,8,0))

def manhattan(state):

d = 0

for i in range(3):

for j in range(3):

v = state[i][j]

if v != 0:

x, y = divmod(v - 1, 3)

d += abs(x - i) + abs(y - j)

return d

def get\_neighbors(state):

neighbors = []

x, y = [(i, j) for i in range(3) for j in range(3) if state[i][j] == 0][0]

for dx, dy in [(-1,0),(1,0),(0,-1),(0,1)]:

nx, ny = x + dx, y + dy

if 0 <= nx < 3 and 0 <= ny < 3:

new\_state = [list(row) for row in state]

new\_state[x][y], new\_state[nx][ny] = new\_state[nx][ny], new\_state[x][y]

neighbors.append(tuple(tuple(row) for row in new\_state))

return neighbors

def solve(start):

heap = [(manhattan(start), 0, start, [])]

seen = set()

while heap:

\_, cost, state, path = heapq.heappop(heap)

if state in seen:

continue

seen.add(state)

if state == goal:

return path + [state]

for neighbor in get\_neighbors(state):

heapq.heappush(heap, (cost + 1 + manhattan(neighbor), cost + 1, neighbor, path + [state]))

# Example start state

start = ((1,2,3),(4,0,6),(7,5,8))

result = solve(start)

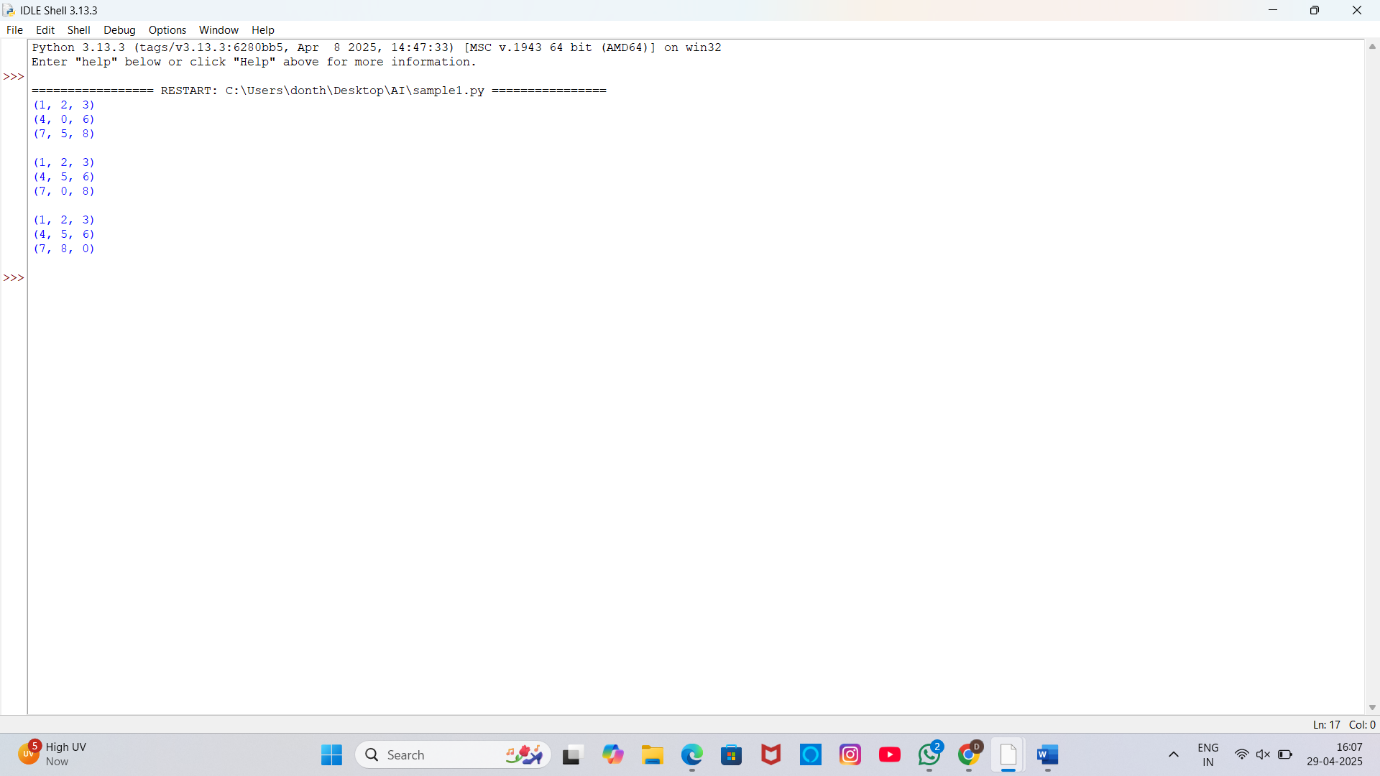
for step in result:

for row in step:

print(row)

print()

**output:**

****