**PROGRAM 1**

import heapq

class PuzzleState:

def \_\_init\_\_(self, board, empty\_pos, moves, cost):

self.board = board

self.empty\_pos = empty\_pos

self.moves = moves

self.cost = cost

def \_\_lt\_\_(self, other):

return self.cost < other.cost

def get\_neighbors(state):

neighbors = []

x, y = state.empty\_pos

directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]

for dx, dy in directions:

nx, ny = x + dx, y + dy

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

new\_board = [row[:] for row in state.board]

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

neighbors.append(PuzzleState(new\_board, (nx, ny), state.moves + [(nx, ny)], 0))

return neighbors

def heuristic(state, goal):

cost = 0

for i in range(3):

for j in range(3):

val = state.board[i][j]

if val != 0:

gx, gy = goal[val]

cost += abs(gx - i) + abs(gy - j)

return cost

def solve\_puzzle(start\_board, goal\_board):

goal\_positions = {val: (i, j) for i, row in enumerate(goal\_board) for j, val in enumerate(row)}

empty\_pos = next((i, j) for i, row in enumerate(start\_board) for j, val in enumerate(row) if val == 0)

start\_state = PuzzleState(start\_board, empty\_pos, [], heuristic(PuzzleState(start\_board, empty\_pos, [], 0), goal\_positions))

open\_list = []

heapq.heappush(open\_list, start\_state)

visited = set()

while open\_list:

current\_state = heapq.heappop(open\_list)

if tuple(map(tuple, current\_state.board)) in visited:

continue

visited.add(tuple(map(tuple, current\_state.board)))

if current\_state.board == goal\_board:

return current\_state.moves

for neighbor in get\_neighbors(current\_state):

neighbor.cost = len(neighbor.moves) + heuristic(neighbor, goal\_positions)

heapq.heappush(open\_list, neighbor)

return None

# Example usage:

start\_board = [

[1, 2, 3],

[4, 0, 5],

[6, 7, 8]

]

goal\_board = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 0]

]

solution = solve\_puzzle(start\_board, goal\_board)

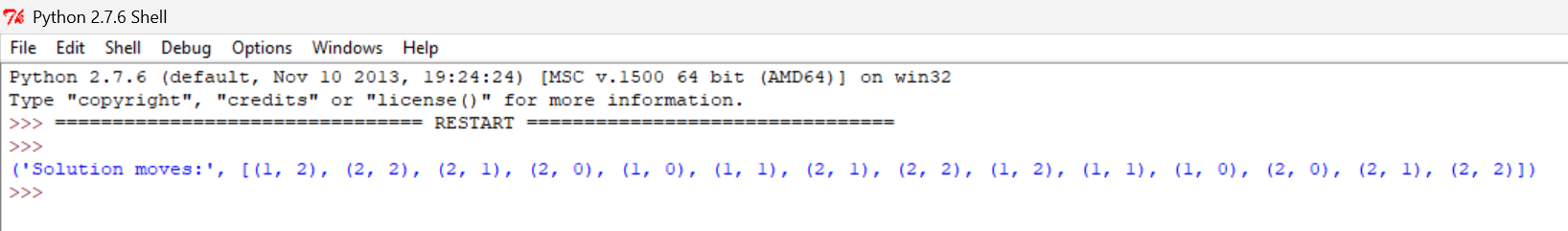
if solution:

print("Solution moves:", solution)

else:

print("No solution found.")

**OUTPUT:**

****