import heapq

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

def heuristic(state):

return sum(abs((val - 1) % 3 - i % 3) + abs((val - 1) // 3 - i // 3)

for i, val in enumerate(state) if val)

def neighbors(state):

i = state.index(0)

moves = []

for d in (-1, 1, -3, 3):

j = i + d

if 0 <= j < 9 and (i % 3 == j % 3 or i // 3 == j // 3):

lst = list(state)

lst[i], lst[j] = lst[j], lst[i]

moves.append(tuple(lst))

return moves

def solve(start):

frontier = [(heuristic(start), 0, start, [])]

visited = set()

while frontier:

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

if state == goal:

return path + [state]

if state in visited:

continue

visited.add(state)

for neighbor in neighbors(state):

heapq.heappush(frontier, (cost + 1 + heuristic(neighbor),

cost + 1, neighbor, path + [state]))

return None

# Example usage:

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

solution = solve(start\_state)

if solution:

for step in solution:

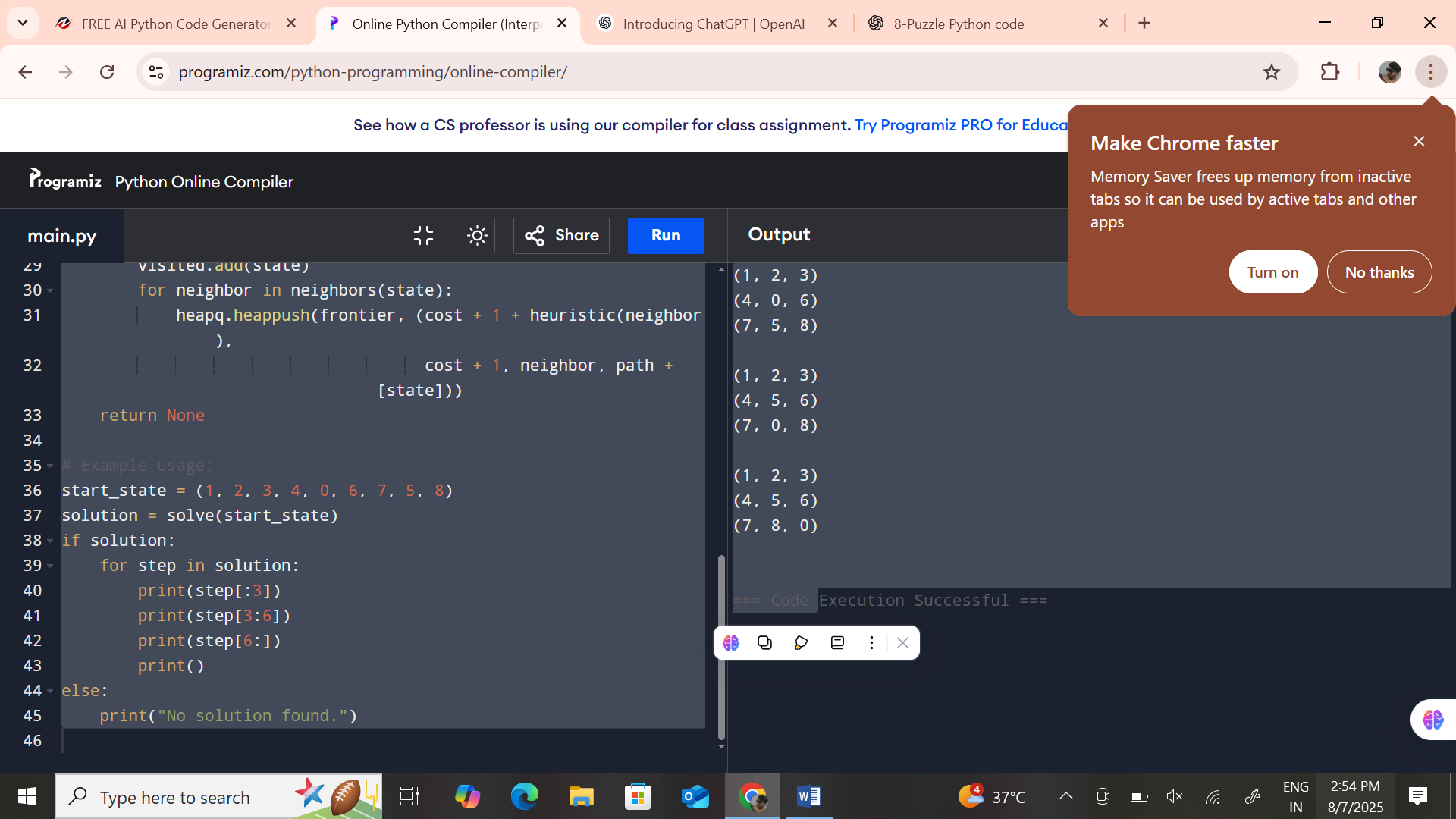
print(step[:3])

print(step[3:6])

print(step[6:])

print()

else:

 print("No solution found.")