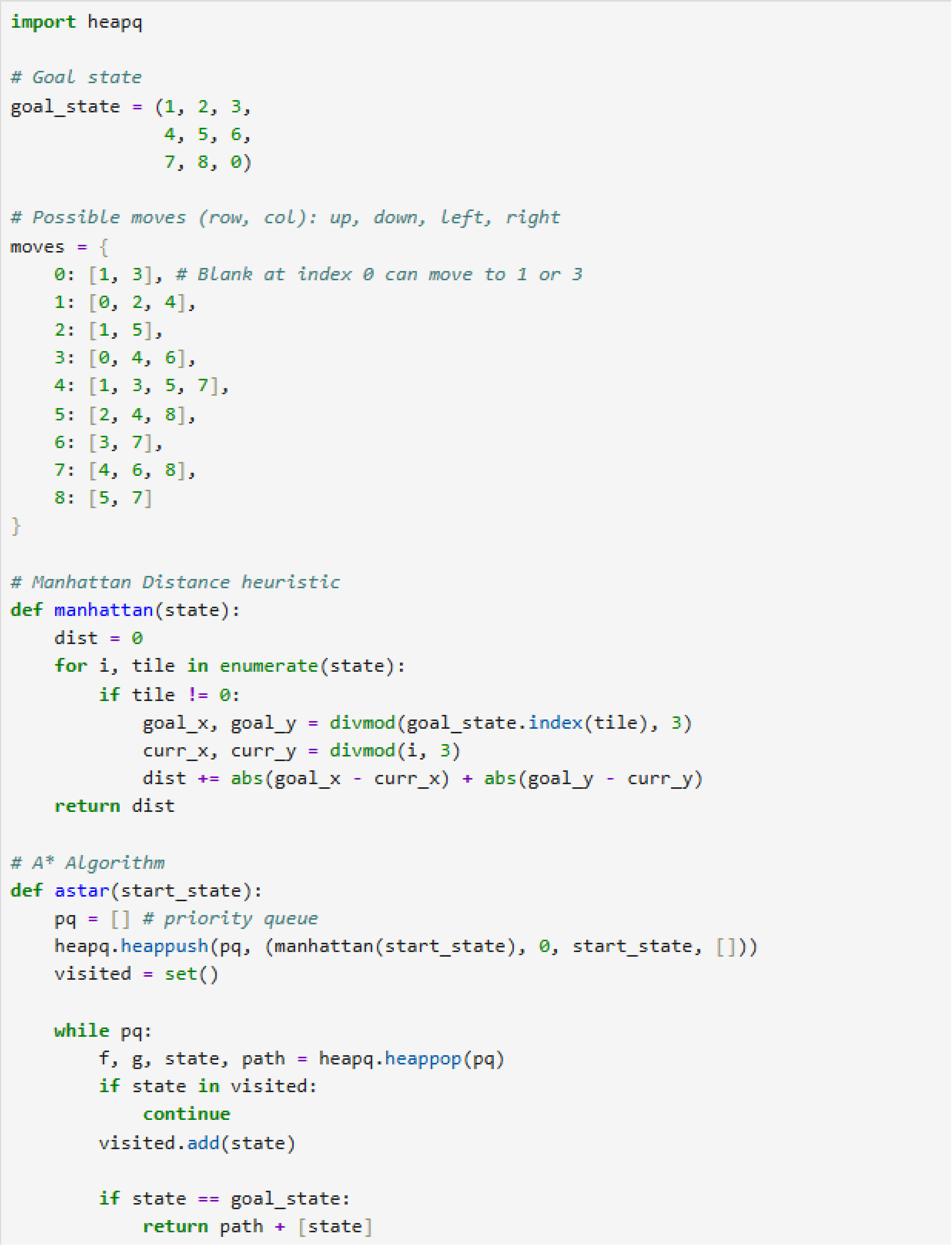
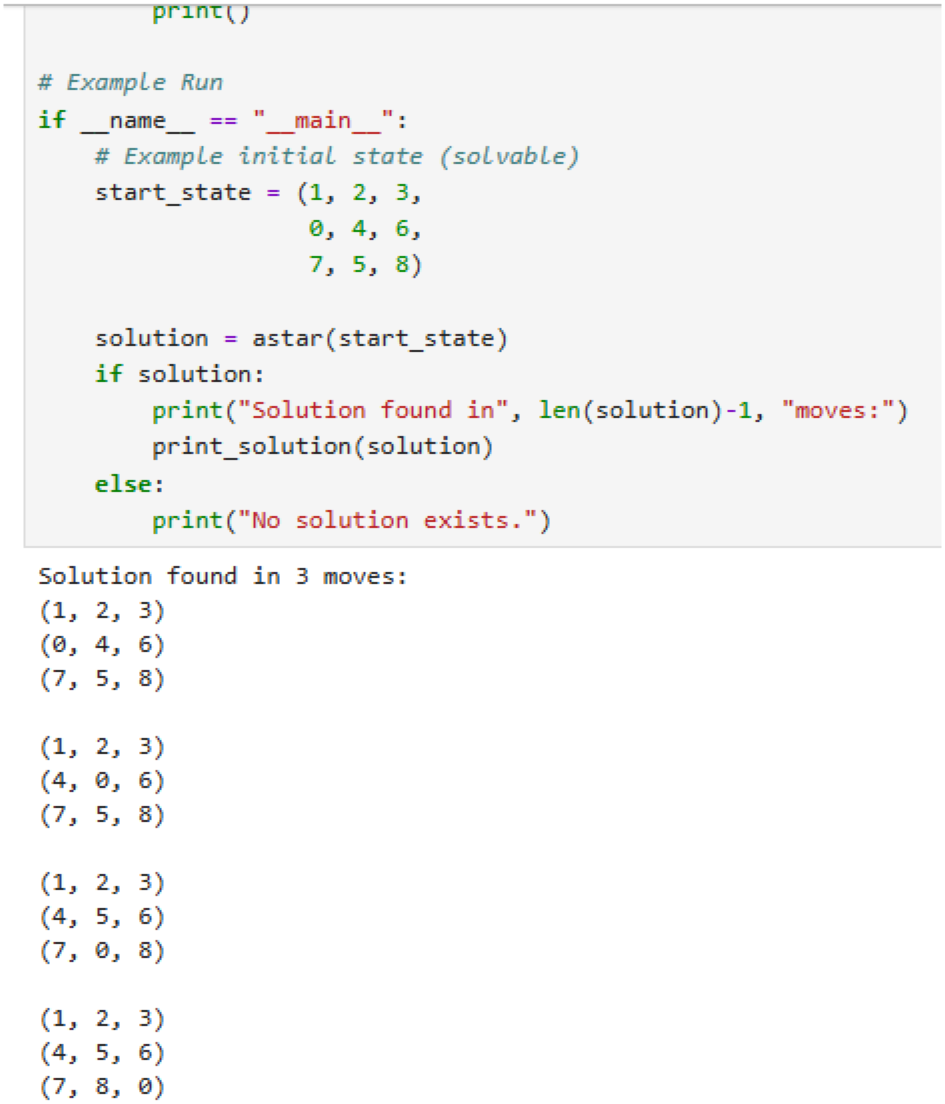
LAB 3
8 Puzzle - A*



```
import heapq

# Goal state
goal_state = (1, 2, 3,
              4, 5, 6,
              7, 8, 0)

# Possible moves (row, col): up, down, left, right
moves = {
    0: [1, 3], # Blank at index 0 can move to 1 or 3
    1: [0, 2, 4],
    2: [1, 5],
    3: [0, 4, 6],
    4: [1, 3, 5, 7],
    5: [2, 4, 8],
    6: [3, 7],
    7: [4, 6, 8],
    8: [5, 7]
}

# Manhattan Distance heuristic
def manhattan(state):
    dist = 0
    for i, tile in enumerate(state):
        if tile != 0:
            goal_x, goal_y = divmod(goal_state.index(tile), 3)
            curr_x, curr_y = divmod(i, 3)
            dist += abs(goal_x - curr_x) + abs(goal_y - curr_y)
    return dist

# A* Algorithm
def astar(start_state):
    pq = [] # priority queue
    heapq.heappush(pq, (manhattan(start_state), 0, start_state, []))
    visited = set()

    while pq:
        f, g, state, path = heapq.heappop(pq)
        if state in visited:
            continue
        visited.add(state)

        if state == goal_state:
            return path + [state]
        f, g, state, path = heapq.heappop(pq)
        if state in visited:
            continue
        visited.add(state)

        if state == goal_state:
            return path + [state]

        blank = state.index(0)
        for move in moves[blank]:
            new_state = list(state)
            new_state[blank], new_state[move] = new_state[move], new_state[blank]
            new_state = tuple(new_state)
            if new_state not in visited:
                heapq.heappush(pq, (g + 1 + manhattan(new_state), g + 1, new_state, path + [state]))
    return None

# Print solution
def print_solution(path):
    for state in path:
        for i in range(0, 9, 3):
            print(state[i:i+3])
        print()

# Example Run
if __name__ == "__main__":
    # Example initial state (solvable)
    start_state = (1, 2, 3,
                   0, 4, 6,
                   7, 5, 8)

    solution = astar(start_state)
    if solution:
        print("Solution found in", len(solution)-1, "moves:")
        print_solution(solution)
    else:
        print("No solution exists.")
```

```
Solution found in 3 moves:
(1, 2, 3)
(0, 4, 6)
(7, 5, 8)

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



```python
        print()

# Example Run
if __name__ == "__main__":
    # Example initial state (solvable)
    start_state = (1, 2, 3,
                   0, 4, 6,
                   7, 5, 8)

    solution = astar(start_state)
    if solution:
        print("Solution found in", len(solution)-1, "moves:")
        print_solution(solution)
    else:
        print("No solution exists.")
```

```
Solution found in 3 moves:
(1, 2, 3)
(0, 4, 6)
(7, 5, 8)

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

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

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