```
from collections import deque

def bfs(maze, start, end):
    # Directions: up, right, down, left
    directions = [(-1, 0), (0, 1), (1, 0), (0, -1)]
    queue = deque([start])  # Queue for BFS
    visited = set(start)    # Keep track of visited cells

    while queue:
        current = queue.popleft()
        if current == end:
            return True  # Path found to exit

        for direction in directions:
            # Calculate the next cell's position
            next_cell = (current[0] + direction[0], current[1] +
direction[1])

            # Check if the next cell is within the maze and not a wall
            if (0 <= next_cell[0] < len(maze) and
                    0 <= next_cell[1] < len(maze[0]) and
                    maze[next_cell[0]][next_cell[1]] != '#' and
                    next_cell not in visited):
                queue.append(next_cell)
                visited.add(next_cell)

    return False  # No path found

# Example maze where '#' is a wall, 'S' is start, and 'E' is end
maze = [
    ['S', '.', '.', '#', '.', '.', '.'],
    ['.', '#', '.', '#', '.', '#', '.'],
    ['.', '#', '.', '.', '.', '.', '.'],
    ['.', '.', '#', '#', '#', '.', '.'],
    ['.', '#', '.', '.', '.', '#', '.'],
    ['.', '#', '#', '#', '.', '#', '.'],
    ['.', '.', '.', '.', '.', '.', 'E'],
]

start = (0, 0)  # Starting position
```

```python
end = (6, 6)     # Ending position (exit)

# Run BFS to find the path
path_exists = bfs(maze, start, end)
print("Path found!" if path_exists else "No path exists.")
```