phase4.py > a_star

```python
import numpy as np
from queue import PriorityQueue
# Grid and obstacle setup
GRID_SIZE = 10
START = (0, 0)
GOAL = (8, 8)
# Randomly place obstacles
np.random.seed(1)
grid = np.zeros((GRID_SIZE, GRID_SIZE), dtype=int)
obstacle_count = 15
for _ in range(obstacle_count):
    x, y = np.random.randint(0, GRID_SIZE, size=2)
    if (x, y) not in [START, GOAL]:
        grid[y][x] = 1
def heuristic(a, b):
    return abs(a[0] - b[0]) + abs(a[1] - b[1])
def get_neighbors(pos):
    neighbors = []
    for dx, dy in [(-1,0),(1,0),(0,-1),(0,1)]:
        nx, ny = pos[0] + dx, pos[1] + dy
        if 0 <= nx < GRID_SIZE and 0 <= ny < GRID_SIZE and grid[ny][nx] == 0:
            neighbors.append((nx, ny))
    return neighbors
def a_star(start, goal):
    frontier = PriorityQueue()
    frontier.put((0, start))
    came_from = {start: None}
    cost = {start: 0}
```

phase4.py > a_star

```
24  def a_star(start, goal):
29
30      while not frontier.empty():
31          _, current = frontier.get()
32
33          if current == goal:
34              break
35
36          for neighbor in get_neighbors(current):
37              new_cost = cost[current] + 1
38              if neighbor not in cost or new_cost < cost[neighbor]:
39                  cost[neighbor] = new_cost
40                  priority = new_cost + heuristic(goal, neighbor)
41                  frontier.put((priority, neighbor))
42                  came_from[neighbor] = current
43
44      # Reconstruct path
45      path = []
46      node = goal
47      while node:
48          path.append(node)
49          node = came_from.get(node)
50      path.reverse()
51      return path if path[0] == start else []
52
53  # Run A* and print results
54  path = a_star(START, GOAL)
55  print("Grid (1 = obstacle):")
56  print(grid)
```

Ln 28, Col 22 Spaces: 4 UTF-8 CRLF {} Python 3.12.4 ('base')

phase4.py > a_star

```python

# Run A* and print results
path = a_star(START, GOAL)
print("Grid (1 = obstacle):")
print(grid)
print("\nPlanned Path from", START, "to", GOAL, ":")
print(path if path else "No path found.")
```