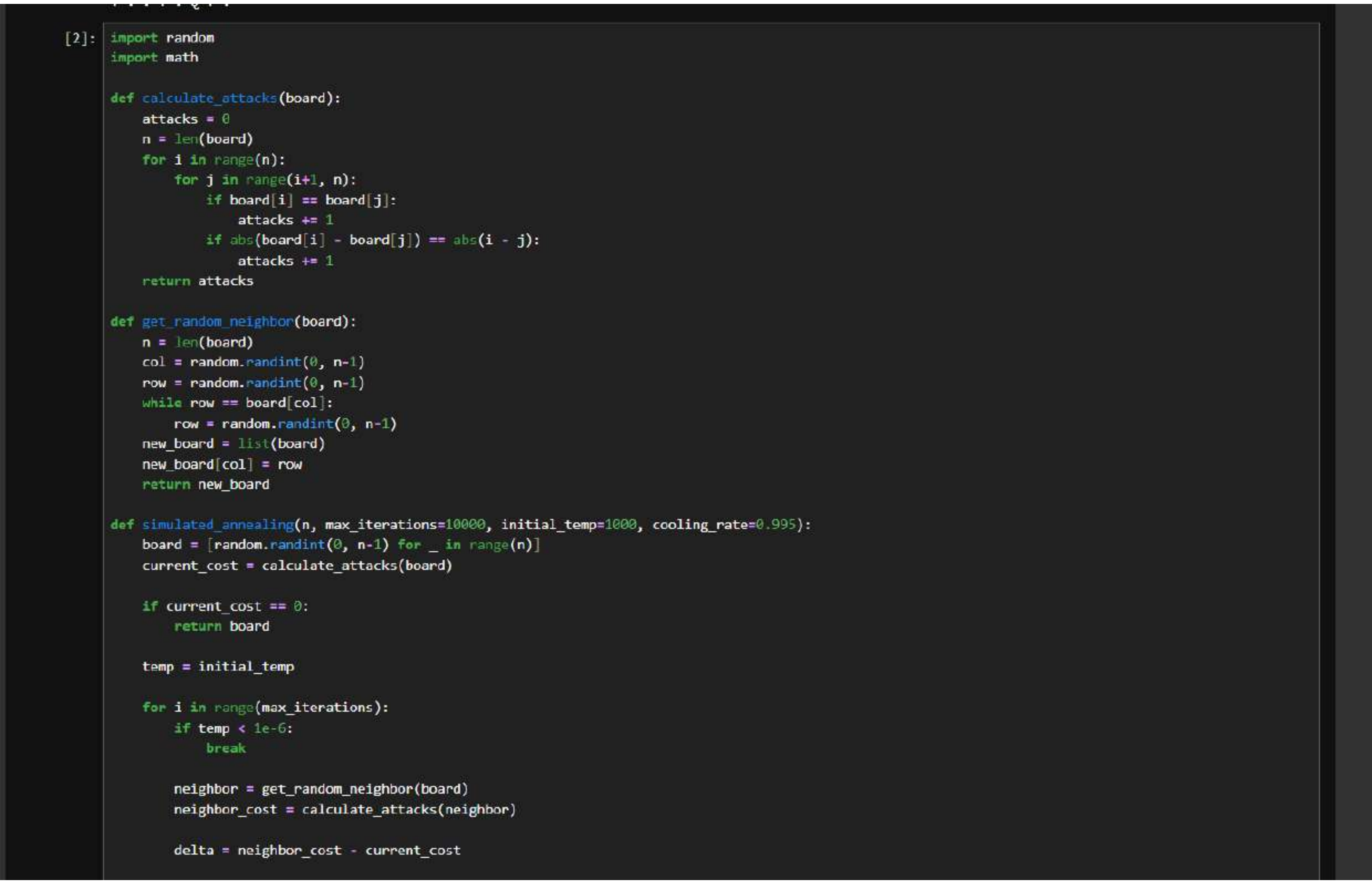
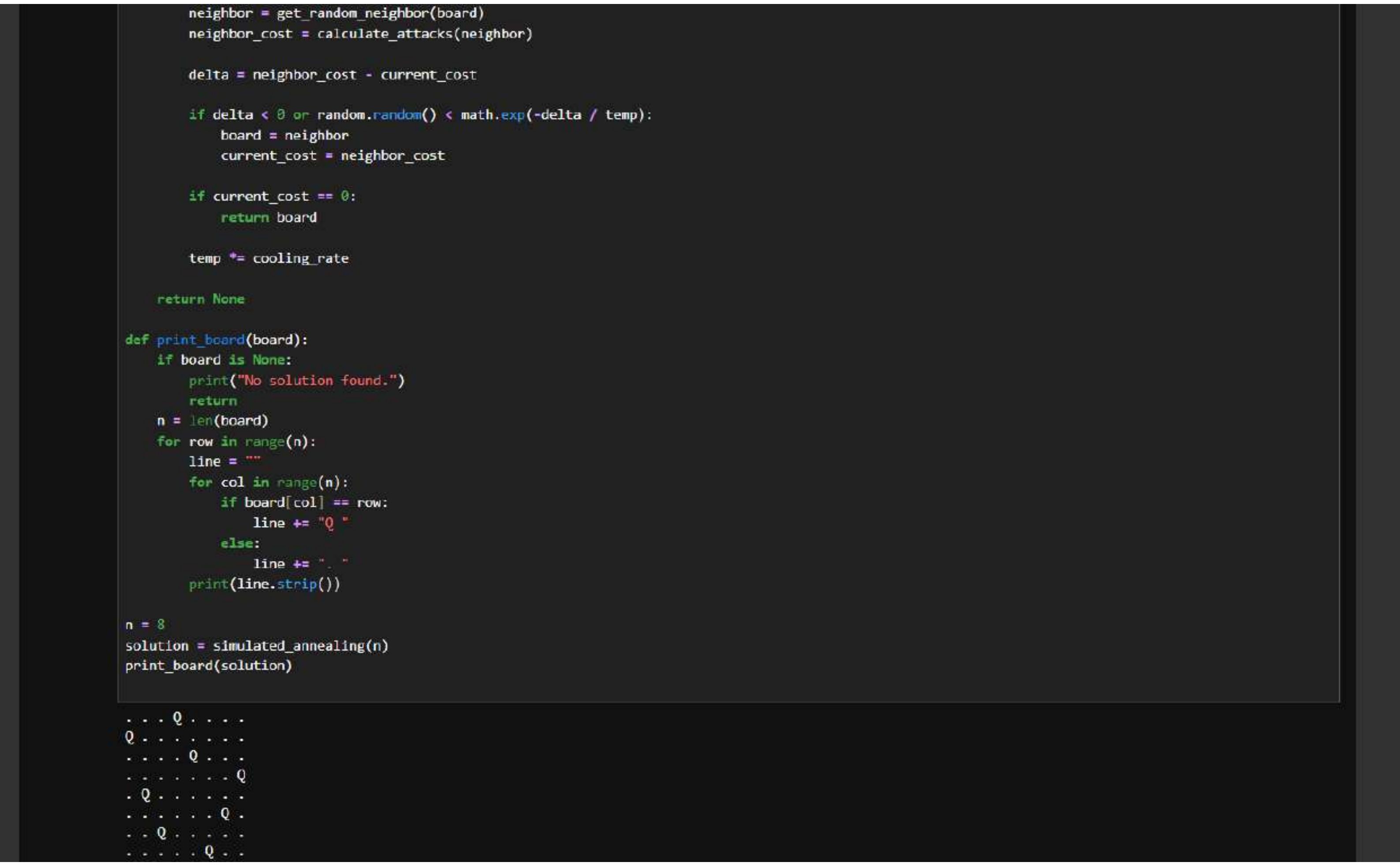


```
. . . . . Q . .
```

[2]:

```
import random
import math

def calculate_attacks(board):
    attacks = 0
    n = len(board)
    for i in range(n):
        for j in range(i+1, n):
            if board[i] == board[j]:
                attacks += 1
            if abs(board[i] - board[j]) == abs(i - j):
                attacks += 1
    return attacks

def get_random_neighbor(board):
    n = len(board)
    col = random.randint(0, n-1)
    row = random.randint(0, n-1)
    while row == board[col]:
        row = random.randint(0, n-1)
    new_board = list(board)
    new_board[col] = row
    return new_board

def simulated_annealing(n, max_iterations=10000, initial_temp=1000, cooling_rate=0.995):
    board = [random.randint(0, n-1) for _ in range(n)]
    current_cost = calculate_attacks(board)

    if current_cost == 0:
        return board

    temp = initial_temp

    for i in range(max_iterations):
        if temp < 1e-6:
            break

        neighbor = get_random_neighbor(board)
        neighbor_cost = calculate_attacks(neighbor)

        delta = neighbor_cost - current_cost
```



```
        neighbor = get_random_neighbor(board)
        neighbor_cost = calculate_attacks(neighbor)

        delta = neighbor_cost - current_cost

        if delta < 0 or random.random() < math.exp(-delta / temp):
            board = neighbor
            current_cost = neighbor_cost

        if current_cost == 0:
            return board

        temp *= cooling_rate

    return None

def print_board(board):
    if board is None:
        print("No solution found.")
        return
    n = len(board)
    for row in range(n):
        line = ""
        for col in range(n):
            if board[col] == row:
                line += "Q "
            else:
                line += ". "
        print(line.strip())

n = 8
solution = simulated_annealing(n)
print_board(solution)
```

```
. . . Q . . . .
Q . . . . . . .
. . . . Q . . .
. . . . . . . Q
. Q . . . . . .
. . . . . . Q .
. . Q . . . . .
. . . . . Q . .
```