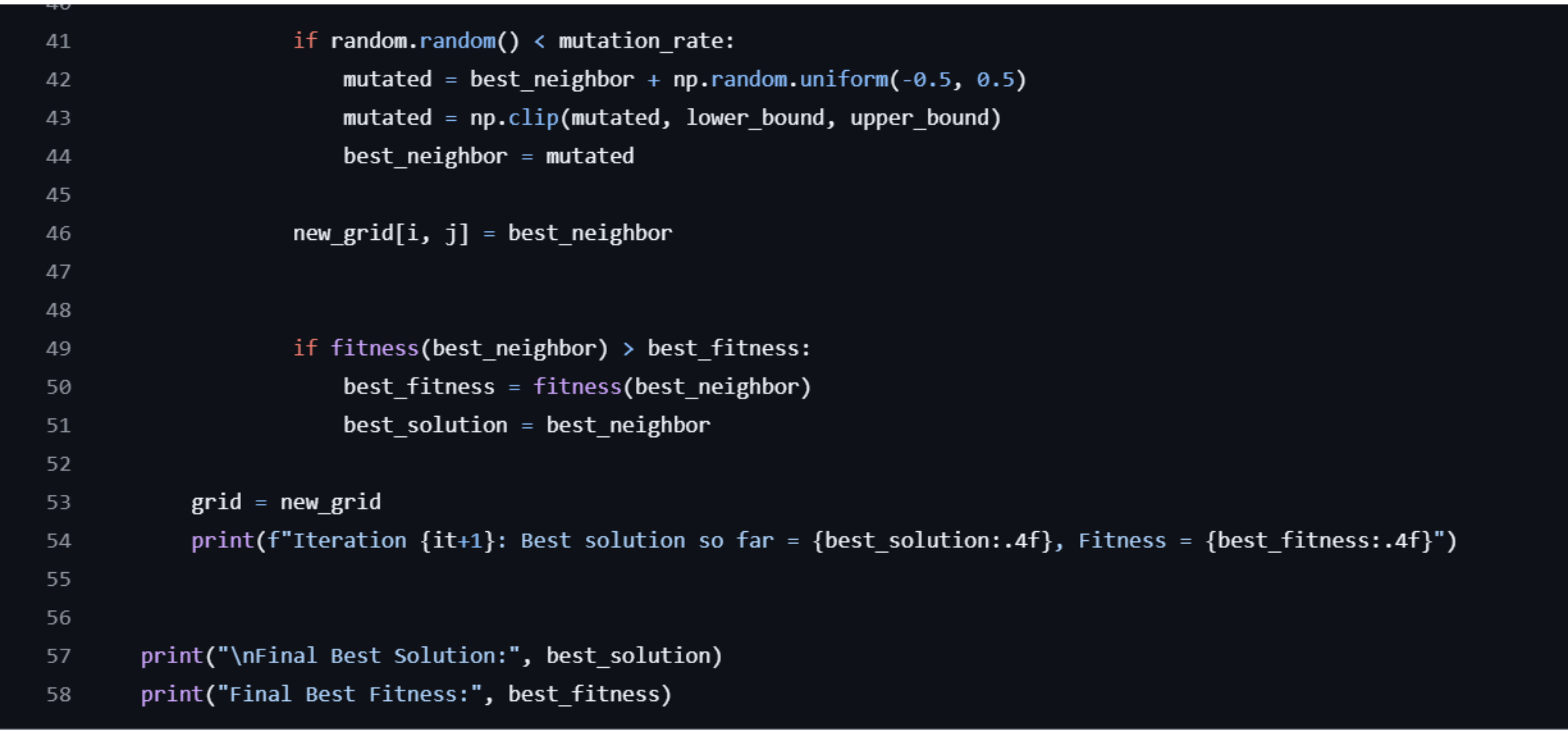
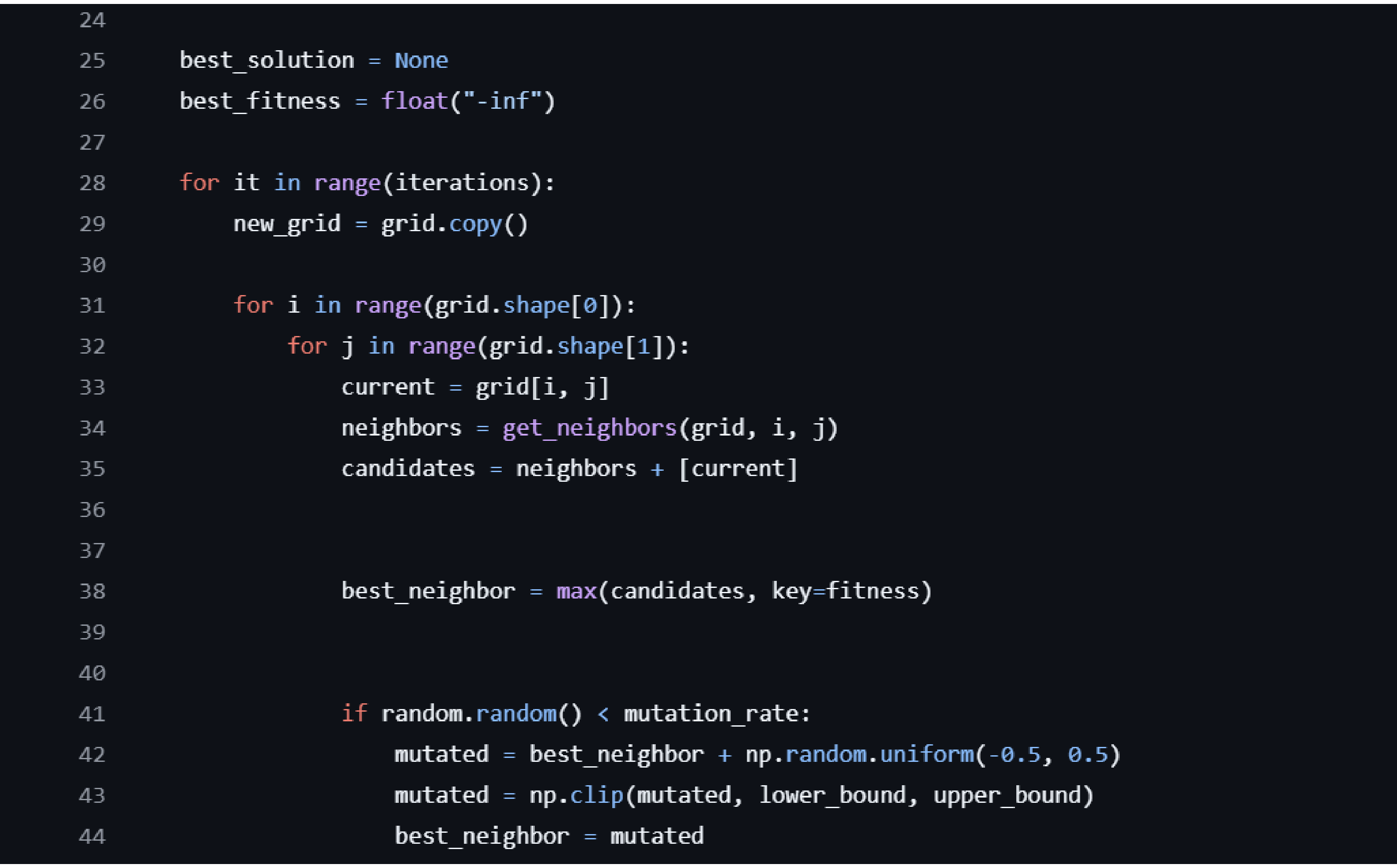
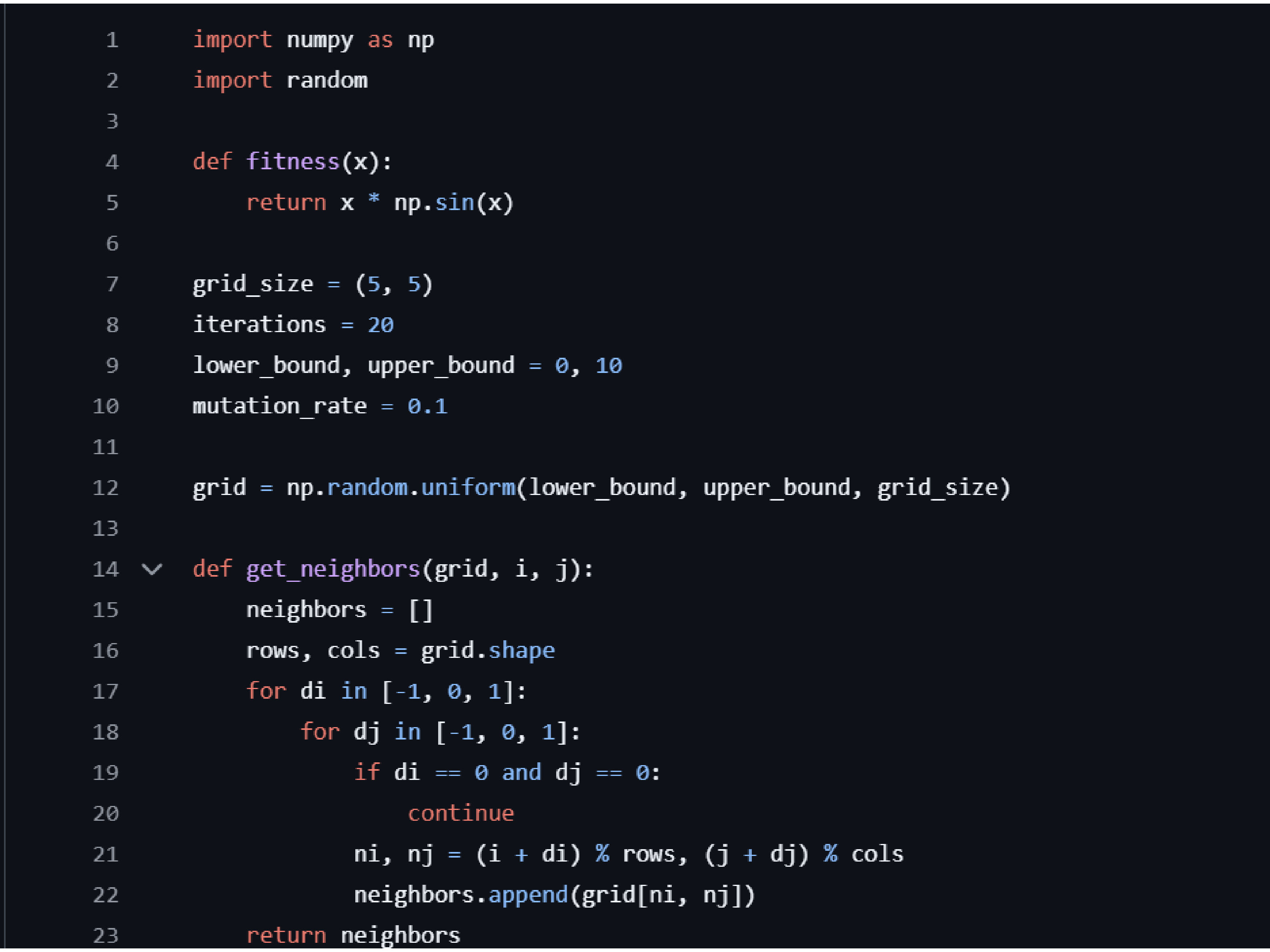


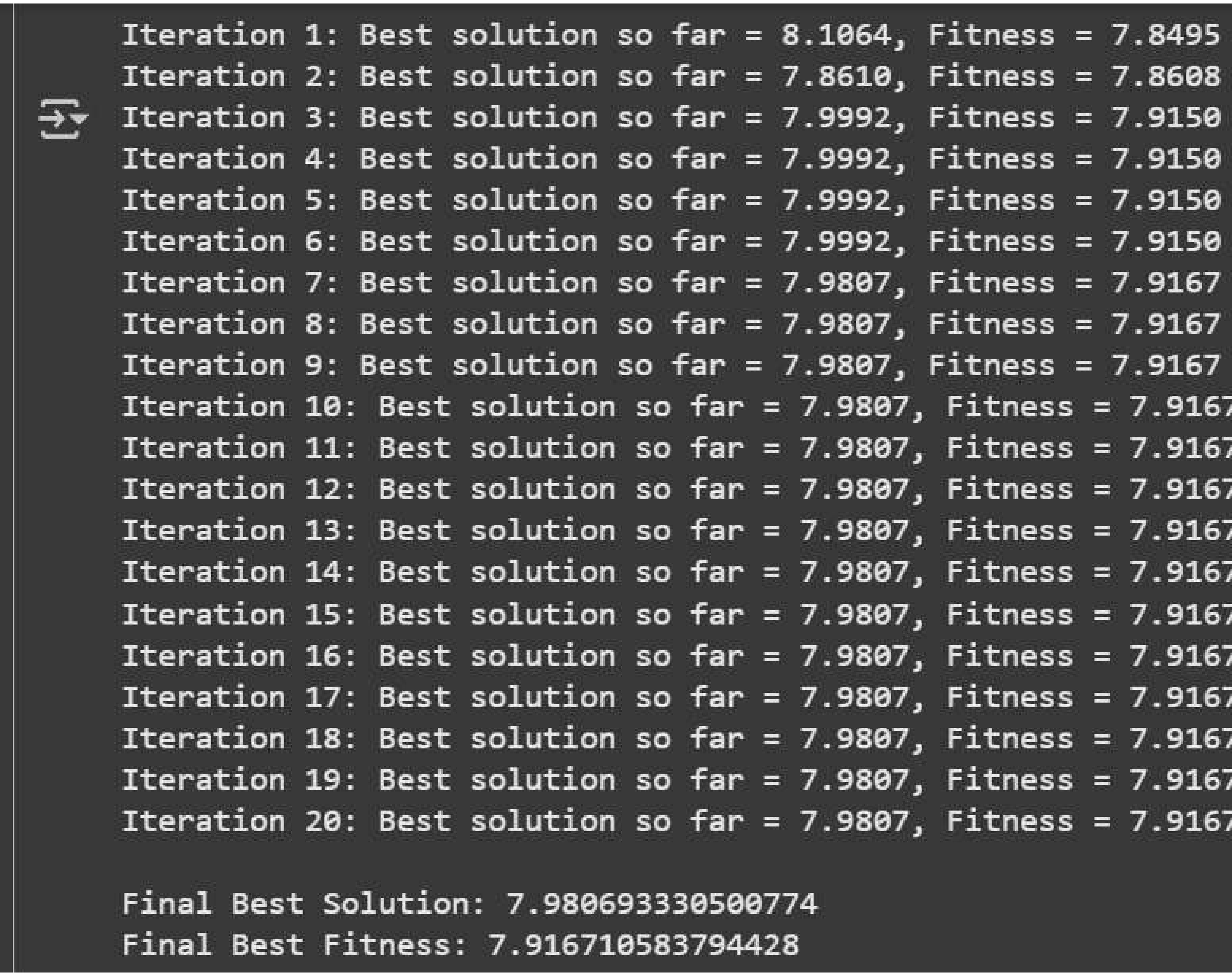
```
            if random.random() < mutation_rate:
                mutated = best_neighbor + np.random.uniform(-0.5, 0.5)
                mutated = np.clip(mutated, lower_bound, upper_bound)
                best_neighbor = mutated

            new_grid[i, j] = best_neighbor


            if fitness(best_neighbor) > best_fitness:
                best_fitness = fitness(best_neighbor)
                best_solution = best_neighbor

    grid = new_grid
    print(f"Iteration {it+1}: Best solution so far = {best_solution:.4f}, Fitness = {best_fitness:.4f}")


print("\nFinal Best Solution:", best_solution)
print("Final Best Fitness:", best_fitness)
```



```python

    best_solution = None
    best_fitness = float("-inf")

    for it in range(iterations):
        new_grid = grid.copy()

        for i in range(grid.shape[0]):
            for j in range(grid.shape[1]):
                current = grid[i, j]
                neighbors = get_neighbors(grid, i, j)
                candidates = neighbors + [current]


                best_neighbor = max(candidates, key=fitness)


                if random.random() < mutation_rate:
                    mutated = best_neighbor + np.random.uniform(-0.5, 0.5)
                    mutated = np.clip(mutated, lower_bound, upper_bound)
                    best_neighbor = mutated
```



```python
import numpy as np
import random

def fitness(x):
    return x * np.sin(x)

grid_size = (5, 5)
iterations = 20
lower_bound, upper_bound = 0, 10
mutation_rate = 0.1

grid = np.random.uniform(lower_bound, upper_bound, grid_size)

def get_neighbors(grid, i, j):
    neighbors = []
    rows, cols = grid.shape
    for di in [-1, 0, 1]:
        for dj in [-1, 0, 1]:
            if di == 0 and dj == 0:
                continue
            ni, nj = (i + di) % rows, (j + dj) % cols
            neighbors.append(grid[ni, nj])
    return neighbors
```



```
Iteration 1: Best solution so far = 8.1064, Fitness = 7.8495
Iteration 2: Best solution so far = 7.8610, Fitness = 7.8608
Iteration 3: Best solution so far = 7.9992, Fitness = 7.9150
Iteration 4: Best solution so far = 7.9992, Fitness = 7.9150
Iteration 5: Best solution so far = 7.9992, Fitness = 7.9150
Iteration 6: Best solution so far = 7.9992, Fitness = 7.9150
Iteration 7: Best solution so far = 7.9807, Fitness = 7.9167
Iteration 8: Best solution so far = 7.9807, Fitness = 7.9167
Iteration 9: Best solution so far = 7.9807, Fitness = 7.9167
Iteration 10: Best solution so far = 7.9807, Fitness = 7.9167
Iteration 11: Best solution so far = 7.9807, Fitness = 7.9167
Iteration 12: Best solution so far = 7.9807, Fitness = 7.9167
Iteration 13: Best solution so far = 7.9807, Fitness = 7.9167
Iteration 14: Best solution so far = 7.9807, Fitness = 7.9167
Iteration 15: Best solution so far = 7.9807, Fitness = 7.9167
Iteration 16: Best solution so far = 7.9807, Fitness = 7.9167
Iteration 17: Best solution so far = 7.9807, Fitness = 7.9167
Iteration 18: Best solution so far = 7.9807, Fitness = 7.9167
Iteration 19: Best solution so far = 7.9807, Fitness = 7.9167
Iteration 20: Best solution so far = 7.9807, Fitness = 7.9167

Final Best Solution: 7.980693330500774
Final Best Fitness: 7.916710583794428
```