```
import numpy as np
    import random
    import matplotlib.pyplot as plt
    def rastrigin(x):
        return 10 * len(x) + sum([(xi ** 2 - 10 * np.cos(2 * np.pi * xi)) for xi in x])
    class Particle:
        def __init__(self, dim, bounds):
            self.position = np.random.uniform(bounds[0], bounds[1], dim)
            self.velocity = np.random.uniform(-1, 1, dim)
            self.best_position = np.copy(self.position)
            self.best value = rastrigin(self.position)
        def evaluate(self):
            current_value = rastrigin(self.position)
            if current_value < self.best_value:
                self.best_value = current_value
                self.best_position = np.copy(self.position)
    def pso(dim, bounds, num_particles=30, max_iter=100, w=0.5, c1=1.5, c2=1.5):
        particles = [Particle(dim, bounds) for _ in range(num_particles)]
        global_best_position = None
        global_best_value = float('inf')
        best_values_over_iterations = []
        for iter in range(max_iter):
            for particle in particles:
                particle.evaluate()
                if particle.best_value < global_best_value:
                    global_best_value = particle.best_value
                    global_best_position = np.copy(particle.best_position)
            best_values_over_iterations.append(global_best_value)
            for particle in particles:
                inertia = w * particle.velocity
                cognitive = c1 * np.random.random() * (particle.best_position - particle.position)
                social = c2 * np.random.random() * (global_best_position - particle.position)
                particle.velocity = inertia + cognitive + social
                particle.position = particle.position + particle.velocity
                particle.position = np.clip(particle.position, bounds[0], bounds[1])
            if (iter+1) % 10 == 0:
                print(f"Iteration {iter+1}/{max_iter}, Global Best Value: {global_best_value}")
        return global_best_position, global_best_value, particles, best_values_over_iterations
    if __name__ == "__main__":
        dim = 2
        bounds = [-5.12, 5.12]
        best_position, best_value, particles, best_values_over_iterations = pso(dim, bounds, num_particles=30, max_iter=100)
        print("\nFinal Best Position:", best_position)
        print("Final Best Value:", best_value)
        fig, ax = plt.subplots(figsize=(8, 6))
        final_best_positions = np.array([particle.best_position for particle in particles])
        ax.scatter(final_best_positions[:, 0], final_best_positions[:, 1], color='blue', label="Particle Best Positions", alpha=0.7)
```

```
ax.set_title("Final Particle Positions in PSO")
ax.set_xlabel("Dimension 1")
ax.set_ylabel("Dimension 2")
ax.legend()
plt.grid(True)
plt.show()

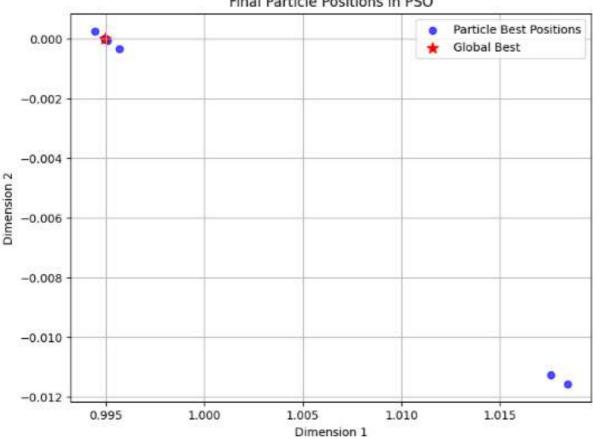
plt.figure(figsize=(8, 6))
plt.plot(range(1, len(best_values_over_iterations) + 1), best_values_over_iterations, color='green')
plt.title("Global Best Value vs. Iterations")
plt.xlabel("Iterations")
plt.ylabel("Global Best Value (Fitness)")
plt.grid(True)
plt.show()
```

Iteration 10/100, Global Best Value: 2.003250667292207 Iteration 20/100, Global Best Value: 1.0371970536607833 Iteration 30/100, Global Best Value: 0.9965161455248861 Iteration 40/100, Global Best Value: 0.9949848667711656 Iteration 50/100, Global Best Value: 0.9949610887864182 Iteration 60/100, Global Best Value: 0.994959059938175 Iteration 70/100, Global Best Value: 0.9949590571109823 Iteration 80/100, Global Best Value: 0.9949590570934674 Iteration 90/100, Global Best Value: 0.9949590570932898 Iteration 100/100, Global Best Value: 0.9949590570932898

Final Best Position: [9.94958638e-01 -8.70961770e-10]

Final Best Value: 0.9949590570932898

Final Particle Positions in PSO



Global Best Value vs. Iterations

