```
import numpy as np

# Rastrigin function
def rastrigin(x):
    A = 10
    return A * len(x) + sum(xi**2 - A * np.cos(2 * np.pi * xi) for xi in x)

# Particle Swarm Optimization (PSO)
def pso(num_particles=30, num_iterations=100, dim=2, w=0.5, c1=1.5, c2=1.5):
    # Initialize particles' positions and velocities
    positions = np.random.uniform(-5.12, 5.12, (num_particles, dim))  # Random positions in [-5.12, 5.12]
    velocities = np.random.uniform(-1, 1, (num_particles, dim))  # Random velocities
    personal_best_positions = positions.copy()
    personal_best_scores = np.array([rastrigin(p) for p in positions])

    global_best_position = personal_best_positions[np.argmin(personal_best_scores)]
    global_best_score = np.min(personal_best_scores)

    # PSO iterations
    for iteration in range(num_iterations):
        for i in range(num_particles):
            # Update particle's velocity
            r1, r2 = np.random.rand(2)  # Random coefficients
            velocities[i] = w * velocities[i] + c1 * r1 * (personal_best_positions[i] - positions[i]) + c2 * r2 * (global_best_position -

            # Update particle's position
            positions[i] += velocities[i]

            # Ensure positions are within bounds
            positions[i] = np.clip(positions[i], -5.12, 5.12)

            # Evaluate new fitness
            current_score = rastrigin(positions[i])

            # Update personal best position and score
            if current_score < personal_best_scores[i]:
                personal_best_positions[i] = positions[i]
                personal_best_scores[i] = current_score

        # Update global best position and score
        best_particle_idx = np.argmin(personal_best_scores)
        if personal_best_scores[best_particle_idx] < global_best_score:
            global_best_position = personal_best_positions[best_particle_idx]
            global_best_score = personal_best_scores[best_particle_idx]

        # Print progress for each iteration
        print(f"Iteration {iteration + 1}, Best fitness: {global_best_score}")

    return global_best_position, global_best_score

# Run PSO for 100 iterations
best_position, best_score = pso(num_particles=30, num_iterations=100, dim=2)

# Print the best result
print("\nBest position found:", best_position)
print("Best fitness (Rastrigin function value):", best_score)
```

```
Iteration 6, Best fitness: 2.0054151779227993
Iteration 7, Best fitness: 2.0054151779227993
Iteration 8, Best fitness: 2.000339992679656
Iteration 9, Best fitness: 1.9477998176853077
Iteration 10, Best fitness: 1.4028465810203095
Iteration 11, Best fitness: 1.4028465810203095
Iteration 12, Best fitness: 1.4028465810203095
Iteration 13, Best fitness: 1.4028465810203095
Iteration 14, Best fitness: 1.1686157957896128
Iteration 15, Best fitness: 1.1686157957896128
Iteration 16, Best fitness: 1.1686157957896128
Iteration 17, Best fitness: 1.0058796625981685
Iteration 18, Best fitness: 1.0058796625981685
Iteration 19, Best fitness: 0.9972130021306391
Iteration 20, Best fitness: 0.9972130021306391
Iteration 21, Best fitness: 0.9972130021306391
Iteration 22, Best fitness: 0.9972130021306391
Iteration 23, Best fitness: 0.9972130021306391
Iteration 24, Best fitness: 0.9972130021306391
Iteration 25, Best fitness: 0.9972130021306391
```

```
Iteration 31, Best fitness: 0.9953420041803582
Iteration 32, Best fitness: 0.9949653158236558
Iteration 33, Best fitness: 0.9949653158236558
Iteration 34, Best fitness: 0.9949653158236558
Iteration 35, Best fitness: 0.9949653158236558
Iteration 36, Best fitness: 0.9949653158236558
Iteration 37, Best fitness: 0.9949653158236558
Iteration 38, Best fitness: 0.9949591786684522
Iteration 39, Best fitness: 0.9949591786684522
Iteration 40, Best fitness: 0.9949591786684522
Iteration 41, Best fitness: 0.9949591786684522
Iteration 42, Best fitness: 0.9949591786684522
Iteration 43, Best fitness: 0.9949591056798752
Iteration 44, Best fitness: 0.9949591056798752
Iteration 45, Best fitness: 0.9949590589486483
Iteration 46, Best fitness: 0.9949590589486483
Iteration 47, Best fitness: 0.9949590589486483
Iteration 48, Best fitness: 0.9949590589486483
Iteration 49, Best fitness: 0.9949590589486483
Iteration 50, Best fitness: 0.9949590589486483
Iteration 51, Best fitness: 0.9949590589486483
Iteration 52, Best fitness: 0.9949590580503873
Iteration 53, Best fitness: 0.9949590580503873
Iteration 54, Best fitness: 0.9949590580319096
Iteration 55, Best fitness: 0.9949590580319096
Iteration 56, Best fitness: 0.9949590580319096
Iteration 57, Best fitness: 0.9949590574311458
Iteration 58, Best fitness: 0.9949590571426512
Iteration 59, Best fitness: 0.9949590571426512
Iteration 60, Best fitness: 0.9949590571078808
Iteration 61, Best fitness: 0.9949590570958193
Iteration 62, Best fitness: 0.9949590570958193
Iteration 63, Best fitness: 0.9949590570934674
```