```
import numpy as np

def rastrigin_function(x):
    return 10 * len(x) + sum(x_i ** 2 - 10 * np.cos(2 * np.pi * x_i) for x_i in x)

def pso(objective_function, dimensions=2, num_particles=30, max_iter=100, bounds=(-5.12, 5.12), w=0.7, c1=1.5, c2=1.5):
    particles = np.random.uniform(bounds[0], bounds[1], (num_particles, dimensions))
    velocities = np.random.uniform(-1, 1, (num_particles, dimensions))
    personal_best_positions = np.copy(particles)
    personal_best_scores = np.array([objective_function(p) for p in particles])
    global_best_position = personal_best_positions[np.argmin(personal_best_scores)]
    global_best_score = min(personal_best_scores)

    for _ in range(max_iter):
        for i in range(num_particles):
            inertia = w * velocities[i]
            cognitive = c1 * np.random.rand(dimensions) * (personal_best_positions[i] - particles[i])
            social = c2 * np.random.rand(dimensions) * (global_best_position - particles[i])
            velocities[i] = inertia + cognitive + social

            particles[i] += velocities[i]
            particles[i] = np.clip(particles[i], bounds[0], bounds[1])

            score = objective_function(particles[i])
            if score < personal_best_scores[i]:
                personal_best_scores[i] = score
                personal_best_positions[i] = np.copy(particles[i])

                if score < global_best_score:
                    global_best_score = score
                    global_best_position = np.copy(particles[i])

    return global_best_position, global_best_score

best_position, best_score = pso(rastrigin_function)
print(f"Best Position: {best_position}")
print(f"Best Score: {best_score}")
```

```
Best Position: [-2.69320356e-07  2.23467065e-06]
Best Score: 1.005112437724165e-09
```