```
import random
import math
import copy


# Fitness functions

# Rastrigin function
def fitness_rastrigin(position):
    fitness_value = 0.0
    for xi in position:
        fitness_value += (xi ** 2) - (10 * math.cos(2 * math.pi * xi)) + 10
    return fitness_value

# Sphere function
def fitness_sphere(position):
    fitness_value = sum(xi ** 2 for xi in position)
    return fitness_value


# Wolf class
class Wolf:
    def __init__(self, fitness, dim, minx, maxx, seed):
        self.rnd = random.Random(seed)
        self.position = [self.rnd.uniform(minx, maxx) for _ in range(dim)]
        self.fitness = fitness(self.position)


# Grey Wolf Optimization (GWO)
def gwo(fitness, max_iter, n, dim, minx, maxx):
    rnd = random.Random(0)

    # Create n random wolves
    population = [Wolf(fitness, dim, minx, maxx, i) for i in range(n)]

    # Sort the population based on fitness values (ascending order)
    population = sorted(population, key=lambda wolf: wolf.fitness)

    # Best 3 solutions will be called as alpha, beta, and gamma
    alpha_wolf, beta_wolf, gamma_wolf = copy.copy(population[:3])

    # Main loop of GWO
    for Iter in range(max_iter):
        # Print iteration number and best fitness value
        if Iter % 10 == 0 and Iter > 1:
            print(f"Iter = {Iter}, best fitness = {alpha_wolf.fitness:.3f}")

        # Linearly decrease a from 2 to 0
        a = 2 * (1 - Iter / max_iter)

        # Update each population member with the help of best three members
        for i in range(n):
            A1, A2, A3 = [a * (2 * rnd.random() - 1) for _ in range(3)]
            C1, C2, C3 = [2 * rnd.random() for _ in range(3)]

            # Update the wolf's position
            Xnew = [0.0 for _ in range(dim)]
            for j in range(dim):
                X1 = alpha_wolf.position[j] - A1 * abs(C1 * alpha_wolf.position[j] - population[i].position[j])
                X2 = beta_wolf.position[j] - A2 * abs(C2 * beta_wolf.position[j] - population[i].position[j])
                X3 = gamma_wolf.position[j] - A3 * abs(C3 * gamma_wolf.position[j] - population[i].position[j])

                Xnew[j] = (X1 + X2 + X3) / 3.0

            # Evaluate the fitness of the new position
            fnew = fitness(Xnew)

            # Greedy selection: update position if the new solution is better
            if fnew < population[i].fitness:
                population[i].position = Xnew
                population[i].fitness = fnew

        # Re-sort the population based on fitness values
        population = sorted(population, key=lambda wolf: wolf.fitness)

        # Update alpha, beta, and gamma wolves
```

```
            alpha_wolf, beta_wolf, gamma_wolf = copy.copy(population[:3])

        # Return the best solution
        return alpha_wolf.position


    # Driver code for Rastrigin function
    print("\nBegin grey wolf optimization on Rastrigin function\n")
    dim = 3
    fitness = fitness_rastrigin
    print(f"Goal is to minimize Rastrigin's function in {dim} variables")
    print(f"Function has known min = 0.0 at ({', '.join(['0'] * (dim - 1))}, 0)")

    num_particles = 50
    max_iter = 100
    print(f"Setting num_particles = {num_particles}")
    print(f"Setting max_iter = {max_iter}")
    print("\nStarting GWO algorithm\n")

    best_position = gwo(fitness, max_iter, num_particles, dim, -10.0, 10.0)

    print("\nGWO completed\n")
    print("\nBest solution found:")
    print([f"{best_position[k]:.6f}" for k in range(dim)])
    err = fitness(best_position)
    print(f"Fitness of best solution = {err:.6f}")

    print("\nEnd GWO for Rastrigin\n")

    print()


    # Driver code for Sphere function
    print("\nBegin grey wolf optimization on Sphere function\n")
    dim = 3
    fitness = fitness_sphere
    print(f"Goal is to minimize Sphere function in {dim} variables")
    print(f"Function has known min = 0.0 at ({', '.join(['0'] * (dim - 1))}, 0)")

    num_particles = 50
    max_iter = 100
    print(f"Setting num_particles = {num_particles}")
    print(f"Setting max_iter = {max_iter}")
    print("\nStarting GWO algorithm\n")

    best_position = gwo(fitness, max_iter, num_particles, dim, -10.0, 10.0)

    print("\nGWO completed\n")
    print("\nBest solution found:")
    print([f"{best_position[k]:.6f}" for k in range(dim)])
    err = fitness(best_position)
    print(f"Fitness of best solution = {err:.6f}")

    print("\nEnd GWO for Sphere\n")
```

```
Begin grey wolf optimization on Rastrigin function

Goal is to minimize Rastrigin's function in 3 variables
Function has known min = 0.0 at (0, 0, 0)
Setting num_particles = 50
Setting max_iter = 100

Starting GWO algorithm

Iter = 10, best fitness = 6.636
Iter = 20, best fitness = 1.047
Iter = 30, best fitness = 1.012
Iter = 40, best fitness = 1.010
Iter = 50, best fitness = 1.008
Iter = 60, best fitness = 1.008
Iter = 70, best fitness = 1.008
Iter = 80, best fitness = 1.006
Iter = 90, best fitness = 1.005

GWO completed
```

```
[ -0.004395 ,  0.995042 ,  0.005800 ]
Fitness of best solution = 1.005465

End GWO for Rastrigin



Begin grey wolf optimization on Sphere function

Goal is to minimize Sphere function in 3 variables
Function has known min = 0.0 at (0, 0, 0)
Setting num_particles = 50
Setting max_iter = 100

Starting GWO algorithm

Iter = 10, best fitness = 0.000
Iter = 20, best fitness = 0.000
Iter = 30, best fitness = 0.000
Iter = 40, best fitness = 0.000
Iter = 50, best fitness = 0.000
Iter = 60, best fitness = 0.000
Iter = 70, best fitness = 0.000
Iter = 80, best fitness = 0.000
Iter = 90, best fitness = 0.000

GWO completed


Best solution found:
['0.000000', '0.000000', '-0.000000']
Fitness of best solution = 0.000000

End GWO for Sphere
```