```
import numpy as np

# Objective function (Example: Sphere function)
def objective_function(x):
    return np.sum(x**2)

# Grey Wolf Optimization Algorithm
class GreyWolfOptimizer:
    def __init__(self, obj_func, dim, n_wolves, max_iter, lb, ub):
        self.obj_func = obj_func  # Objective function
        self.dim = dim            # Dimensionality of the problem
        self.n_wolves = n_wolves # Number of wolves
        self.max_iter = max_iter # Maximum number of iterations
        self.lb = lb              # Lower bound of the search space
        self.ub = ub              # Upper bound of the search space
        self.alpha_pos = np.zeros(dim)  # Position of alpha wolf
        self.alpha_score = float("inf") # Fitness of alpha wolf
        self.beta_pos = np.zeros(dim)   # Position of beta wolf
        self.beta_score = float("inf")  # Fitness of beta wolf
        self.delta_pos = np.zeros(dim)  # Position of delta wolf
        self.delta_score = float("inf") # Fitness of delta wolf
        self.positions = np.random.rand(n_wolves, dim) * (ub - lb) + lb  # Initial positions of wolves
        self.scores = np.zeros(n_wolves)  # Fitness scores

    def optimize(self):
        # Main optimization loop
        for t in range(self.max_iter):
            a = 2 - t * (2 / self.max_iter)  # Decreases linearly from 2 to 0
            for i in range(self.n_wolves):
                # Evaluate fitness of each wolf
                self.scores[i] = self.obj_func(self.positions[i])

                # Update alpha, beta, and delta wolves
                if self.scores[i] < self.alpha_score:
                    self.alpha_score = self.scores[i]
                    self.alpha_pos = self.positions[i]
                elif self.scores[i] < self.beta_score:
                    self.beta_score = self.scores[i]
                    self.beta_pos = self.positions[i]
                elif self.scores[i] < self.delta_score:
                    self.delta_score = self.scores[i]
                    self.delta_pos = self.positions[i]

            # Update the positions of the wolves
            for i in range(self.n_wolves):
                # Calculate random values for A and C
                r1 = np.random.rand(self.dim)
                r2 = np.random.rand(self.dim)
                A = 2 * a * r1 - a
                C = 2 * r2

                # Update the position of the wolf
                D_alpha = np.abs(C * self.alpha_pos - self.positions[i])
                D_beta = np.abs(C * self.beta_pos - self.positions[i])
                D_delta = np.abs(C * self.delta_pos - self.positions[i])

                X1 = self.alpha_pos - A * D_alpha
                X2 = self.beta_pos - A * D_beta
                X3 = self.delta_pos - A * D_delta

                # Calculate new position for the wolf
                self.positions[i] = (X1 + X2 + X3) / 3

                # Apply boundary constraints (if any)
                self.positions[i] = np.clip(self.positions[i], self.lb, self.ub)

            # Optionally, print the best solution found so far
            print(f"Iteration {t+1}/{self.max_iter} - Best Score: {self.alpha_score}")

        # Return the best solution found
        return self.alpha_pos, self.alpha_score

# Hyperparameters
dim = 30              # Number of dimensions (variables)
n_wolves = 50         # Number of wolves (population size)
max_iter = 100        # Maximum number of iterations
```

```
lb = -10              # Lower bound of search space
ub = 10               # Upper bound of search space

# Instantiate the optimizer
optimizer = GreyWolfOptimizer(obj_func=objective_function, dim=dim, n_wolves=n_wolves, max_iter=max_iter, lb=lb, ub=ub)

# Perform optimization
best_position, best_score = optimizer.optimize()

# Output the best solution found
print("\nBest Position: ", best_position)
print("Best Score: ", best_score)
```

```
Iteration 1/100 - Best Score: 576.8546371418655
Iteration 2/100 - Best Score: 576.8546371418655
Iteration 3/100 - Best Score: 576.8546371418655
Iteration 4/100 - Best Score: 576.8546371418655
Iteration 5/100 - Best Score: 576.8546371418655
Iteration 6/100 - Best Score: 576.8546371418655
Iteration 7/100 - Best Score: 576.8546371418655
Iteration 8/100 - Best Score: 576.8546371418655
Iteration 9/100 - Best Score: 576.8546371418655
Iteration 10/100 - Best Score: 567.6503411188247
Iteration 11/100 - Best Score: 332.98449995146206
Iteration 12/100 - Best Score: 332.98449995146206
Iteration 13/100 - Best Score: 332.98449995146206
Iteration 14/100 - Best Score: 320.95353655339466
Iteration 15/100 - Best Score: 320.95353655339466
Iteration 16/100 - Best Score: 320.95353655339466
Iteration 17/100 - Best Score: 320.95353655339466
Iteration 18/100 - Best Score: 320.95353655339466
Iteration 19/100 - Best Score: 320.95353655339466
Iteration 20/100 - Best Score: 320.95353655339466
Iteration 21/100 - Best Score: 320.95353655339466
Iteration 22/100 - Best Score: 320.95353655339466
Iteration 23/100 - Best Score: 320.95353655339466
Iteration 24/100 - Best Score: 320.95353655339466
Iteration 25/100 - Best Score: 320.95353655339466
Iteration 26/100 - Best Score: 193.58689098783609
Iteration 27/100 - Best Score: 193.58689098783609
Iteration 28/100 - Best Score: 193.58689098783609
Iteration 29/100 - Best Score: 193.58689098783609
Iteration 30/100 - Best Score: 165.4989403601319
Iteration 31/100 - Best Score: 141.7637082984632
Iteration 32/100 - Best Score: 107.95359960965696
Iteration 33/100 - Best Score: 91.19845823044612
Iteration 34/100 - Best Score: 91.19845823044612
Iteration 35/100 - Best Score: 91.19845823044612
Iteration 36/100 - Best Score: 91.19845823044612
Iteration 37/100 - Best Score: 91.19845823044612
Iteration 38/100 - Best Score: 91.19845823044612
Iteration 39/100 - Best Score: 87.74441078612715
Iteration 40/100 - Best Score: 58.950262931585634
Iteration 41/100 - Best Score: 43.17916057457401
Iteration 42/100 - Best Score: 26.407327961339835
Iteration 43/100 - Best Score: 23.17725687432357
Iteration 44/100 - Best Score: 18.043511016987747
Iteration 45/100 - Best Score: 11.95134015089711
Iteration 46/100 - Best Score: 9.448487385144116
Iteration 47/100 - Best Score: 5.680343879506539
Iteration 48/100 - Best Score: 5.680343879506539
Iteration 49/100 - Best Score: 4.1510350687146005
Iteration 50/100 - Best Score: 3.1123097644703157
Iteration 51/100 - Best Score: 2.561527570558623
Iteration 52/100 - Best Score: 1.7757975432969382
Iteration 53/100 - Best Score: 1.132018555611898
Iteration 54/100 - Best Score: 0.8121393487687071
Iteration 55/100 - Best Score: 0.774808018393388
Iteration 56/100 - Best Score: 0.5800107679412722
Iteration 57/100 - Best Score: 0.5068993625211085
Iteration 58/100 - Best Score: 0.33100752298407066
```