Grey Wolf Optimizer (GWO):

```
import numpy as np
def objective_function(x):
    return np.sum(x**2)
def grey_wolf_optimizer(obj_func, dim=2, n_wolves=20, max_iter=100):

wolves = np.random.uniform(-5, 5, (n_wolves, dim))
 alpha_pos = np.zeros(dim)
    beta_pos = np.zeros(dim)
    delta_pos = np.zeros(dim)
    alpha_score = float("inf")
    beta_score = float("inf")
    delta_score = float("inf")

    for t in range(max_iter):
        for i in range(n_wolves):

            fitness = obj_func(wolves[i])
 if fitness < alpha_score:
                delta_score = beta_score
                delta_pos = beta_pos.copy()
                beta_score = alpha_score
                beta_pos = alpha_pos.copy()
                alpha_score = fitness
                alpha_pos = wolves[i].copy()
            elif fitness < beta_score:
                delta_score = beta_score
                delta_pos = beta_pos.copy()
                beta_score = fitness
                beta_pos = wolves[i].copy()
            elif fitness < delta_score:
                delta_score = fitness
                delta_pos = wolves[i].copy()
```

```
    for t in range(max_iter):
        for i in range(n_wolves):

            fitness = obj_func(wolves[i])
if fitness < alpha_score:
                delta_score = beta_score
                delta_pos = beta_pos.copy()
                beta_score = alpha_score
                beta_pos = alpha_pos.copy()
                alpha_score = fitness
                alpha_pos = wolves[i].copy()
            elif fitness < beta_score:
                delta_score = beta_score
                delta_pos = beta_pos.copy()
                beta_score = fitness
                beta_pos = wolves[i].copy()
            elif fitness < delta_score:
                delta_score = fitness
                delta_pos = wolves[i].copy()


        a = 2 - t * (2 / max_iter)


        for i in range(n_wolves):
            for j in range(dim):
                r1, r2 = np.random.rand(), np.random.rand()
                A1 = 2 * a * r1 - a
                C1 = 2 * r2
                D_alpha = abs(C1 * alpha_pos[j] - wolves[i][j])
                X1 = alpha_pos[j] - A1 * D_alpha
```

```
        for i in range(n_wolves):
            for j in range(dim):
                r1, r2 = np.random.rand(), np.random.rand()
                A1 = 2 * a * r1 - a
                C1 = 2 * r2
                D_alpha = abs(C1 * alpha_pos[j] - wolves[i][j])
                X1 = alpha_pos[j] - A1 * D_alpha

                r1, r2 = np.random.rand(), np.random.rand()
                A2 = 2 * a * r1 - a
                C2 = 2 * r2
                D_beta = abs(C2 * beta_pos[j] - wolves[i][j])
                X2 = beta_pos[j] - A2 * D_beta

                r1, r2 = np.random.rand(), np.random.rand()
                A3 = 2 * a * r1 - a
                C3 = 2 * r2
                D_delta = abs(C3 * delta_pos[j] - wolves[i][j])
                X3 = delta_pos[j] - A3 * D_delta

                wolves[i][j] = (X1 + X2 + X3) / 3


            wolves[i] = np.clip(wolves[i], -5, 5)

        print(f"Iteration {t+1}: Best Fitness = {alpha_score:.6f}")

    return alpha_pos, alpha_score

best_pos, best_val = grey_wolf_optimizer(objective_function, dim=2, n_wolves=25, max_iter=50)
print("\nBest Solution:", best_pos)
print("Best Fitness Value:", best_val)
```