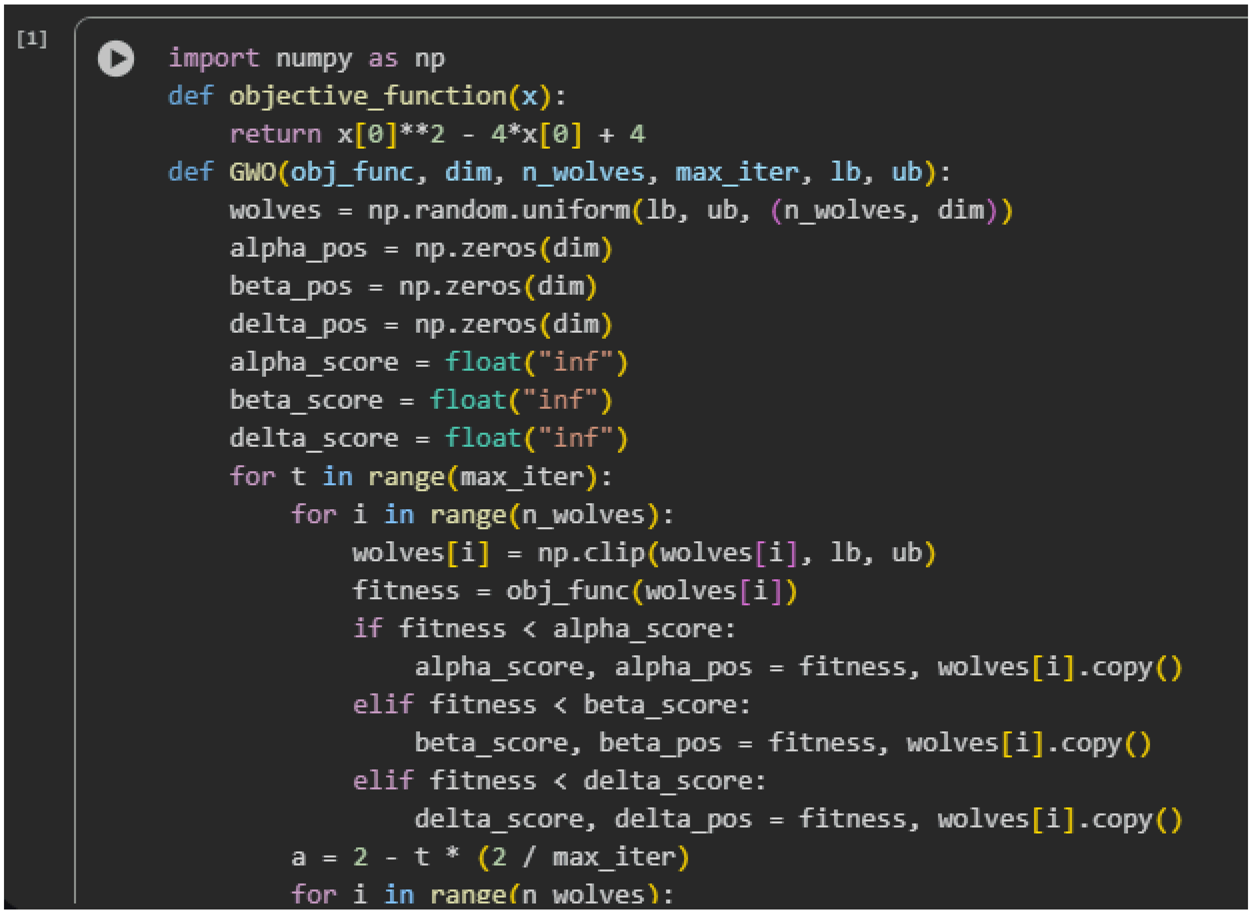
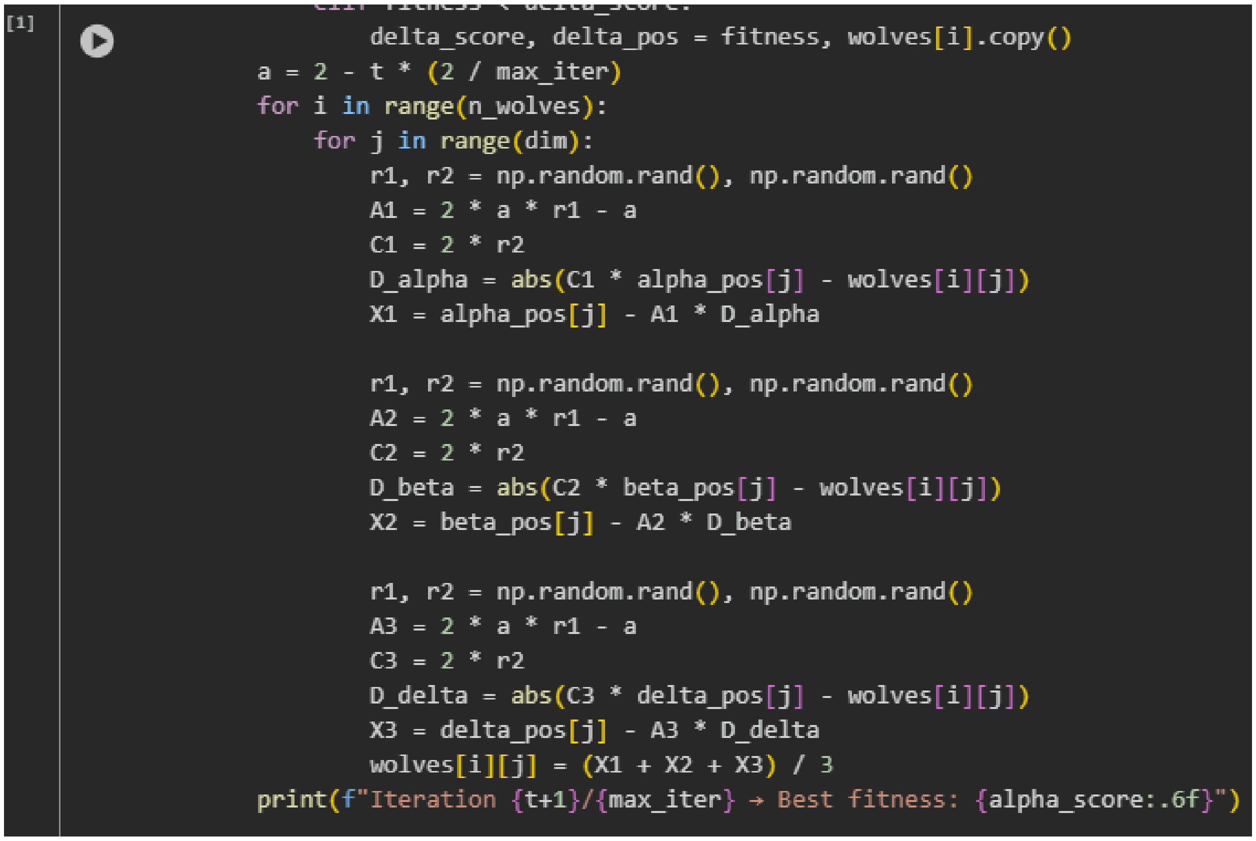
Lab 6
Grey wolf optimizer



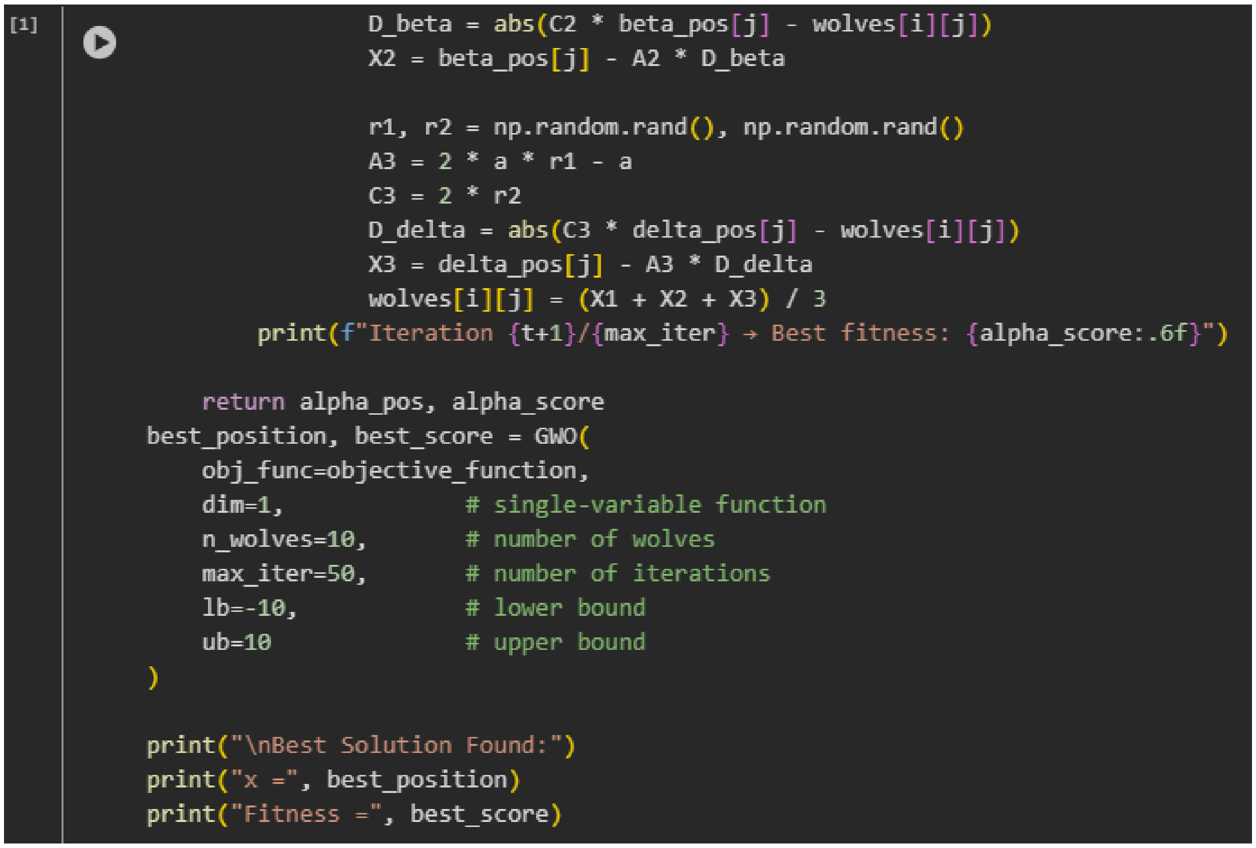
```
[1]
import numpy as np
def objective_function(x):
    return x[0]**2 - 4*x[0] + 4
def GWO(obj_func, dim, n_wolves, max_iter, lb, ub):
    wolves = np.random.uniform(lb, ub, (n_wolves, dim))
    alpha_pos = np.zeros(dim)
    beta_pos = np.zeros(dim)
    delta_pos = np.zeros(dim)
    alpha_score = float("inf")
    beta_score = float("inf")
    delta_score = float("inf")
    for t in range(max_iter):
        for i in range(n_wolves):
            wolves[i] = np.clip(wolves[i], lb, ub)
            fitness = obj_func(wolves[i])
            if fitness < alpha_score:
                alpha_score, alpha_pos = fitness, wolves[i].copy()
            elif fitness < beta_score:
                beta_score, beta_pos = fitness, wolves[i].copy()
            elif fitness < delta_score:
                delta_score, delta_pos = fitness, wolves[i].copy()
        a = 2 - t * (2 / max_iter)
        for i in range(n_wolves):
```



```
[1]
                delta_score, delta_pos = fitness, 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

                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
        print(f"Iteration {t+1}/{max_iter} → Best fitness: {alpha_score:.6f}")
```

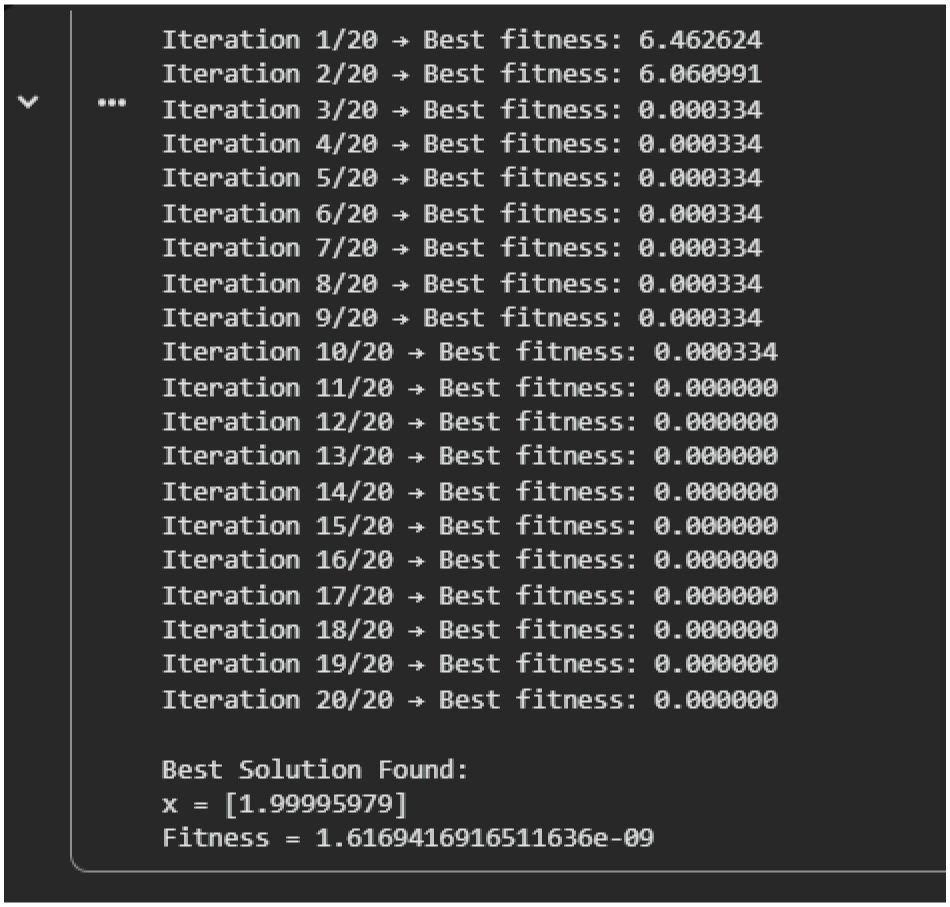


```
[1]
                D_beta = abs(C2 * beta_pos[j] - wolves[i][j])
                X2 = beta_pos[j] - A2 * D_beta

                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
        print(f"Iteration {t+1}/{max_iter} → Best fitness: {alpha_score:.6f}")

    return alpha_pos, alpha_score
best_position, best_score = GWO(
    obj_func=objective_function,
    dim=1,             # single-variable function
    n_wolves=10,       # number of wolves
    max_iter=50,       # number of iterations
    lb=-10,            # lower bound
    ub=10              # upper bound
)

print("\nBest Solution Found:")
print("x =", best_position)
print("Fitness =", best_score)
```



```
Iteration 1/20 → Best fitness: 6.462624
Iteration 2/20 → Best fitness: 6.060991
Iteration 3/20 → Best fitness: 0.000334
Iteration 4/20 → Best fitness: 0.000334
Iteration 5/20 → Best fitness: 0.000334
Iteration 6/20 → Best fitness: 0.000334
Iteration 7/20 → Best fitness: 0.000334
Iteration 8/20 → Best fitness: 0.000334
Iteration 9/20 → Best fitness: 0.000334
Iteration 10/20 → Best fitness: 0.000334
Iteration 11/20 → Best fitness: 0.000000
Iteration 12/20 → Best fitness: 0.000000
Iteration 13/20 → Best fitness: 0.000000
Iteration 14/20 → Best fitness: 0.000000
Iteration 15/20 → Best fitness: 0.000000
Iteration 16/20 → Best fitness: 0.000000
Iteration 17/20 → Best fitness: 0.000000
Iteration 18/20 → Best fitness: 0.000000
Iteration 19/20 → Best fitness: 0.000000
Iteration 20/20 → Best fitness: 0.000000

Best Solution Found:
x = [1.99995979]
Fitness = 1.6169416916511636e-09
```