PRACTICAL: 3

Aim: Let's use it to optimize a simple linear equation with 4 inputs and 1 output.

Y = w1X1 + w2X2 + w3X3 + w4X4

We want to get the values of w1 to w4 to make the following equation hold:

Y = w1(4) + w2(-2) + w3(3.5) + w4(5)

Implement the above simple linear equation using pygad library.

Code:

```
import pygad
import numpy as np
import matplotlib.pyplot as plt


inputs = np.array([4, -2, 3.5, 5])


target_output = 44


def fitness_func(ga_instance, solution, solution_idx):
   output = np.sum(solution * inputs)
   fitness = 1.0 / (abs(output - target_output) + 0.0001)
   return fitness


ga_instance = pygad.GA(
   num_generations=100,        # number of generations
   num_parents_mating=2,       # parents for mating
   fitness_func=fitness_func,  # our custom fitness function
   sol_per_pop=10,             # population size
   num_genes=len(inputs),      # number of weights to optimize
   mutation_probability=0.2,   # chance of mutation
   mutation_type="random",     # mutation operator
   mutation_percent_genes=50   # % of genes to mutate
)
ga_instance.run()
```

```
solution, solution_fitness, solution_idx = ga_instance.best_solution()
print("Best Weights (w1, w2, w3, w4):", solution)
print("Fitness Score:", solution_fitness)

predicted_output = np.sum(solution * inputs)
print("Predicted Output:", predicted_output)
print("Target Output:", target_output)

plt.plot(ga_instance.best_solutions_fitness, label="Best Fitness")
plt.xlabel("Generation")
plt.ylabel("Fitness")
plt.title("Genetic Algorithm Fitness Over Generations")
plt.legend()
plt.grid(True)
plt.show()
```

Output:

```
Best Weights (w1, w2, w3, w4): [ 3.61768589 -1.05601624  1.18796877  4.65280394]
Fitness Score: 208.92268378501126
Predicted Output: 44.00468645966959
Target Output: 44
```