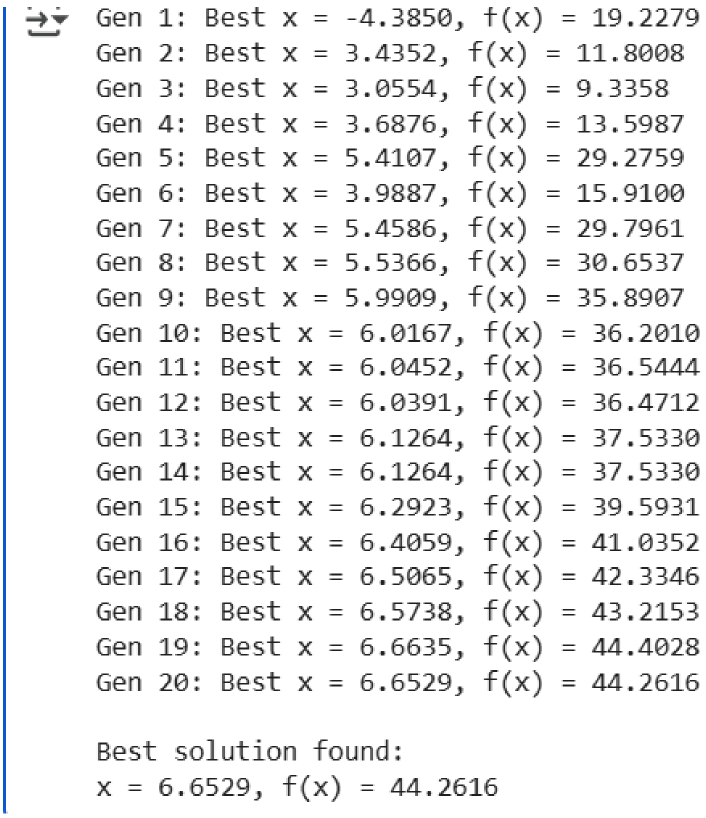
Optimization via Gene Expression Algorithms:

```
import random


def objective_function(x):
    return x ** 2


POP_SIZE = 20
GENS = 20
GENE_LENGTH = 10
CROSSOVER_RATE = 0.8
MUTATION_RATE = 0.1
BOUNDS = [-10, 10]


def create_population():
    return [[random.uniform(BOUNDS[0], BOUNDS[1]) for _ in range(GENE_LENGTH)] for _ in range(POP_SIZE)]


def gene_expression(gene):
    return sum(gene) / len(gene)


def evaluate(population):
    expressed = [gene_expression(g) for g in population]
    return [objective_function(x) for x in expressed], expressed


def select(population, fitness):
    i, j = random.sample(range(len(population)), 2)
    return population[i] if fitness[i] > fitness[j] else population[j]


def crossover(parent1, parent2):
    if random.random() < CROSSOVER_RATE:
        point = random.randint(1, GENE_LENGTH - 1)
        return parent1[:point] + parent2[point:]
    return parent1[:]

def mutate(gene):
    return [g + random.uniform(-1, 1) if random.random() < MUTATION_RATE else g for g in gene]

def gene_expression_algorithm():
    population = create_population()
    for gen in range(GENS):
        fitness, expressed = evaluate(population)
        new_population = []
        for _ in range(POP_SIZE):
            parent1 = select(population, fitness)
            parent2 = select(population, fitness)
            child = crossover(parent1, parent2)
            child = mutate(child)
            new_population.append(child)
        population = new_population
        best_idx = fitness.index(max(fitness))
        best_x = expressed[best_idx]
        best_fit = fitness[best_idx]
        print(f"Gen {gen+1}: Best x = {best_x:.4f}, f(x) = {best_fit:.4f}")
    return best_x, best_fit

best_x, best_val = gene_expression_algorithm()
print("\nBest solution found:")
print(f"x = {best_x:.4f}, f(x) = {best_val:.4f}")
```

OUTPUT:



```
Gen 1: Best x = -4.3850, f(x) = 19.2279
Gen 2: Best x = 3.4352, f(x) = 11.8008
Gen 3: Best x = 3.0554, f(x) = 9.3358
Gen 4: Best x = 3.6876, f(x) = 13.5987
Gen 5: Best x = 5.4107, f(x) = 29.2759
Gen 6: Best x = 3.9887, f(x) = 15.9100
Gen 7: Best x = 5.4586, f(x) = 29.7961
Gen 8: Best x = 5.5366, f(x) = 30.6537
Gen 9: Best x = 5.9909, f(x) = 35.8907
Gen 10: Best x = 6.0167, f(x) = 36.2010
Gen 11: Best x = 6.0452, f(x) = 36.5444
Gen 12: Best x = 6.0391, f(x) = 36.4712
Gen 13: Best x = 6.1264, f(x) = 37.5330
Gen 14: Best x = 6.1264, f(x) = 37.5330
Gen 15: Best x = 6.2923, f(x) = 39.5931
Gen 16: Best x = 6.4059, f(x) = 41.0352
Gen 17: Best x = 6.5065, f(x) = 42.3346
Gen 18: Best x = 6.5738, f(x) = 43.2153
Gen 19: Best x = 6.6635, f(x) = 44.4028
Gen 20: Best x = 6.6529, f(x) = 44.2616

Best solution found:
x = 6.6529, f(x) = 44.2616
```