Gene expression

```
import random

# Parameters
POP_SIZE = 10
GENE_LENGTH = 5         # bits per gene
CHROM_LENGTH = GENE_LENGTH * 2  # total bits (2 genes: x and y)
MAX_GEN = 30
CROSSOVER_RATE = 0.7
MUTATION_RATE = 0.01

# Gene Expression: binary chromosome -> phenotype (x, y)
def gene_expression(chromosome):
    x_bits = chromosome[:GENE_LENGTH]
    y_bits = chromosome[GENE_LENGTH:]
    x = int(x_bits, 2)
    y = int(y_bits, 2)
    return x, y

# Fitness Function: f(x, y) = x^2 + y^2
def fitness(chromosome):
    x, y = gene_expression(chromosome)
    return x**2 + y**2

# Initialize population: list of random chromosomes
def init_population():
    population = []
    for _ in range(POP_SIZE):
        chrom = ''.join(random.choice('01') for _ in range(CHROM_LENGTH))
        population.append(chrom)
    return population

# Roulette wheel selection
def select(population):
    total_fitness = sum(fitness(ch) for ch in population)
    pick = random.uniform(0, total_fitness)
    current = 0
    for ch in population:
        current += fitness(ch)
        if current >= pick:
            return ch
    return population[-1]
```

```
# Single-point crossover
def crossover(parent1, parent2):
    if random.random() < CROSSOVER_RATE:
        point = random.randint(1, CHROM_LENGTH - 1)
        child1 = parent1[:point] + parent2[point:]
        child2 = parent2[:point] + parent1[point:]
        return child1, child2
    else:
        return parent1, parent2

# Bit-flip mutation
def mutate(chromosome):
    chrom_list = list(chromosome)
    for i in range(CHROM_LENGTH):
        if random.random() < MUTATION_RATE:
            chrom_list[i] = '1' if chrom_list[i] == '0' else '0'
    return ''.join(chrom_list)

# Main GA loop
def genetic_algorithm():
    population = init_population()

    for gen in range(1, MAX_GEN + 1):
        new_population = []

        while len(new_population) < POP_SIZE:
            # Selection
            parent1 = select(population)
            parent2 = select(population)

            # Crossover
            child1, child2 = crossover(parent1, parent2)

            # Mutation
            child1 = mutate(child1)
            child2 = mutate(child2)

            new_population.extend([child1, child2])

        # Keep population size constant
        population = new_population[:POP_SIZE]

        # Find the best chromosome in the generation
        best_chrom = max(population, key=fitness)
```

```
        best_x, best_y = gene_expression(best_chrom)
        best_fit = fitness(best_chrom)

        print(f"Gen {gen:02d} | Best Chromosome: {best_chrom} | x = {best_x}, y = {best_y} |
Fitness = {best_fit}")

    # Final best solution
    best_chrom = max(population, key=fitness)
    best_x, best_y = gene_expression(best_chrom)
    best_fit = fitness(best_chrom)
    print("\nBest solution found:")
    print(f"Chromosome: {best_chrom}")
    print(f"x = {best_x}, y = {best_y}")
    print(f"Fitness = {best_fit}")

if __name__ == "__main__":
    genetic_algorithm()
```

Output:

```
Gen 01 | Best Chromosome: 1101011111 | x = 26, y = 31 | Fitness = 1637
Gen 02 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 03 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 04 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 05 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 06 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 07 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 08 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 09 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 10 | Best Chromosome: 1111111111 | x = 31, y = 31 | Fitness = 1922
Gen 11 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 12 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 13 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 14 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 15 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 16 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 17 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 18 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 19 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 20 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 21 | Best Chromosome: 1111111111 | x = 31, y = 31 | Fitness = 1922
Gen 22 | Best Chromosome: 1111111111 | x = 31, y = 31 | Fitness = 1922
Gen 23 | Best Chromosome: 1111111111 | x = 31, y = 31 | Fitness = 1922
Gen 24 | Best Chromosome: 1111111111 | x = 31, y = 31 | Fitness = 1922
Gen 25 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 26 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 27 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 28 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 29 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861
Gen 30 | Best Chromosome: 1111011111 | x = 30, y = 31 | Fitness = 1861

Best solution found:
Chromosome: 1111011111
x = 30, y = 31
```