import random

num\_genotypes = 4

genotype\_length = 10

def fitness\_function(genotype):

return sum(genotype)

population = [[random.randint(0, 1) for \_ in range(genotype\_length)] for \_ in range(num\_genotypes)]

def crossover(parent1, parent2):

crossover\_point = random.randint(1, genotype\_length - 1)

child1 = parent1[:crossover\_point] + parent2[crossover\_point:]

child2 = parent2[:crossover\_point] + parent1[crossover\_point:]

return child1, child2

def mutate(genotype, mutation\_rate):

mutated\_genotype = [bit ^ (random.random() < mutation\_rate) for bit in genotype]

return mutated\_genotype

mutation\_rate = 0.1

fitness\_values = [fitness\_function(genotype) for genotype in population]

best\_genotype\_index = max(range(num\_genotypes), key=lambda i: fitness\_values[i])

parent1\_index, parent2\_index = random.sample(range(num\_genotypes), 2)

child1, child2 = crossover(population[parent1\_index], population[parent2\_index])

mutated\_genotype\_index = random.randint(0, num\_genotypes - 1)

mutated\_genotype = mutate(population[mutated\_genotype\_index], mutation\_rate)

child1\_fitness = fitness\_function(child1)

child2\_fitness = fitness\_function(child2)

mutated\_genotype\_fitness = fitness\_function(mutated\_genotype)

print(f"Fitness of Child 1: {child1\_fitness}")

print(f"Fitness of Child 2: {child2\_fitness}")

print(f"Fitness of Mutated Genotype: {mutated\_genotype\_fitness}")

print(f"Fitness of Best Genotype in the Initial Population: {fitness\_values[best\_genotype\_index]}")