

➤ **Parameter settings**

```
gene_length = 30  #二進制的長度
population_size= 500  #每一代染色體的數量
mutation_rate = 0.5  #變異機率
tolerance = 1e-11  # 停止條件
aus = 1e6
target = float(input("請輸入您要計算的根號值："))
```

➤ **def Binary(chromosome)**

這個部分主要是將我的數值轉為二進制的定義

➤ **def decimal_number(chromosome)**

這個部分主要是將我的數值轉為十進制的定義

➤ **def generate_population(population_size, gene_length)**

生成初始族群的函數

➤ **def fitness_function(individual)**

```
# 評估染色體適應度的函數
    chromosome = decimal_number(individual) / (10**11)
    score = aus - abs(chromosome**2 - target)
    return max(0, score)  #確保適應度值大於等於零
```

def crossover(parent1, parent2)

```
# 交叉遺傳的函數
    # 隨機選擇交叉點，避免在邊界處交叉
    point = random.randint(1, len(parent1) - 2)
    # 生成子代 1，將父母 1 的前段和父母 2 的後段結合
    child1 = parent1[:point] + parent2[point:]
    # 生成子代 2，將父母 2 的前段和父母 1 的後段結合
    child2 = parent2[:point] + parent1[point:]
    return child1, child2
```

➤ **def mutate(individual)**

變異的函數

隨機選擇染色體中的一個位置

```
mutated_index = random.randint(0, len(individual) - 1)
```

將選定位置的基因取反，即 0 變為 1，1 變為 0

```
mutated_gene = '0' if individual[mutated_index] == '1' else '1'
```

產生變異後的染色體，將選定位置的基因替換為變異後的基因

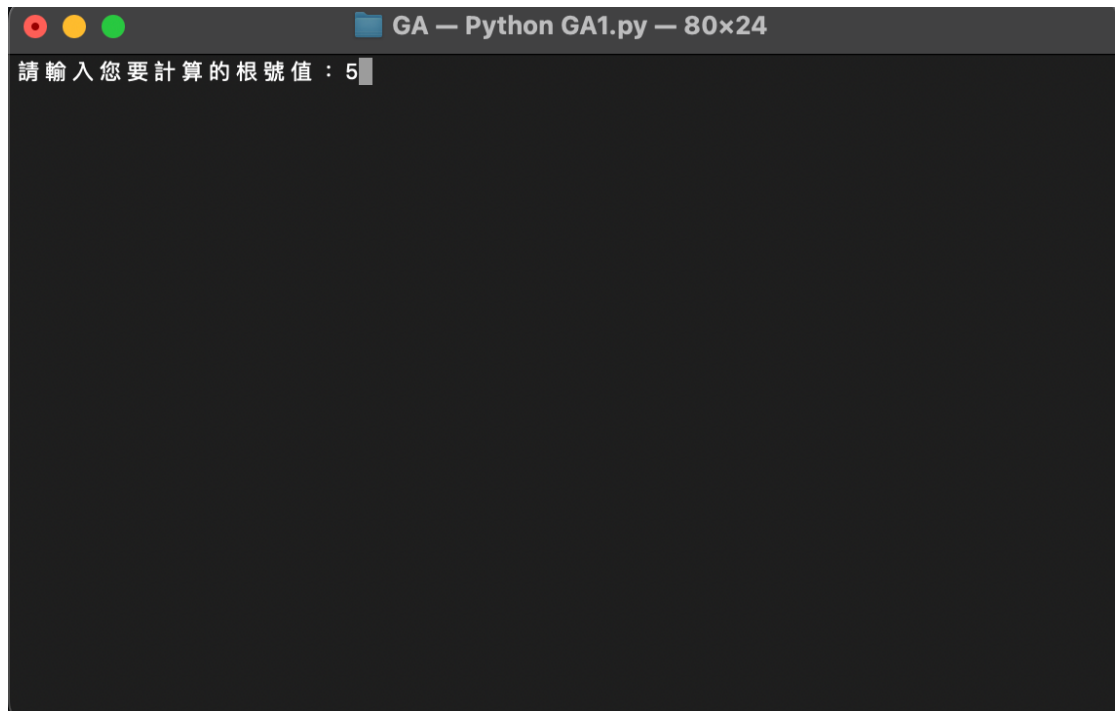
```
return individual[:mutated_index] + mutated_gene +
```

```
individual[mutated_index + 1:]
```

➤ **def genetic_algorithm()**

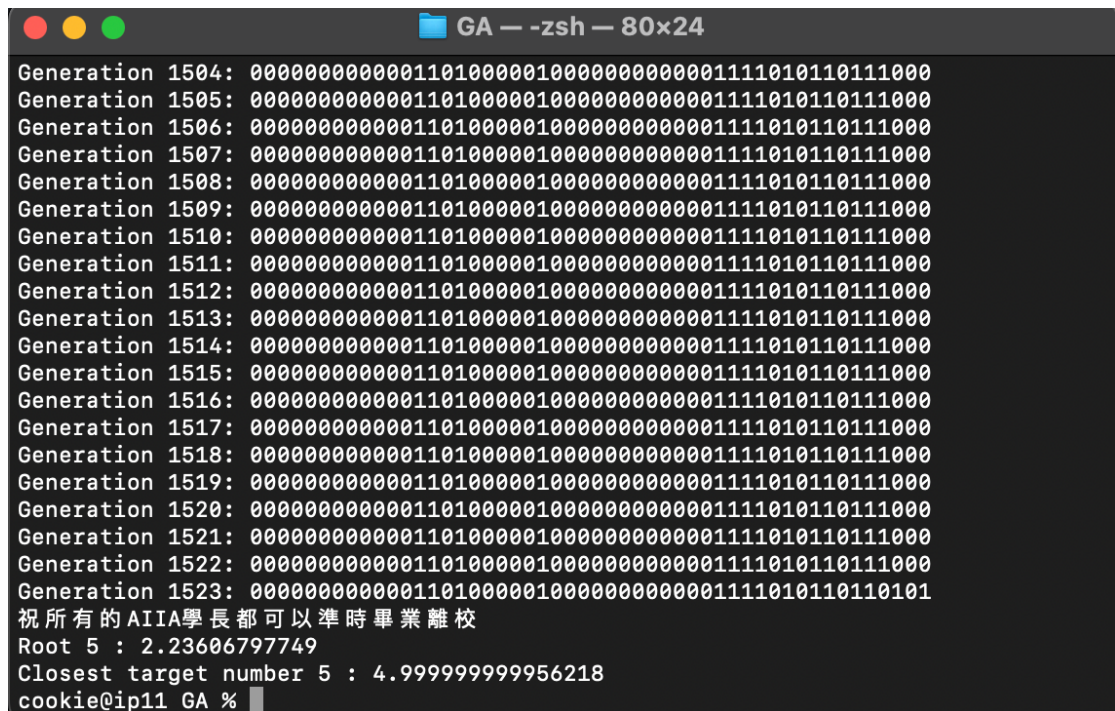
遺傳演算法主要的函數，會先生成一個族群，然後將族群按照適應的程度去排序，產生最佳的染色體，這邊也會去選擇父母的染色體，進行交叉遺傳，然後變異，將好的子# 檢查是否達到停止條件，子代留下來，產生新的族群

(以下為執行結果)



```
GA — Python GA1.py — 80x24
請輸入您要計算的根號值：5
```

(圖一) 可以輸入需要計算的根號數字



```
GA — -zsh — 80x24
Generation 1504: 00000000000011010000010000000000001111010110111000
Generation 1505: 00000000000011010000010000000000001111010110111000
Generation 1506: 00000000000011010000010000000000001111010110111000
Generation 1507: 00000000000011010000010000000000001111010110111000
Generation 1508: 00000000000011010000010000000000001111010110111000
Generation 1509: 00000000000011010000010000000000001111010110111000
Generation 1510: 00000000000011010000010000000000001111010110111000
Generation 1511: 00000000000011010000010000000000001111010110111000
Generation 1512: 00000000000011010000010000000000001111010110111000
Generation 1513: 00000000000011010000010000000000001111010110111000
Generation 1514: 00000000000011010000010000000000001111010110111000
Generation 1515: 00000000000011010000010000000000001111010110111000
Generation 1516: 00000000000011010000010000000000001111010110111000
Generation 1517: 00000000000011010000010000000000001111010110111000
Generation 1518: 00000000000011010000010000000000001111010110111000
Generation 1519: 00000000000011010000010000000000001111010110111000
Generation 1520: 00000000000011010000010000000000001111010110111000
Generation 1521: 00000000000011010000010000000000001111010110111000
Generation 1522: 00000000000011010000010000000000001111010110111000
Generation 1523: 00000000000011010000010000000000001111010110110101
祝所有的AIIA學長都可以準時畢業離校
Root 5 : 2.23606797749
Closest target number 5 : 4.999999999956218
cookie@ip11 GA %
```

(圖二) 為輸入(根號)5 所得到最終的結果

(以下為程式碼)

```
1 import random
2
3
4 gene_length = 50 #binary length
5 population_size= 1000 #The number of chromosomes in each generation
6 mutation_rate = 0.4 #Mutation probability
7 tolerance = 1e-11 # Stop condition
8 aus = 1e6
9 # Enter the target number to be calculated
10 target = float(input("請輸入您要計算的根號值:"))
11
12 def Binary(chromosome):
13     #Convert to binary(str)
14     binary_string=bin(chromosome)[2:]
15     return binary_string
16
17 def decimal_number(chromosome):
18     #轉乘10進制(str)
19     return int(chromosome,2)
20
21 def generate_population(population_size,gene_length):
22     return[''.join(random.choice('01') for _ in range(gene_length)) for _ in range(population_size)]
23
24
25 def fitness_function(individual):
26     chromosome = decimal_number(individual) / (10**11)
27     score = aus - abs(chromosome**2 - target)
28     return max(0, score) # 確保適應度值大於等於零
29
30 def select_parents(population):
31     #選擇2個染色體(當父母)
32     #population為這一代中所有的父母
33     return random.choices(population, k=2, weights=[fitness_function(ind) for ind in population])
34
35 def crossover(parent1, parent2):
36     #單點交叉(交配)
37     #1, len(parent1) - 2)是為了避免選到投跟尾
38     point = random.randint(1, len(parent1) - 2)
39     child1 = parent1[:point] + parent2[point:]
40     child2 = parent2[:point] + parent1[point:]
41     return child1, child2
42
43 def mutate(individual):
44     #選擇一個點進行變異
45     #使1-->0 和 0-->1
46     mutated_index = random.randint(0, len(individual) - 1)
47     mutated_gene = '0' if individual[mutated_index] == '1' else '1'
48     return individual[:mutated_index] + mutated_gene + individual[mutated_index+1:]
49
50 def genetic_algorithm():
51     #Randomly generate the first generation chromosomes
52     population=generate_population(population_size,gene_length)
53     generation=0
54
55     while True:
56         generation+=1
57         #Rank dyeing replacements with high scores ranked first
58         population=sorted(population,key=fitness_function,reverse=True)
59         new_population=[population[0]]
60         #Generate the next generation (population_size chromosomes)
61         while len(new_population)<population_size:
62             #Choose parents to have children
63             parent1,parent2=select_parents(population)
64             child1,child2=crossover(parent1,parent2)
65
66             #Mutations
67             if random.random()<mutation_rate:
68                 child1=mutate(child1)
69             if random.random()<mutation_rate:
70                 child2=mutate(child2)
71
72             new_population.extend([child1,child2])
73
74         population=new_population
75
76         #The best solution of this generation
77         ans=population[0]
78         a=fitness_function(ans)
79         print(f"Generation {generation}: {ans}")
80         # Check whether the stopping condition is met
81         if -tolerance < aus - a < tolerance:
82             break
83         # print(f"100 - a: {100 - a}")
84         # print(f"a - target = {a - target}")
85
86         print('祝所有的AIIA學長都可以準時畢業離校')
87         print(f'Root {int(target)} :', decimal_number(ans) / 10**11)
88         print(f'Closest target number {int(target)} :', (decimal_number(ans) / 10**11)**2)
89
90 if __name__ == '__main__':
91     genetic_algorithm()
92
93
94
95 # ans = # 2.23606797749979
```