```
# Γενετικός Αλγόριθμος για την επίλυση
# του προβλήματος των 8 Βασιλισσών (version 2)
# Python v3.6.3
# 10/1/18, 30/10/16, Κ.Σγάρμπας

import random

def mutation(gene, prob=0.001): # Αλλάζει ένα σύμβολο με πιθανότητα prob
    s=""
    for c in gene:
        if random.random()>prob: s+=c
        else: s+=random.choice("12345678")
    return s

def crossover(gene1, gene2):
    split=random.randint(1,len(gene1)-1) # το σημείο του split
    newgene1=gene1[:split]+gene2[split:]
    newgene2=gene2[:split]+gene1[split:]
    return newgene1, newgene2

def selection(pool, addprobs): # επιστρέφει ένα τυχαίο gene, ανάλογα με την
πιθανότητά του
    c=random.random()
    i=0
    while c>addprobs[i]: i+=1
    return pool[i]

def randomgene(): # δημιουργεί ένα τυχαίο gene
    g=""
    for i in range(8):
        g+=random.choice("12345678")
    return g

def initialize(size=40): # δημιουργεί τον αρχικό τυχαίο πληθυσμό
    L=[]
    for i in range(size):
        L.append(randomgene())
    return L

def evaluation(x): # υπολογίζει το πλήθος των απειλών στο gene x
    y=[int(i) for i in list(x)]
    L=len(y)
    att=0
    for i1 in range(L-1):
        for i2 in range(i1+1,L):
            q1,q2=y[i1],y[i2]
            if q1==q2: att+=1           # στην ίδια γραμμή
            elif abs(q1-q2)==abs(i1-i2): att+=1     # διαγώνια
    return att

def monitor(PS,generation,population,eva_values): # τυπώνει στατιστικά πληθυσμού
    Seva=sum(eva_values)
    print("gener.",generation,"Av.eva=%.2f" %(Seva/PS), end=" ")
    print("(min/max)=("+str(min(eva_values))+"/"+str(max(eva_values)), end=") ")
    sample=random.choice(population)
    print("Sample=", sample, "eval=",evaluation(sample), "diff.genes=", end=" ")
    print(len(set(population)), "sum=",Seva)

def update_population(PS,population,fit_values): # δημιουργεί τη νέα γενιά
    Sfit=sum(fit_values)
    probs=[x/Sfit for x in fit_values]  # πίνακας πιθανοτήτων
    addprobs=[]                         # πίνακας αθροιστικών πιθανοτήτων
    h=0
    for x in probs:
       h+=x
       addprobs.append(h)
    newpopulation=[]
    for i in range(int(PS/2)):
        g1=selection(population, addprobs)
        g2=selection(population, addprobs)
        g1,g2=crossover(g1,g2)
        g1,g2=mutation(g1),mutation(g2)
        newpopulation.append(g1)
        newpopulation.append(g2)
    return newpopulation

PS=500 # Population Size
population=initialize(PS)
generation=0
while True:
    eva_values=[evaluation(x) for x in population]
    monitor(PS,generation,population,eva_values)  # τυπώνει στατιστικά
    fit_values=[28-x for x in eva_values] # linear
#    fit_values=[1/(x+0.1) for x in eva_values] # enhanced (βελτιωμένη σύγκλιση)
    population=update_population(PS,population,fit_values)
    generation+=1

### Monte Carlo
##g=randomgene()
##bestf=evaluation(g)
##i=0
##print(i,g,bestf)
##while True:
##    g=randomgene()
##    f=evaluation(g)
##    i+=1
##    if f<bestf:
##        print(i,g,f)
##        bestf=f

# Λύσεις: 42861357, 72418536, κλπ.
```