Tugas 4

Kelompok 5

Kelompok 5

FIhriz Ilham Rabbany

5025211040

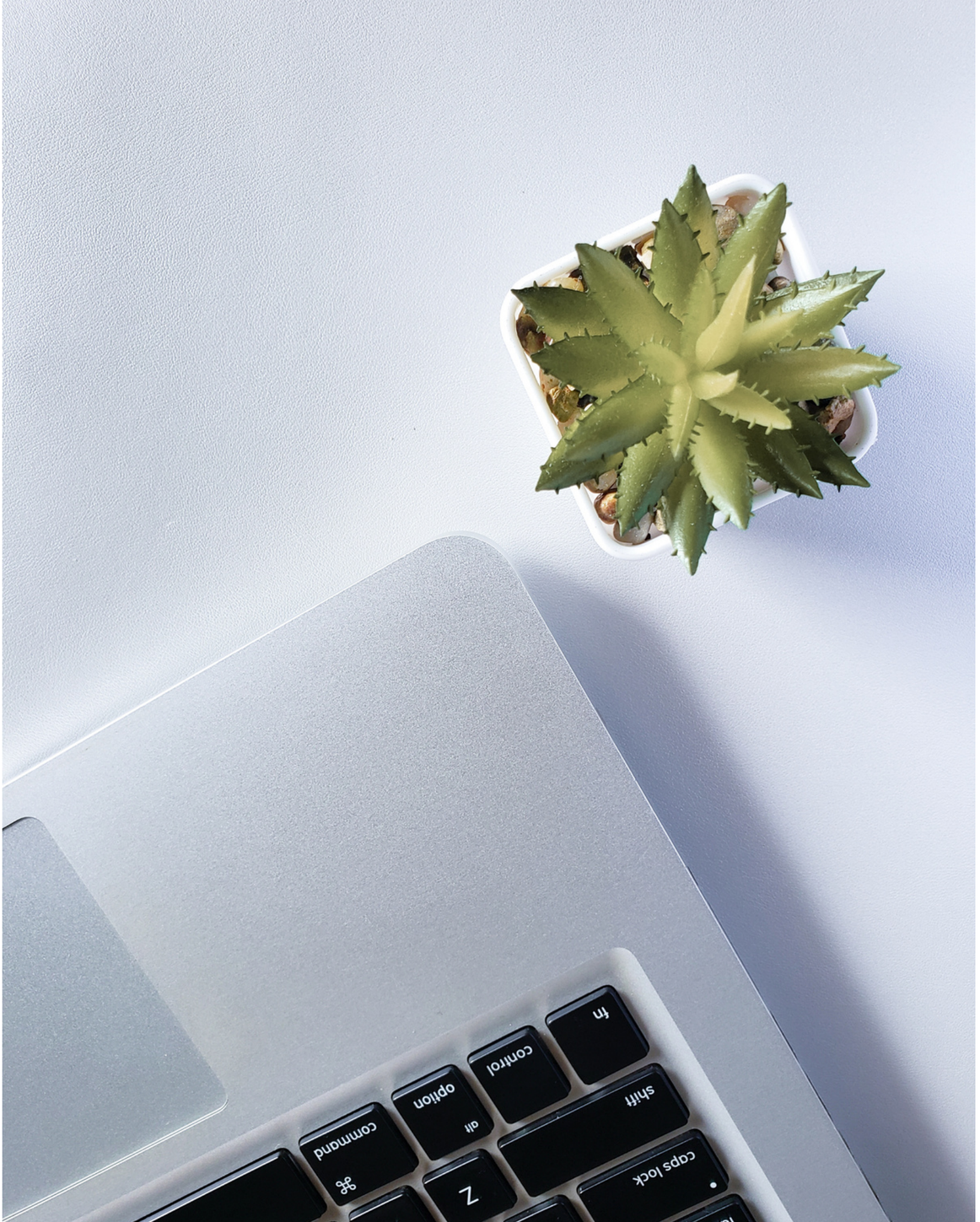
Syomeron Ansell Widjaya

5025211250

Muhammad Ahyun Irsyad

5025211251

8-QUEENS PROBLEM DENGAN MENGGUNAKAN GA

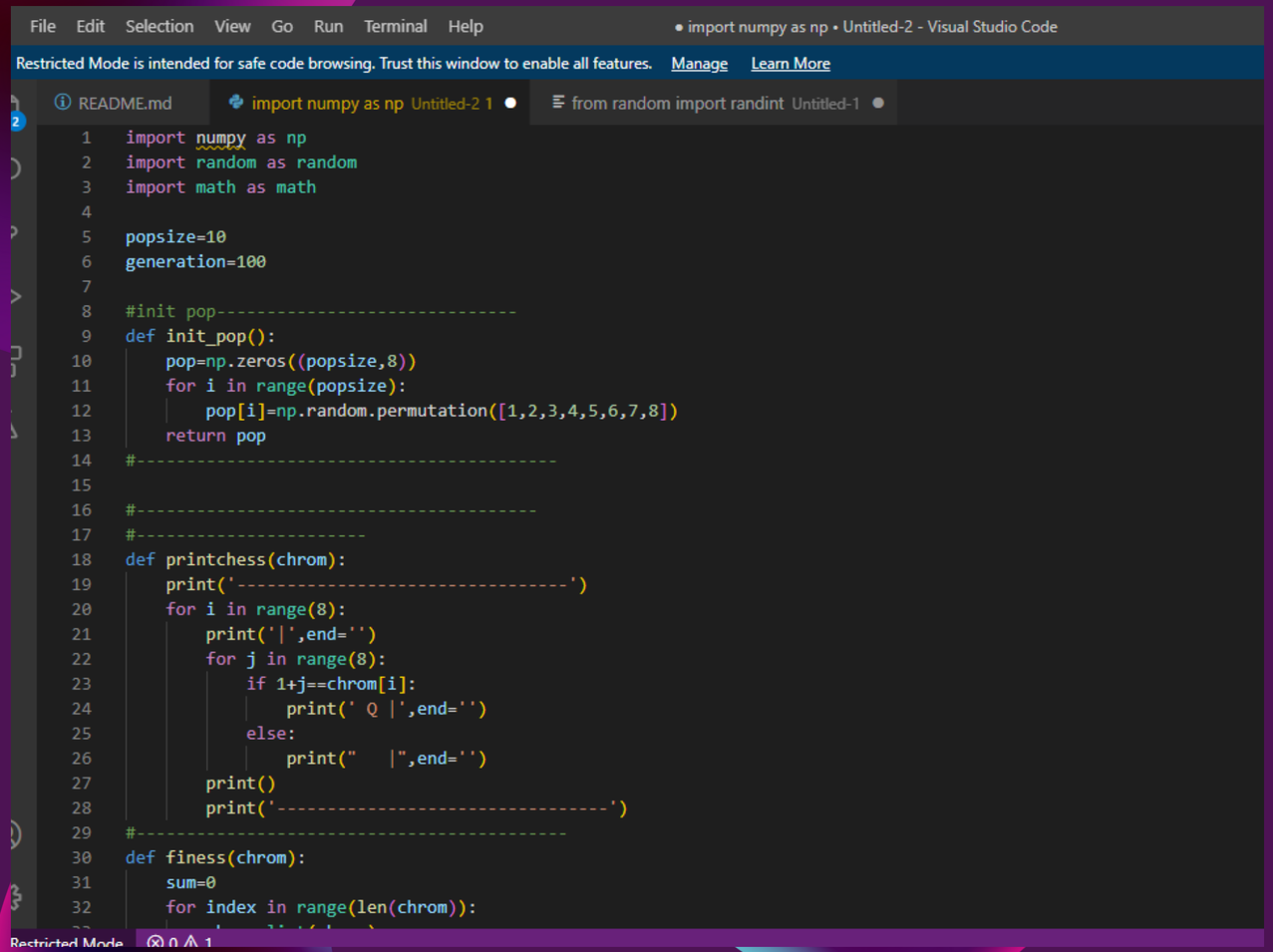


GENETIC ALGORITHM

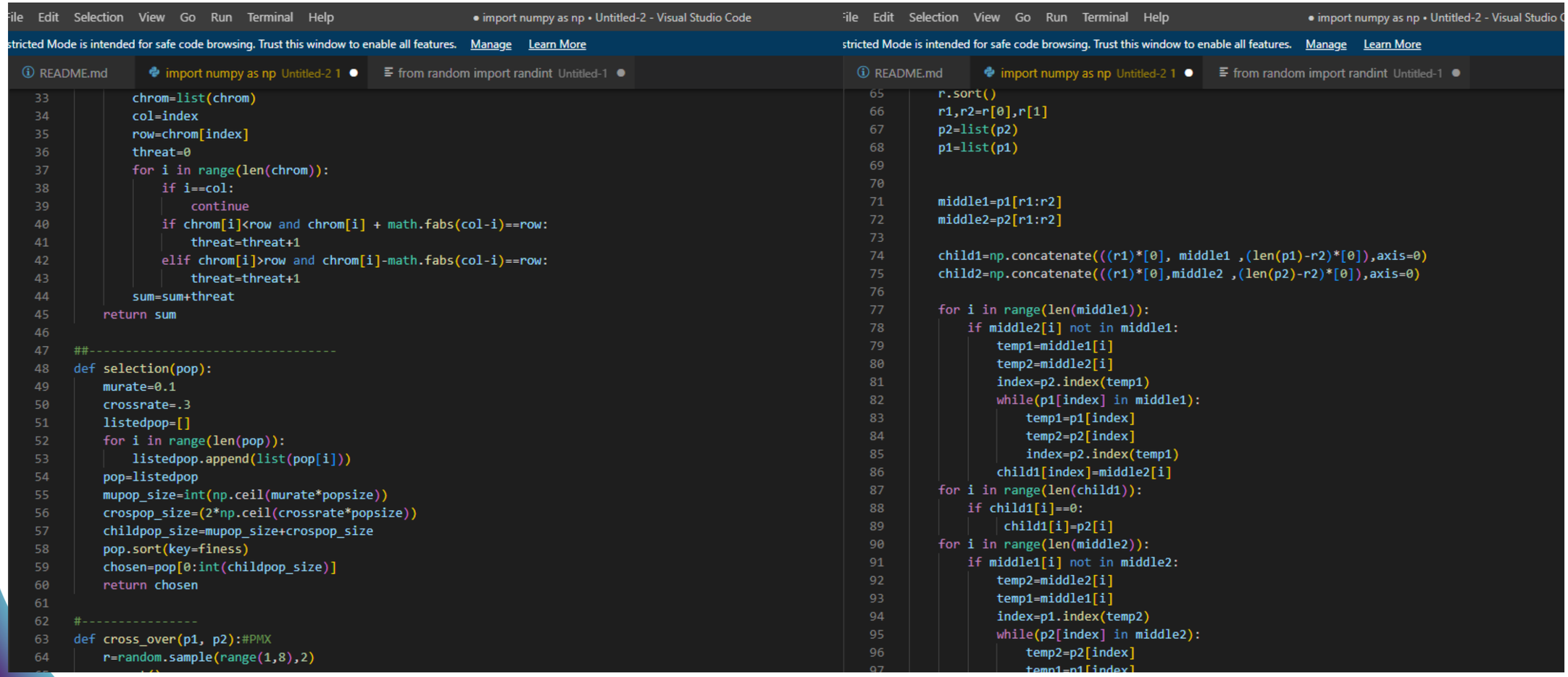
Genetic Algorithm (GA) adalah bagian dari Evolutionary Algorithm yaitu suatu algoritma yang mencontoh proses evolusi alami dimana konsep utamanya adalah individu-individu yang paling unggul akan bertahan hidup, sedangkan individu-individu yang lemah akan punah. Keunggulan individu-individu ini diuji melalui suatu fungsi yang dikenal sebagai fitness function. Fitness dalam GA didefinisikan sebagai gambaran kelayakan suatu solusi terhadap suatu permasalahan. Fitness Function akan menghasilkan suatu nilai fitness value yang akan menjadi referensi untuk proses GA selanjutnya.

02

Source Code



Source Code



File Edit Selection View Go Run Terminal Help • import numpy as np • Untitled-2 - Visual Studio Code

stricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

README.md | import numpy as np Untitled-2 1 | from random import randint Untitled-1

```
            chrom=list(chrom)
            col=index
            row=chrom[index]
            threat=0
            for i in range(len(chrom)):
                if i==col:
                    continue
                if chrom[i]<row and chrom[i] + math.fabs(col-i)==row:
                    threat=threat+1
                elif chrom[i]>row and chrom[i]-math.fabs(col-i)==row:
                    threat=threat+1
            sum=sum+threat
        return sum

    ##---------------------------------
    def selection(pop):
        murate=0.1
        crossrate=.3
        listedpop=[]
        for i in range(len(pop)):
            listedpop.append(list(pop[i]))
        pop=listedpop
        mupop_size=int(np.ceil(murate*popsize))
        crospop_size=(2*np.ceil(crossrate*popsize))
        childpop_size=mupop_size+crospop_size
        pop.sort(key=finess)
        chosen=pop[0:int(childpop_size)]
        return chosen

    #---------------
    def cross_over(p1, p2):#PMX
        r=random.sample(range(1,8),2)
```

File Edit Selection View Go Run Terminal Help • import numpy as np • Untitled-2 - Visual Studio C

stricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

README.md | import numpy as np Untitled-2 1 | from random import randint Untitled-1

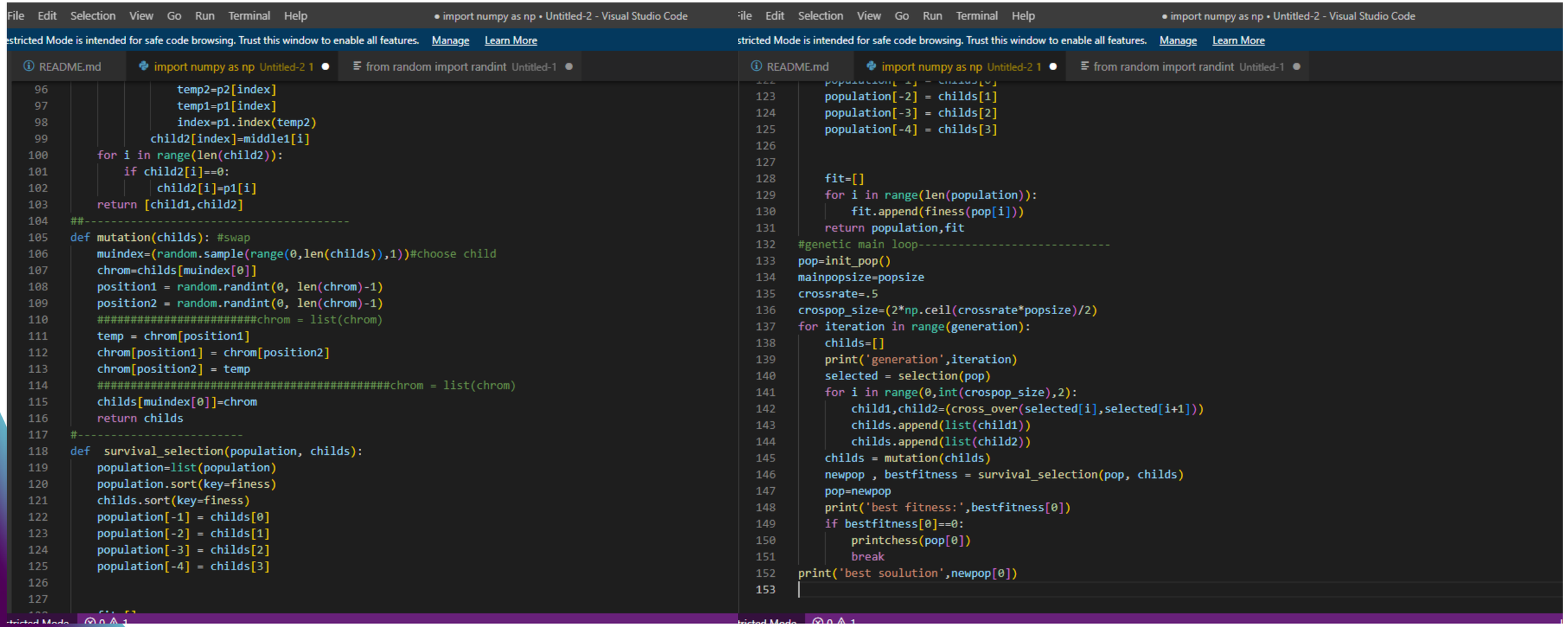
```
        r.sort()
        r1,r2=r[0],r[1]
        p2=list(p2)
        p1=list(p1)


        middle1=p1[r1:r2]
        middle2=p2[r1:r2]

        child1=np.concatenate(((r1)*[0], middle1 ,(len(p1)-r2)*[0]),axis=0)
        child2=np.concatenate(((r1)*[0],middle2 ,(len(p2)-r2)*[0]),axis=0)

        for i in range(len(middle1)):
            if middle2[i] not in middle1:
                temp1=middle1[i]
                temp2=middle2[i]
                index=p2.index(temp1)
                while(p1[index] in middle1):
                    temp1=p1[index]
                    temp2=p2[index]
                    index=p2.index(temp1)
                child1[index]=middle2[i]
        for i in range(len(child1)):
            if child1[i]==0:
                child1[i]=p2[i]
        for i in range(len(middle2)):
            if middle1[i] not in middle2:
                temp2=middle2[i]
                temp1=middle1[i]
                index=p1.index(temp2)
                while(p2[index] in middle2):
                    temp2=p2[index]
```

Source Code


File Edit Selection View Go Run Terminal Help
• import numpy as np • Untitled-2 - Visual Studio Code
estricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
README.md
import numpy as np Untitled-2 1
from random import randint Untitled-1
temp2=p2[index]
temp1=p1[index]
index=p1.index(temp2)
child2[index]=middle1[i]
for i in range(len(child2)):
if child2[i]==0:
child2[i]=p1[i]
return [child1,child2]
##---------------------------------------
def mutation(childs): #swap
muindex=(random.sample(range(0,len(childs)),1))#choose child
chrom=childs[muindex[0]]
position1 = random.randint(0, len(chrom)-1)
position2 = random.randint(0, len(chrom)-1)
########################chrom = list(chrom)
temp = chrom[position1]
chrom[position1] = chrom[position2]
chrom[position2] = temp
##chrom = list(chrom)
childs[muindex[0]]=chrom
return childs
#------------------------
def survival_selection(population, childs):
population=list(population)
population.sort(key=finess)
childs.sort(key=finess)
population[-1] = childs[0]
population[-2] = childs[1]
population[-3] = childs[2]
population[-4] = childs[3]
File Edit Selection View Go Run Terminal Help
• import numpy as np • Untitled-2 - Visual Studio Code
stricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
README.md
import numpy as np Untitled-2 1
from random import randint Untitled-1
population[-2] = childs[1]
population[-3] = childs[2]
population[-4] = childs[3]
fit=[]
for i in range(len(population)):
fit.append(finess(pop[i]))
return population,fit
#genetic main loop---------------------------
pop=init_pop()
mainpopsize=popsize
crossrate=.5
crospop_size=(2*np.ceil(crossrate*popsize)/2)
for iteration in range(generation):
childs=[]
print('generation',iteration)
selected = selection(pop)
for i in range(0,int(crospop_size),2):
child1,child2=(cross_over(selected[i],selected[i+1]))
childs.append(list(child1))
childs.append(list(child2))
childs = mutation(childs)
newpop , bestfitness = survival_selection(pop, childs)
pop=newpop
print('best fitness:',bestfitness[0])
if bestfitness[0]==0:
printchess(pop[0])
break
print('best soulution',newpop[0])

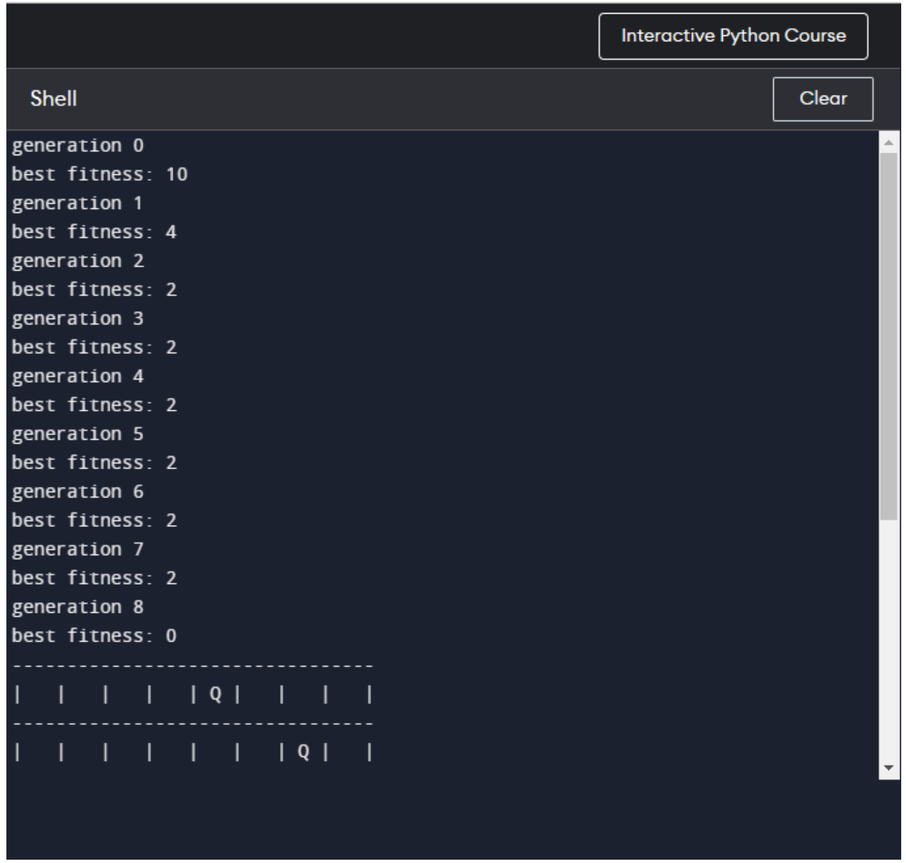
Penjelasan

Pada awalnya, populasi awal dibuat dengan memilih secara acak N angka dari 1 sampai N dan menempatkannya pada setiap baris di papan catur. Kemudian, untuk setiap individu pada populasi, fitness-nya dihitung berdasarkan jumlah ancaman yang dialami oleh setiap ratu. Jika ada dua ratu yang saling menyerang, maka nilai fitness-nya akan lebih besar.

Selanjutnya, pada setiap iterasi, individu pada populasi dipilih berdasarkan nilai fitness-nya, kemudian dilakukan operasi crossover dan mutasi untuk menghasilkan anak baru. Setelah itu, dilakukan seleksi untuk menentukan individu mana yang akan ditambahkan ke populasi berikutnya. Proses iterasi tersebut akan terus dilakukan hingga ditemukan solusi yang memenuhi kriteria atau telah mencapai jumlah iterasi yang telah ditentukan sebelumnya.

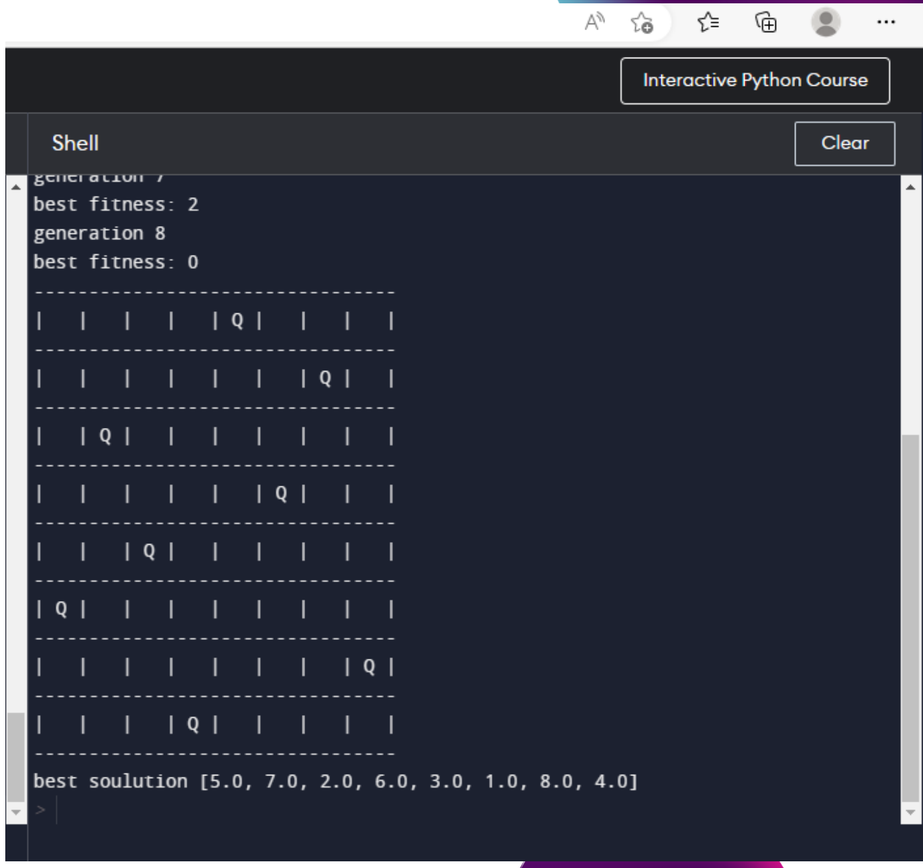
Dalam kode ini, operasi crossover yang digunakan adalah Partially Matched Crossover (PMX), sedangkan operasi mutasi yang digunakan adalah Swap. Seleksi orang tua dilakukan dengan metode elitism, yaitu memilih individu dengan nilai fitness terbaik dari populasi saat ini.

OUTPUT


Interactive Python Course
Shell
Clear
generation 0
best fitness: 10
generation 1
best fitness: 4
generation 2
best fitness: 2
generation 3
best fitness: 2
generation 4
best fitness: 2
generation 5
best fitness: 2
generation 6
best fitness: 2
generation 7
best fitness: 2
generation 8
best fitness: 0

| | | | | | Q | | | |

| | | | | | | | Q | |


Interactive Python Course
Shell
Clear
best fitness: 2
generation 8
best fitness: 0

| | | | | | Q | | | |

| | | | | | | | Q | |

| | | Q | | | | | | |

| | | | | | Q | | | |

| | Q | | | | | | | |

| Q | | | | | | | | |

| | | | | | | | | Q |

| | | | | Q | | | | |
----
best soulution [5.0, 7.0, 2.0, 6.0, 3.0, 1.0, 8.0, 4.0]

TERIMAKASIH