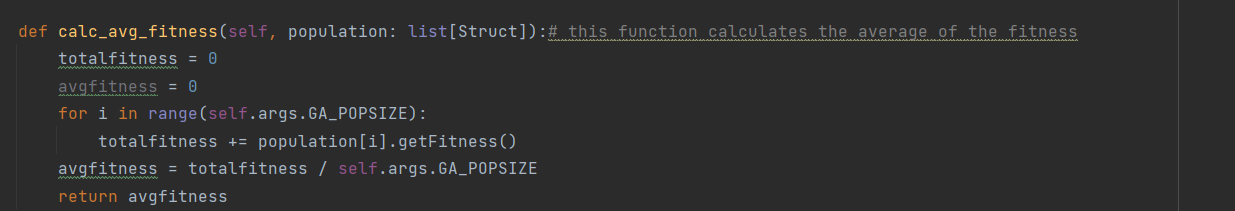
**AI – Lab 1**

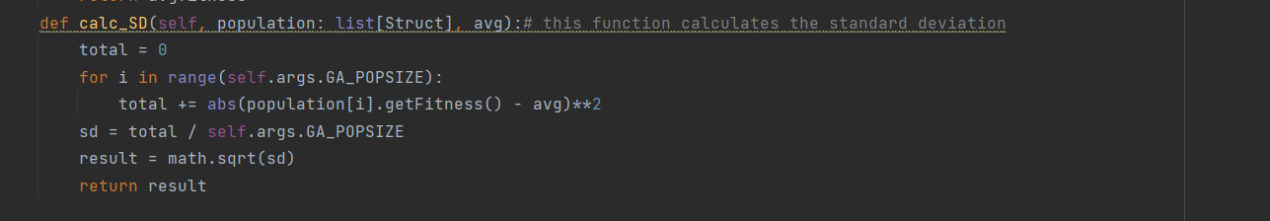
**Part 1**

**Zinat abo hgool 206721714**

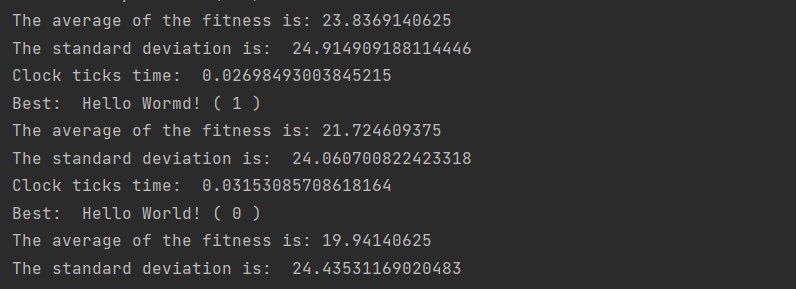
**Elie hadad 207931536**

**1)the average of the fitness**

**The Standard Deviation**

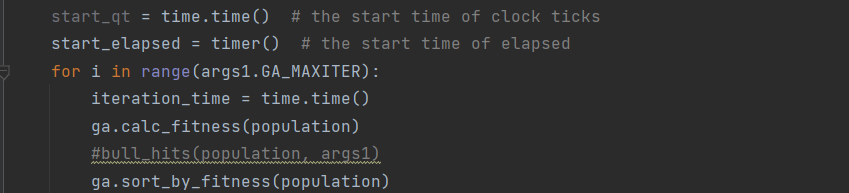
****

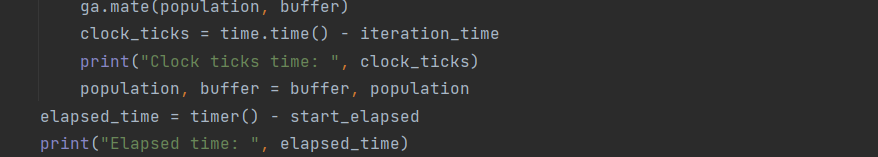
You can see the standard deviation and the average fitness for every generation:

****

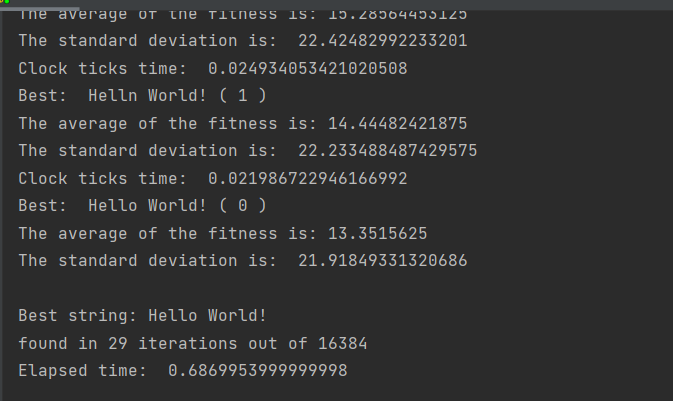
**2) clock ticks and elapsed:**

To show the time of clock ticks and elapsed we added the following lines to the run function in the main file



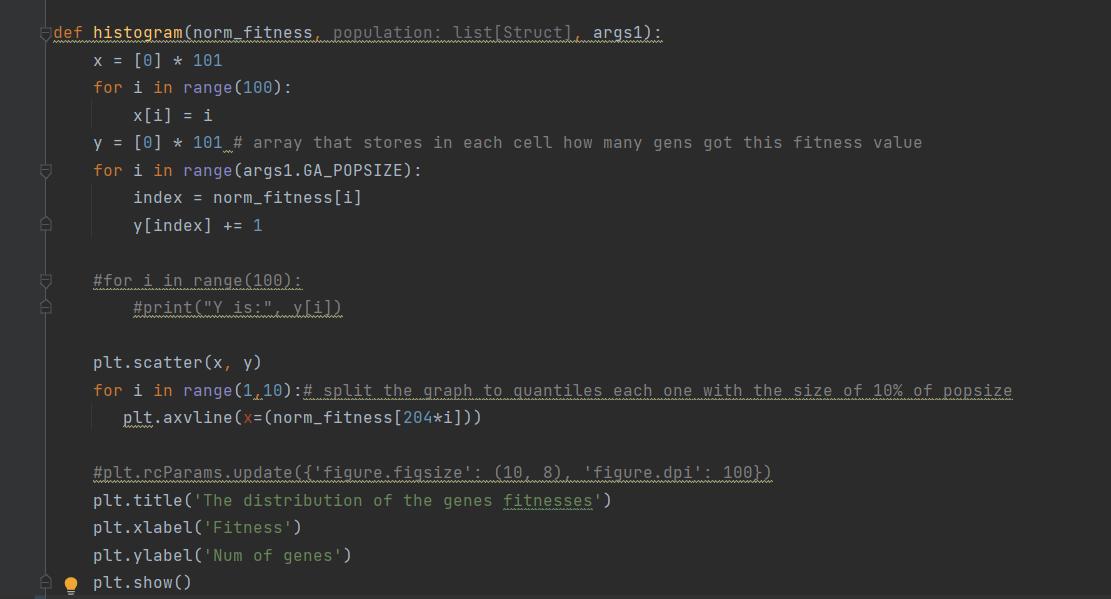


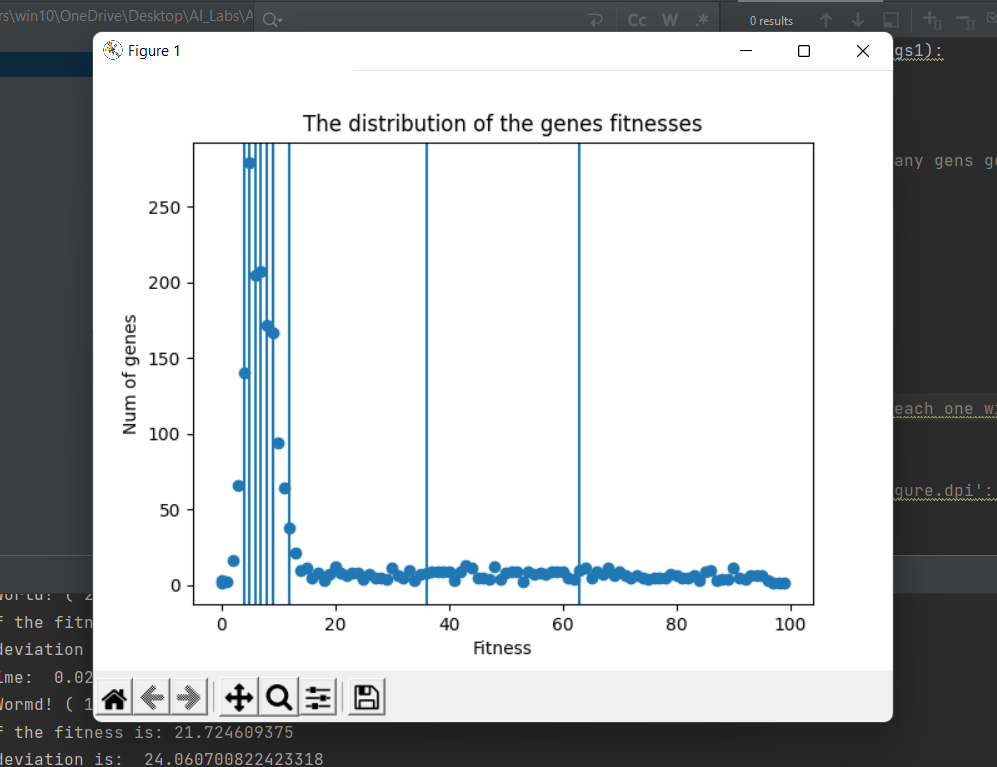
This is the output you can show the values of clock ticks and elapsed time:



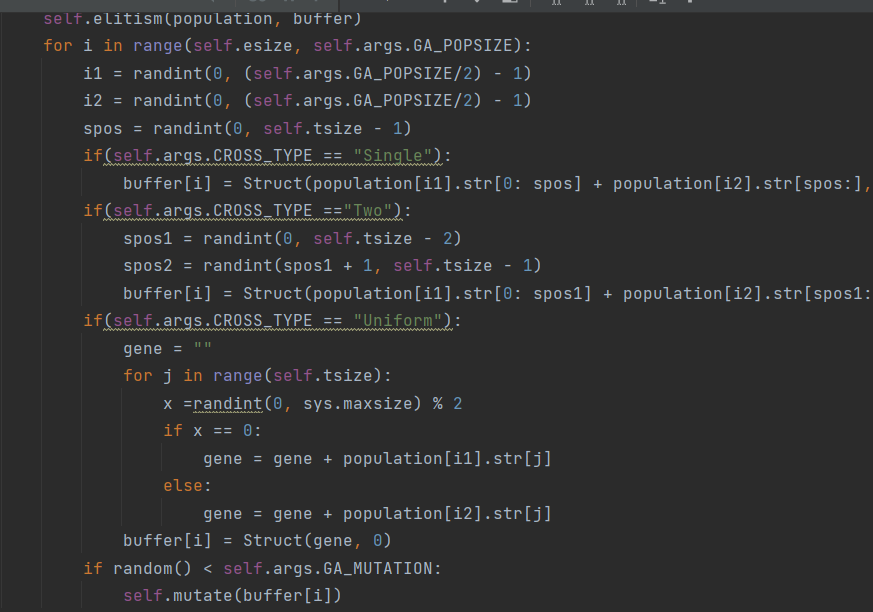
**3)fitness histogram:**

To implement the fitness histogram, we first normalized the fitness values of the population to be between the range (0,100).

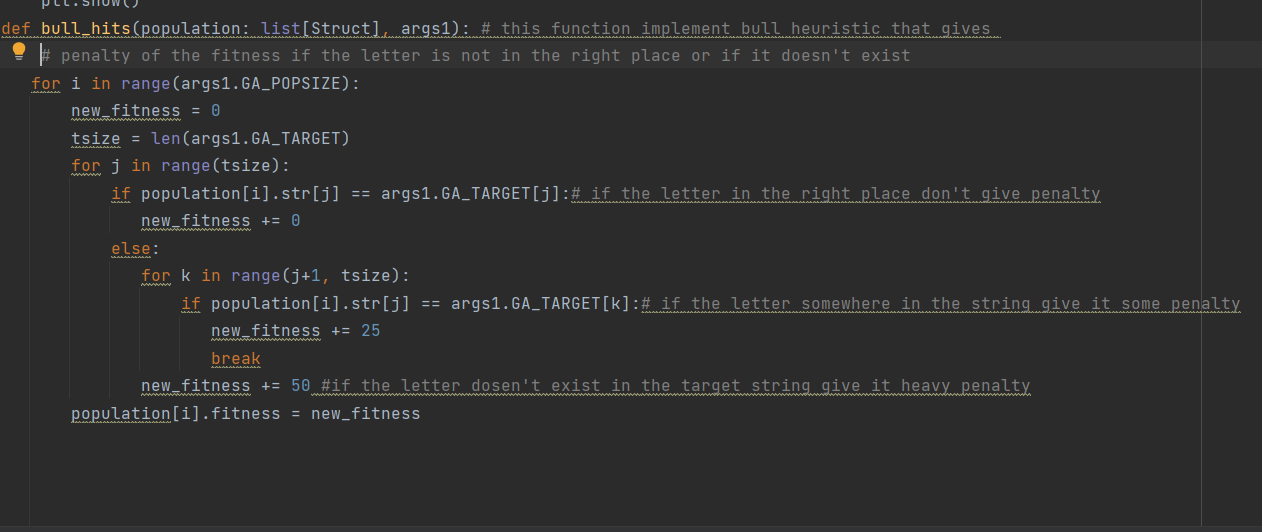
Then we stored how many genes had the same value of fitness, and in the final histogram we divided the graph by vertical lines each one of them is a quantile that represents 10% of the population size.

****

**4)single, two, uniform crossover:**

****According to what we learned in the lecture we implemented the different types of crossover as the following:

**5)Bull's eye:**

****We implemented heuristic function called bull\_hits that based on the idea of bull's eye, it gives penalty on the fitness of the gene if the letter was not on the right place or was missing:

**6)Bull's eye VS. distance of letters heuristic**

**We concluded that bull's eye heuristic performance is preferable over the performance of the original heuristic(distance of the letters). If we imply the different crossover type in both heuristic, we can see clearly that bull's eye brings less generation number, with less numbers of iterations to find the solution compared to original heuristic.**

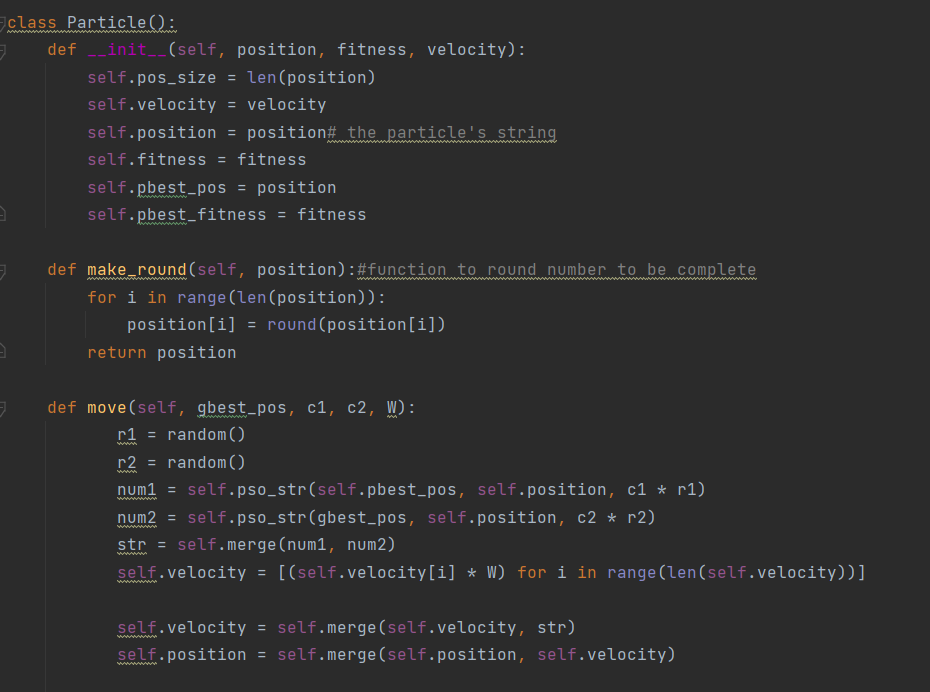
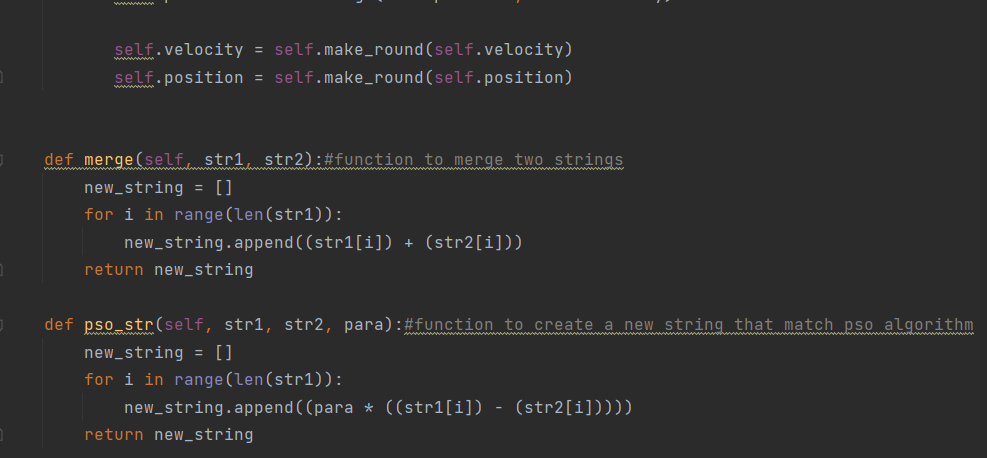
**7)exploration VS. exploitation**

**Exploration: our code implemented exploration in the section of crossover, because it brings new child to the population, and by this operation the code explore new string for the new child, also we consider the part of mutate function as exploration, because in this function we change the string.**

**Exploitation: in the elitism function we have exploitation because in this function we concentrate on specific percent of the best genes in the population, this means that we want to stay with this specific percent and not to explore for others.**

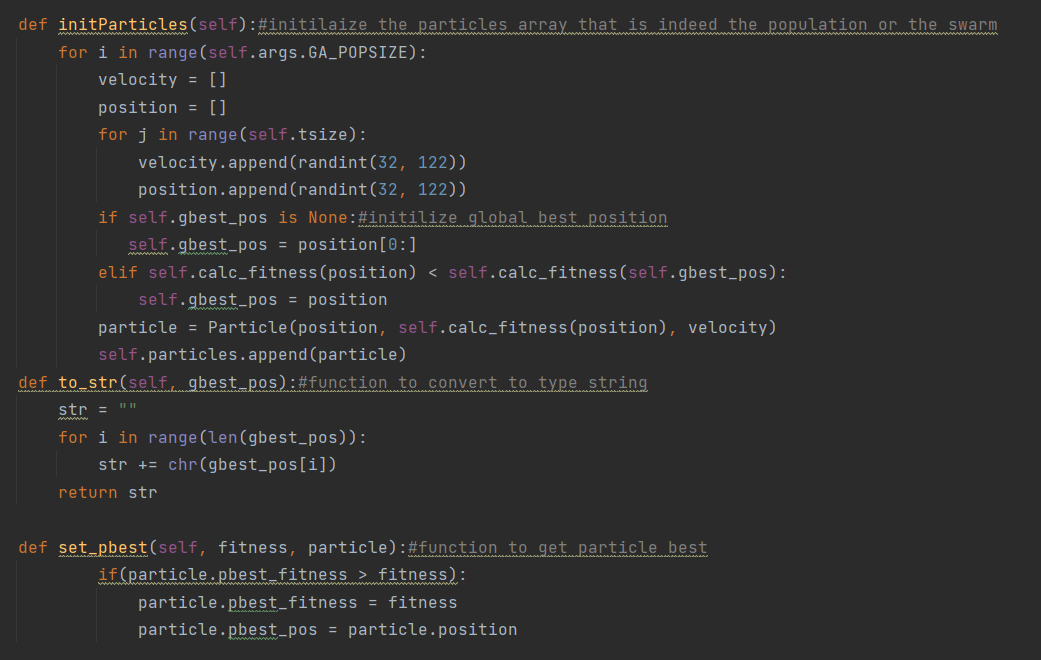
**8)PSO**

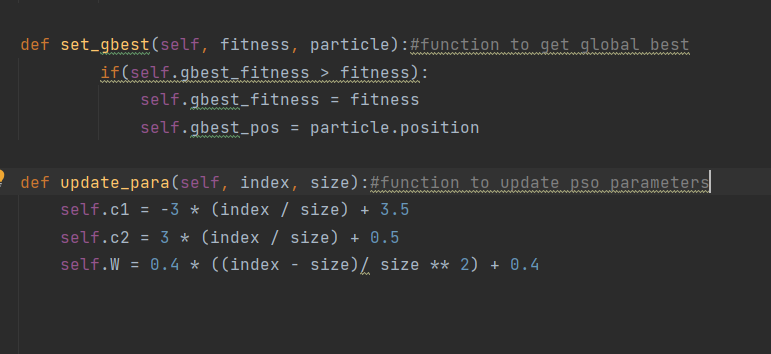
**This is our implementation of PSO algorithm:**

**This first class for the particle and its attributes and functions**

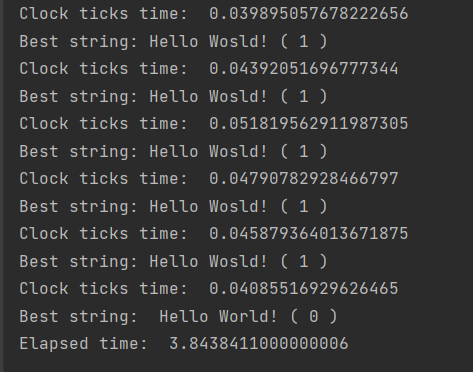
**The second class for the whole swarm that consist of the particles and other different attributes and functions**

****

****

****

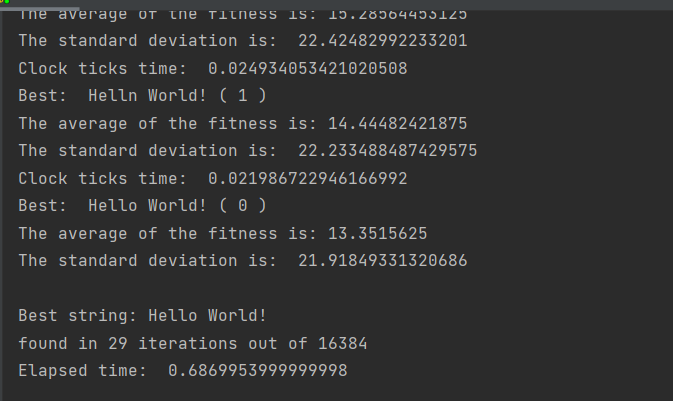
**When we run the PSO algorithm we get the following output:**

****

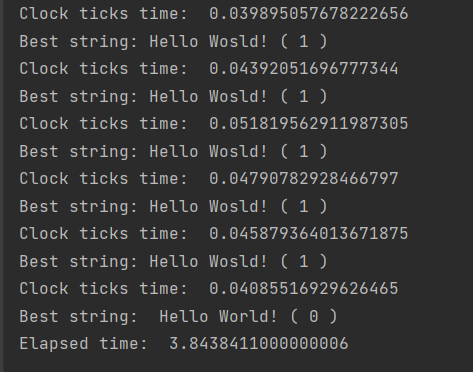
**9)PSO VS. GA:**

**We can see clearly when we run both algorithms the difference between elapsed time, so the performance of GA is preferable over PSO**

**GA time:**



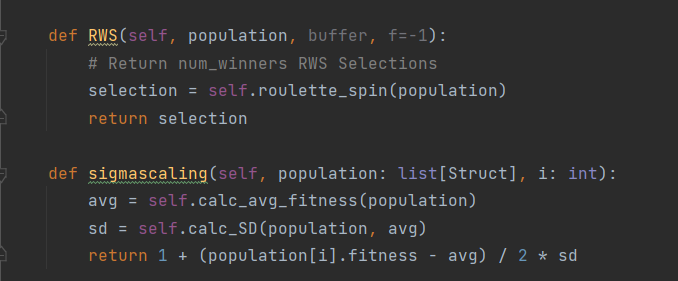
**PSO time:**

****

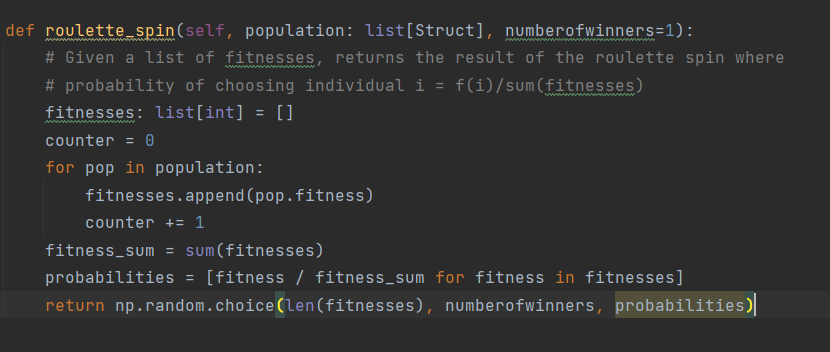
**Lab 1 – part 2:**

1)

**RWS + sigma scaling:**

****

**Ruolette spin:**

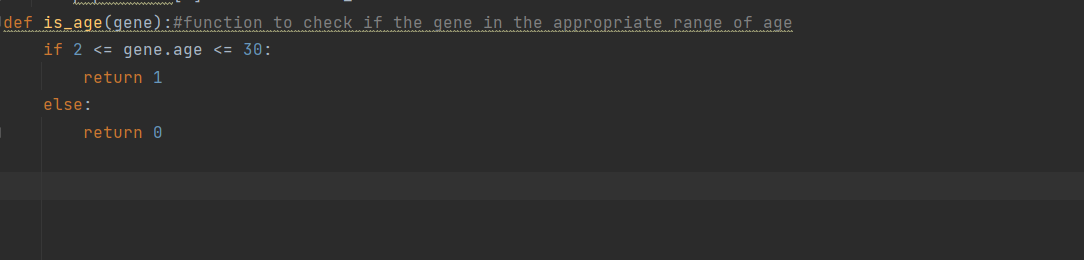
****

**Tournament selection:**

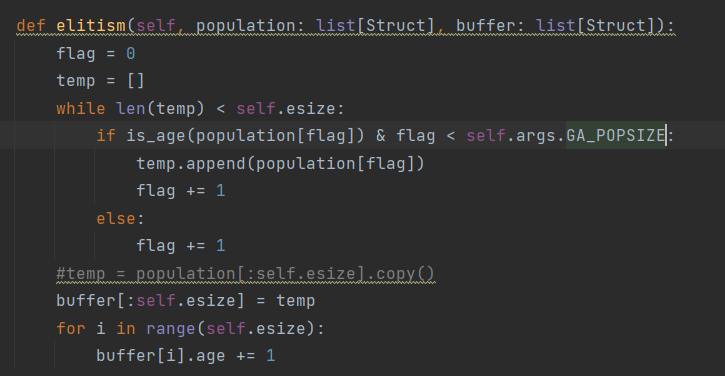
****

**2)Aging:**

**To support the aging feature in the genetic algorithm we implemented the following:**

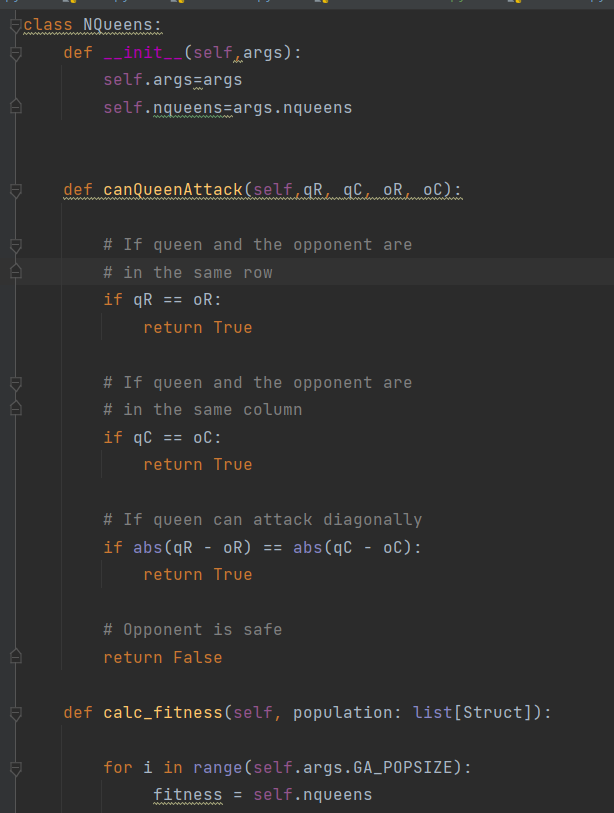
**\*we selected the ideal aging range to be between 2 and 30**

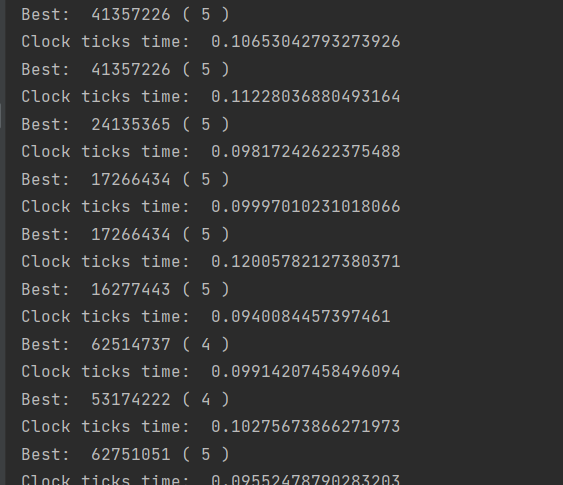
**And we changed the elitism function in order to support aging:**

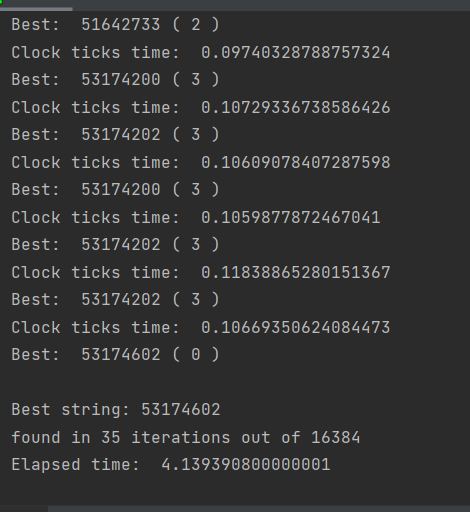
****

**3) N-queens:**

**We added a file to implement N-queens problem you can check the full code to see it, here is a snip:**

****

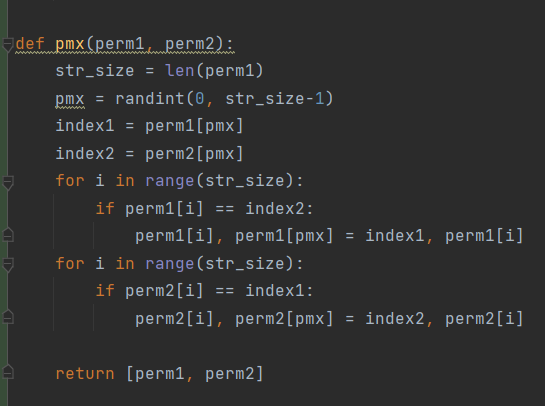
**This is the following output when we run N-queens:**

****

**4) applying PMX & XC & simple inverse mutate & swap mutate**

**PMX:**

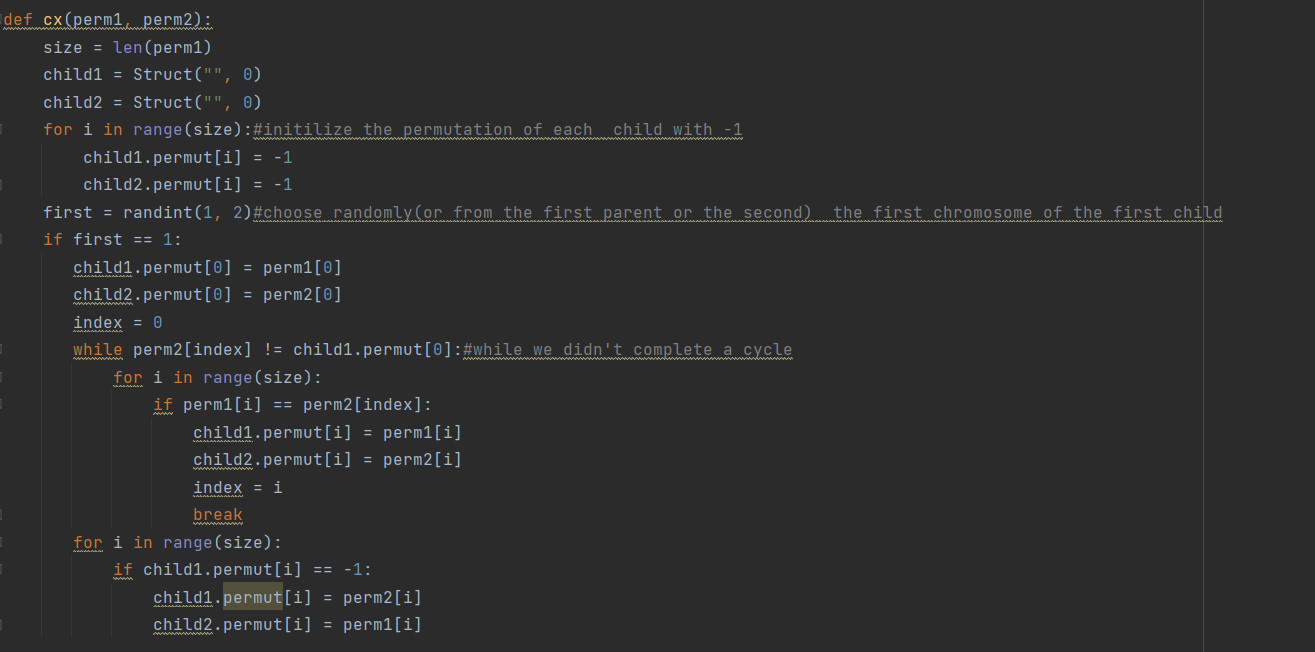
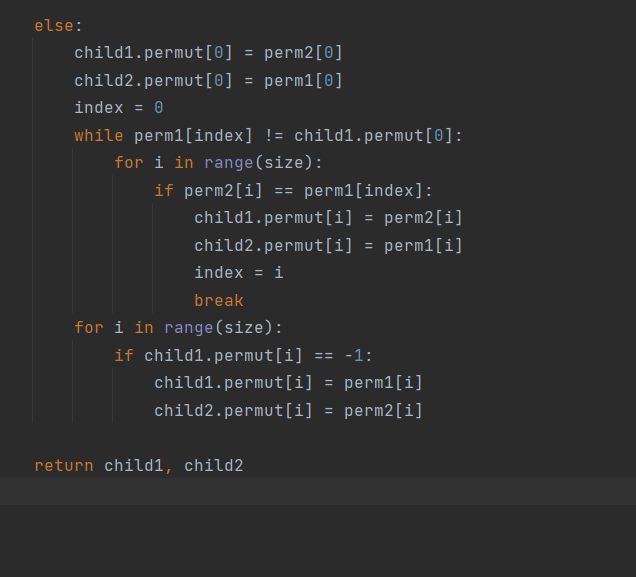
**This function takes the permutations of two genes and implement the following:**

****

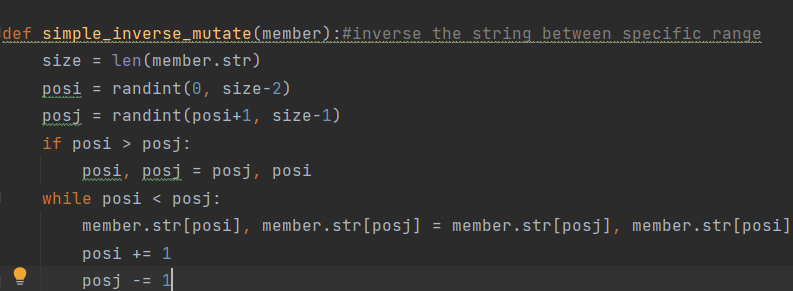
**CX:**

**we based our algorithm for implementing the cycle crossover on the following article:**

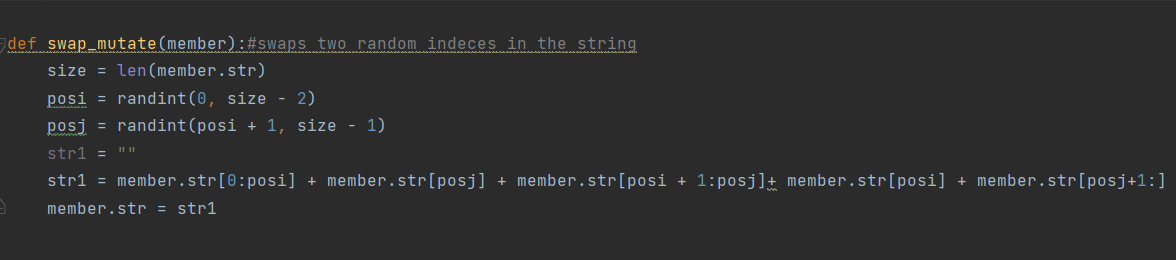
[**https://www.hindawi.com/journals/cin/2017/7430125/**](https://www.hindawi.com/journals/cin/2017/7430125/)

**also here the function take two permutations of the parents and returns the possible children as the following:**

**simple inverse mutate:**

****

**Swap mutation:**

****

**5) N-queens VS. Bull's eye:**

|  |  |  |
| --- | --- | --- |
| Bull's eye | N-queens |  |
| The more the population size is bigger we get more run time and less generated genes, and vice versa | The more the population size is bigger we get more run time and less generated genes, and vice versa | **Population size** |
| When we increase the mutation rate, generally we get we get less generated genes and less run time, vice versa( in some cases we get slightly more) | When we increase the mutation probability, we get less generated genes and less run time, vice versa | **Mutate probability** |
| The fastest strategy was tournament, comparing to sus and rws | The fastest strategy was tournament, comparing to sus and rws | **Selection strategy** |
| We choose the perfect age to be between 2 to 30, although survival strategy based on fitness was preferable | We choose the perfect age to be between 2 to 30, although survival strategy based on fitness was preferable | **Survival strategy** |
| We get better performance when we applied uniform corssover | We get better performance when we applied CX rather than PMX | **Crossover type** |

**\*the optimal collection of parameters that we want to use in applying the next sections is:**

**POP\_SIZE = 1800**

**MUTATE\_RATE = 0.5**

**ELITRATE = 0.1**

**CROSSOVER\_TYPE = uniform(for bull's eye)/ CX (for N-queens)**

**SELECTION\_TYPE = tournament**

**NUM\_QUEENS = 8**

**6)minimal conflicts:**

**We added new file that support the implementation of minimal conflicts, here a snip of it:**

****

**7)**