**TECNOLÓGICO NACIONAL DE MÉXICO**

*INSTITUTO TECNOLÓGICO DE CULIACÁN*

**MATERIA:**

*INTELIGENCIA ARTIFICIAL.*

**TEMA:**

PROYECTO FINAL ALGORITMOS GENÉTICOS ENE-JUL 2020.

**GRUPO:**

EB02.

**HORA:**

09:00-10:00.

**PROFESORA:**

ING. NORA ESMERALDA CANCELA GARCÍA.

**INTEGRANTES:**

AGUIRRE CASTRO ANDRÉS

CORTEZ ESPARZA EDUARDO ALEJANDRO

DIAZ VILLEGAS FRACISCO JAVIER

REYES PEÑA PEDRO CESAR

**CARRERA:**

INGENIERIA EN SISTEMAS COMPUTACIONALES.

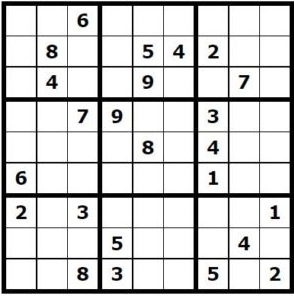
**DESCRIPCIÓN DEL PROBLEMA**

El Sudoku puede considerarse un problema de optimización con restricciones, a nivel de fila, columna y subcuadrícula.

**Objetivo:** Utilizar un algoritmo genético para resolver el sudoku.

Considere las siguientes recomendaciones:

**Representación del individuo**: El Sudoku es un claro ejemplo de representación basado en otros problemas de optimización combinatoria. Cada individuo (tablero) se codifica en un array de 81 posiciones; este array representa el cromosoma que está formado por 9 genes (filas). De esta forma, supongamos que tenemos el siguiente tablero.



Una representación recomendada es que cada individuo o cromosoma se representa por un array compuesto por 9 genes de 9 elementos cada uno. Cada gen se corresponde con una fila del tablero.

Las posiciones fijas del tablero se respetan y las posiciones vacías se representan como 0:



Cada valor del individuo o cromosoma tendrá un valor del 1 al 9. El cromosoma se inicializa con valores aleatorios entre 1 y 9, respetando las posiciones fijas y sin generar elementos repetidos dentro del gen (fila).

**Función de aptitud**: La función de aptitud se basa en que los elementos de cada fila y columna deben ser los valores del conjunto Dígitos = {1,2,3,4,5,6,7,8,9}. La función calcula el número de elementos que faltan en cada submatriz y columna. A nivel de fila no existe penalización pues los elementos son permutaciones.

**Aptitud=** Total de faltas de cada subcuadrícula + total de faltas de las columnas

Se trata de un problema de minimización y el objetivo a conseguir aptitud 0.

**HERRAMIENTAS DE APOYO PARA EL PROYECTO EN EQUIPO**

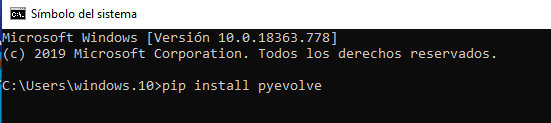
Se utilizaron diferentes herramientas para poder trabajar en equipo y poder llevar de manera exitosa este proyecto con la participación de todos, porque estuvimos en comunicación a distancia, entonces las herramientas usadas fueron las siguientes: un **repositorio de Git** para subir las versiones de nuestro proyecto algoritmo evolutivo que se encuentra en **GitHub**, videoconferencias en la aplicación de **discord** y **zoom** para tratar asuntos relacionados con el proyecto.

**INSTALACIÓN DEL SERVIDOR JUPYTER NOTEBOOK**

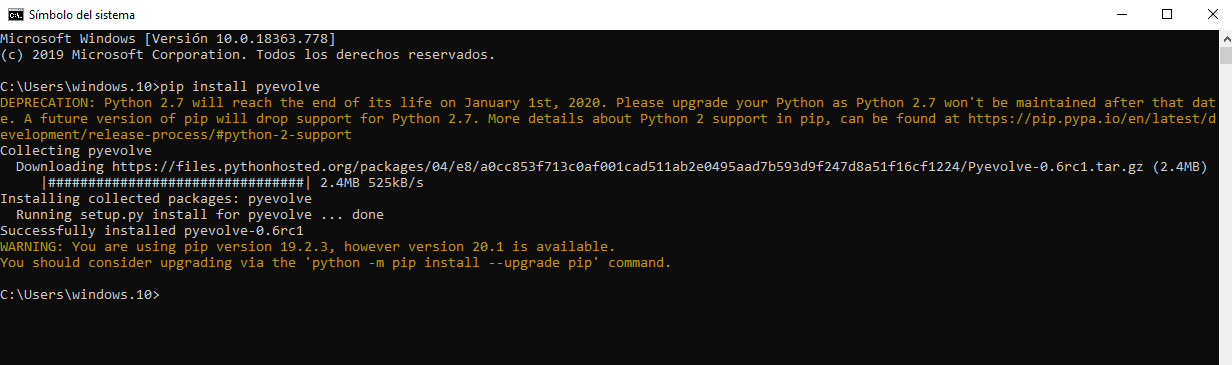
**Jupyter Notebook** es una aplicación web de código abierto que permite crear y compartir documentos que pueden contener código y compilarlo en tiempo teal, además de que puede contener texto, y contenido multimedia que puede interactuar con el programa.

Para ello, se realizaron los siguientes pasos de instalación del Jupyter Notebook:

**Paso 1.-** Entrar al CMD y escribir la siguiente instrucción: “pip install pyevolve” para instalar las dependencias o paquetes de la librería pyevolve.



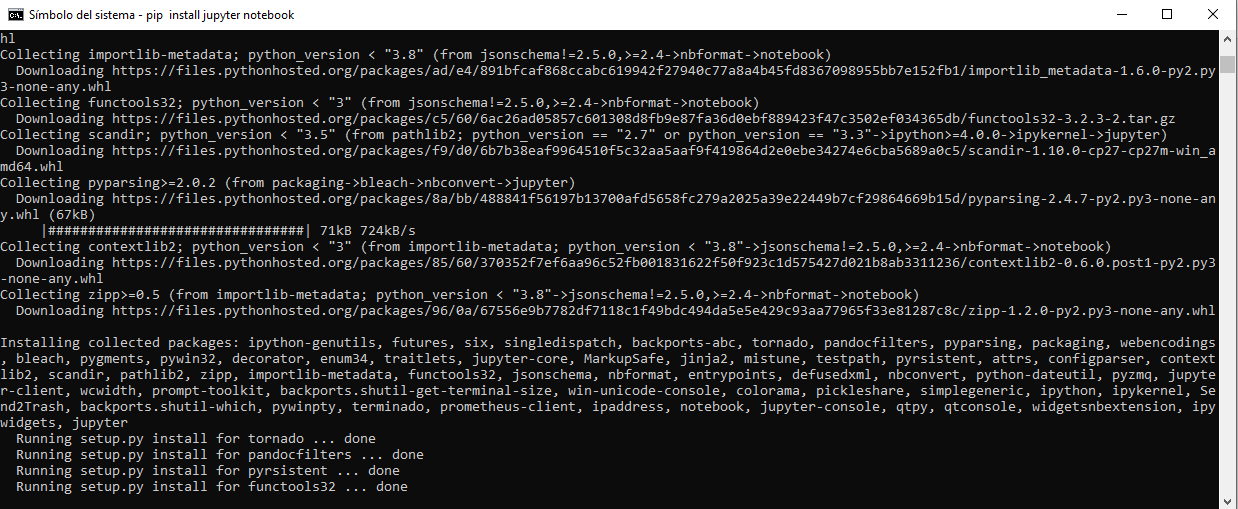
**Paso 2.-** Verificar correctamente la instalación de las dependencias o paquetes de la librería pyevolve.

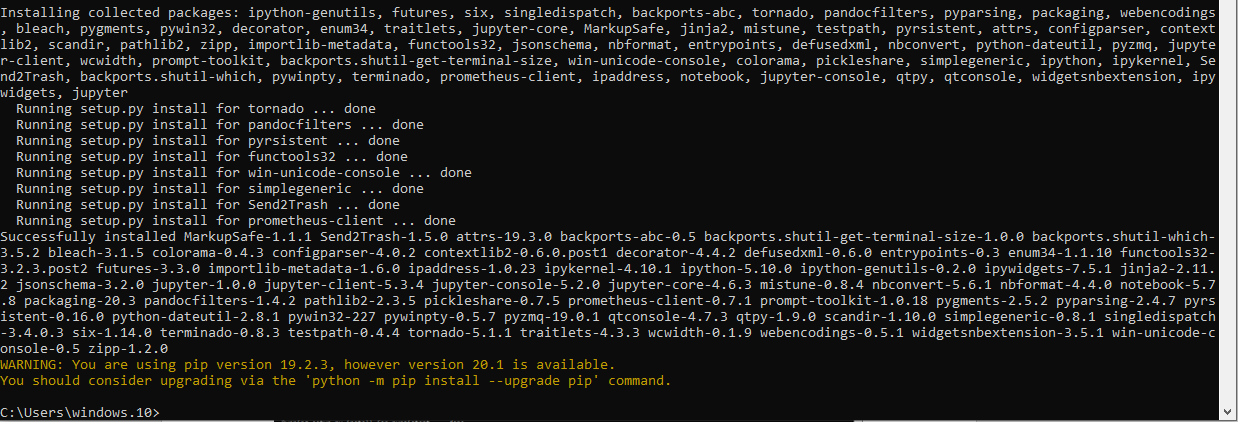


**Paso 3.-** Escribir el siguiente comando: “pip install jupyter notebook” para instalar las dependencias o paquetes del servidor jupyter notebook.

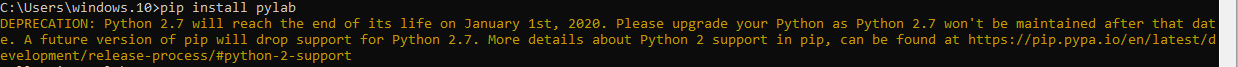


**Paso 4.-** Verificar correctamente la instalación de las dependencias o paquetes del servidor jupyter notebook.

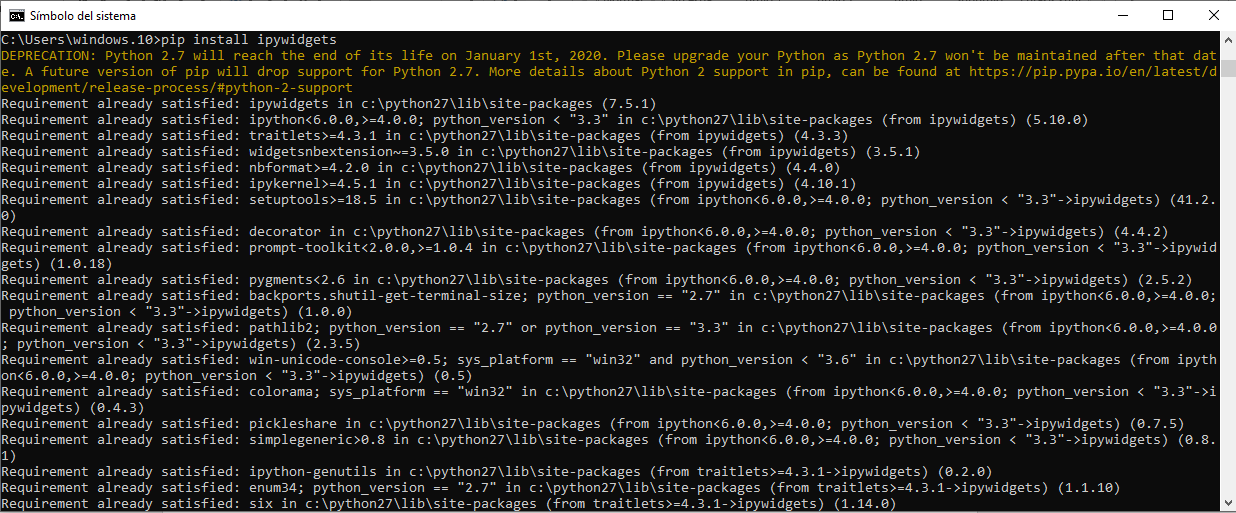




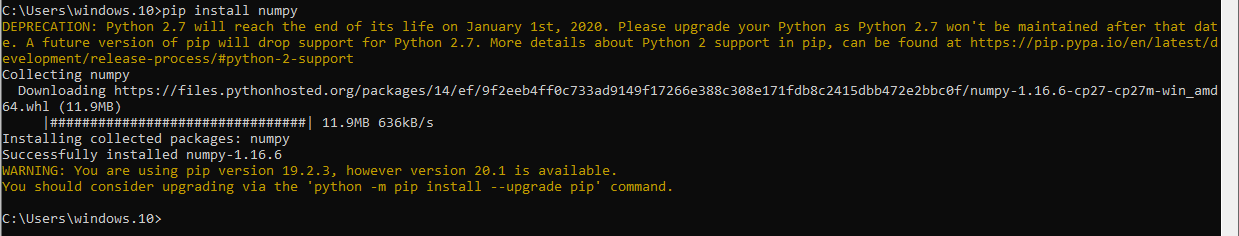
**Paso 5.-** Escribir el siguiente comando: “pip install pylab”. PyLab es un módulo que pertenece a la biblioteca matemática de Python Matplotlib. PyLab combina el módulo numérico numpy con el módulo de trazado gráfico pyplot.



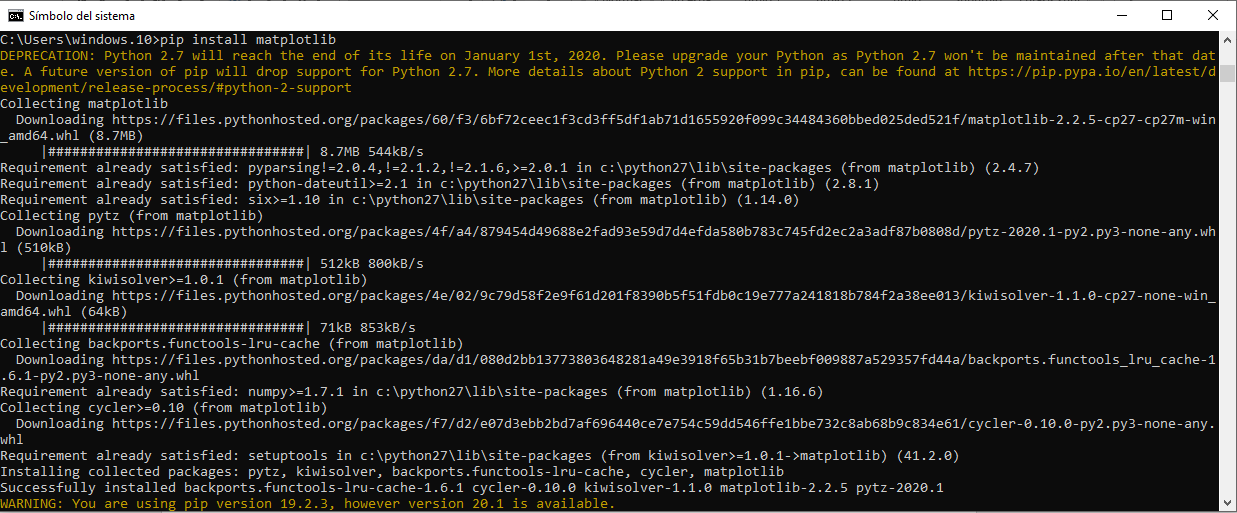
**Paso 6.-** Escribir el siguiente comando: “pip install ipywidgets”. La librería ipywidgets nos permite transformar un notebook de Jupyter en un explorador de datos interactivo.



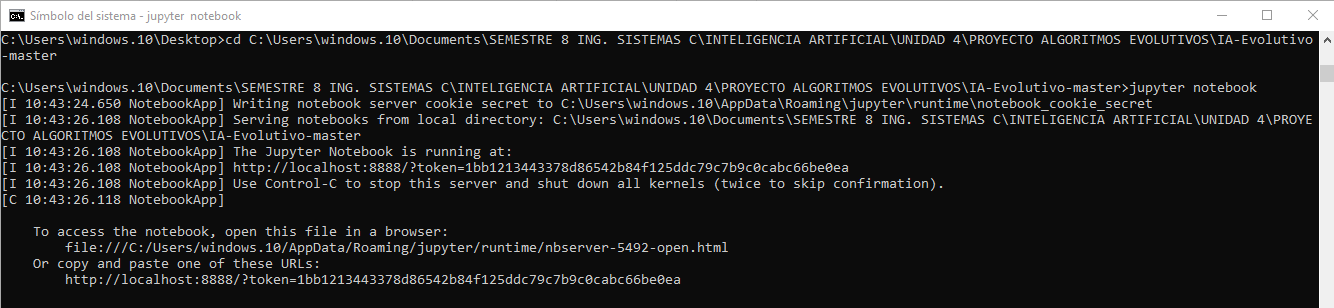
**Paso 7.-** Escribir el siguiente comando: “pip install numpy”. Es necesario esta librería para la gráfica de la “función de aptitud”.



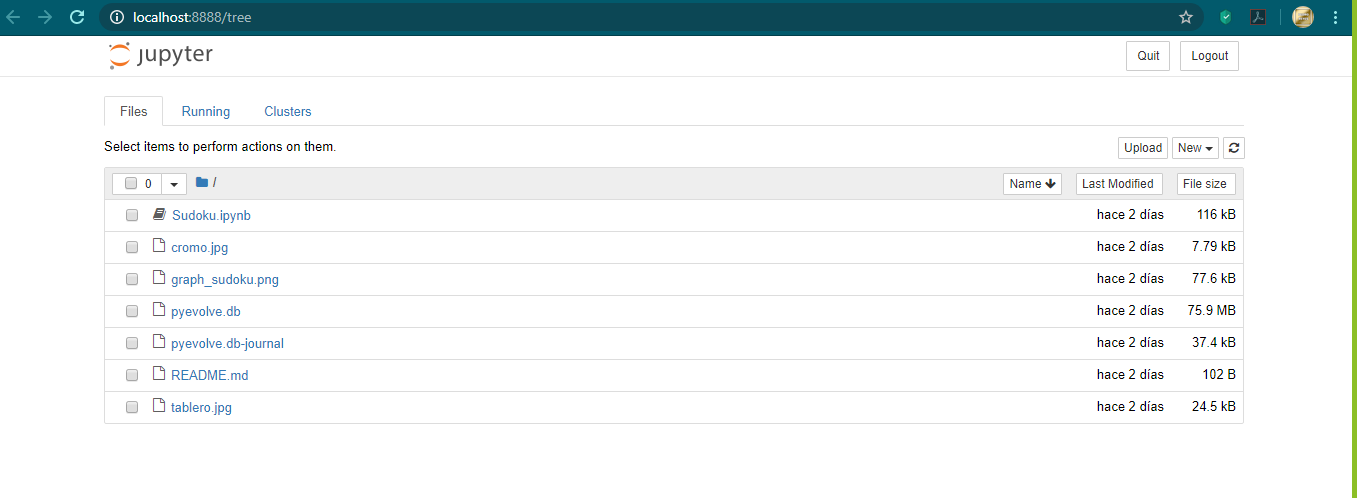
**Paso 8.-** Escribir el siguiente comando: “pip install matplotlib”. Es otro complemento importante que se debe instalar para la gráfica de la “función de aptitud”.

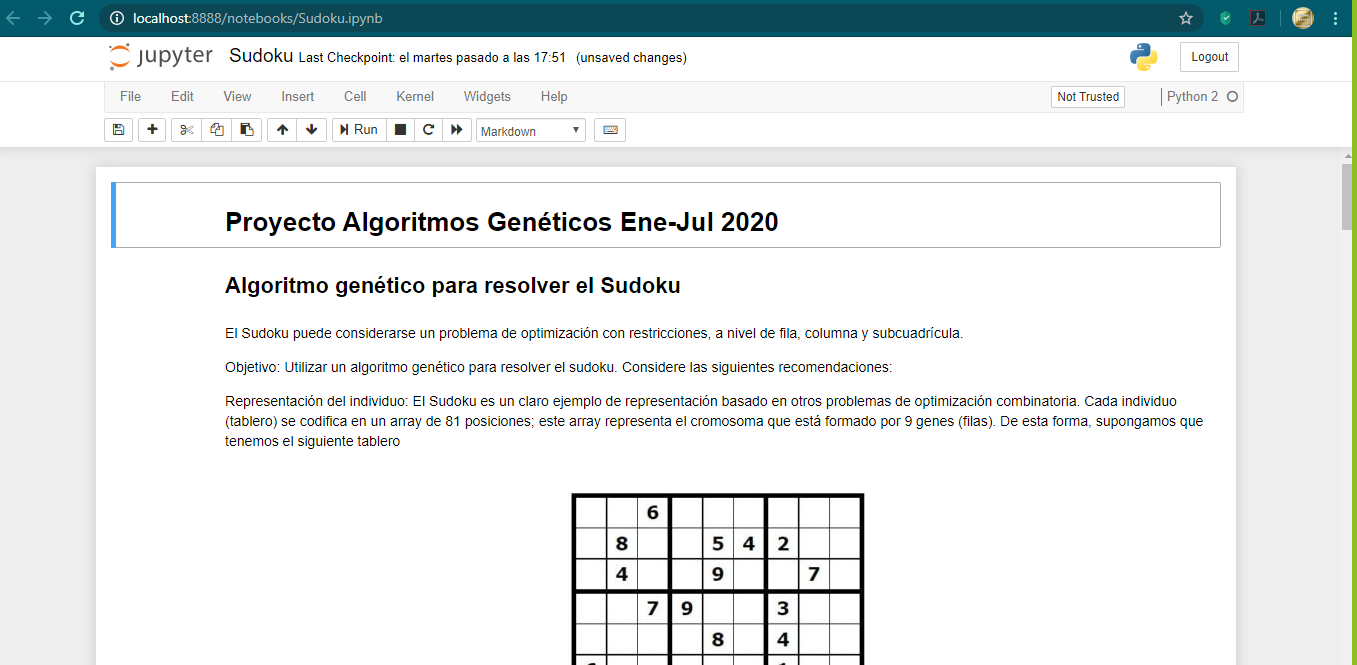


**Paso 9.-** Ahora, accedemos al jupyter mediante nuestro CMD y realizamos lo siguiente:



**Paso 10.-** Finalmente, se puede observar que el servidor está corriendo.





**IMPLEMENTACIÓN UTILIZANDO CUALQUIER LENGUAJE DE PROGRAMACIÓN, BIBLIOTECA, LIBRERÍA.**

Se utilizo el lenguaje de **Python** como lenguaje de programación de nuestro Algoritmo Genético ya que este lenguaje tiene mayor cantidad de bibliotecas, es claro que hay una mayor cantidad de recursos open-source disponibles en Python en comparación a Java para poder trabajar con Inteligencia Artificial, además de que los prototipos se pueden programar más rápido, al ser Python un lenguaje dinámico y con un tipado débil, esto permite desarrollar más rápidamente que con otros lenguajes, donde se debe explicitar mucho más los tipos de datos.

Nuestra implementación se basó en la **representación binaria, selección por ruleta y luego cruza de un punto.**

Utilizamos **librerías** como:

* pyevolve
* jupyter
* pylab
* ipywidgets
* numpy
* matplotlib

**Código Fuente:**

import ipywidgets as widgets

from pyevolve import Consts

from pyevolve import Crossovers

from pyevolve import G1DList

from pyevolve import Selectors

from pyevolve import DBAdapters

from pyevolve import FunctionSlot

from pyevolve import GAllele

from pyevolve import GenomeBase

from pyevolve import GPopulation

from pyevolve import GSimpleGA

from pyevolve import GTree

from pyevolve import Initializators

from pyevolve import Interaction

from pyevolve import Migration

from pyevolve import Mutators

from pyevolve import Scaling

from pyevolve import Statistics

from pyevolve import Util

from IPython.display import Image

sudoku=[

0,0,6,0,0,0,0,0,0,

0,8,0,0,5,4,2,0,0,

0,4,0,0,9,0,0,7,0,

0,0,7,9,0,0,3,0,0,

0,0,0,0,8,0,4,0,0,

6,0,0,0,0,0,1,0,0,

2,0,3,0,6,7,9,8,1,

0,0,0,5,0,0,0,4,0,

4,7,8,3,1,9,5,6,2]

def gen(tablero,ngen):

# esta funcion retorna una lista de gen(fila) del sudoku

gen=[]

for i in range(9):

gen.append(tablero[ngen\*9+i])

return gen

def columna(tablero, ncolumna):

# esta funcion las columnas del sudoku

columna=[]

for i in range(9):

columna.append(tablero[i\*9+ncolumna])

return columna

def subcuadricula(tablero,ncuadricula):

# funcion que retorna el subcuadricula o cuadrante de izquierda a derecha y luego hacia abajo

subcuadricula=[]

cuadriculainicio=int(ncuadricula/3)\*9\*3+(ncuadricula-int(ncuadricula/3)\*3)\*3

for indiceX in range(3):

for indiceY in range(3):

subcuadricula.append(tablero[cuadriculainicio+indiceX\*9+indiceY])

return subcuadrícula

def aptitud(cromosoma):

#esta es la funcion de aptitud, sin errores, significa que se encontro la solucion

error=0

for index in range (9):

for elemento in [gen(cromosoma,index),columna(cromosoma,index),subcuadricula(cromosoma,index)]:

for getal in [1,2,3,4,5,6,7,8,9]:

# verifica los números repetidos del individuo

if (elemento.count(getal))>1:

error+=elemento.count(getal)-1

for i,getal in enumerate(cromosoma):

# verifica que los números que no coinciden con los números iniciales

if getal!=sudoku[i] and sudoku[i]!=0:

error+=2

return error

def data(ngeneraciones,cruza,mutacion):

sqlite\_adapter = DBAdapters.DBSQLite(identify="sudoku", resetDB=True) #Para generar la grafica de aptitud

genoma = G1DList.G1DList(81)

genoma.setParams(rangemin=1,rangemax=9)

genoma.evaluator.set(aptitud)

genoma.setParams(bestrawscore=0.00, rounddecimal=2) # redondea las aptitudes

genoma.crossover.set(Crossovers.G1DListCrossoverSinglePoint) # cambia la cruza a de un solo punto

ga = GSimpleGA.GSimpleGA(genoma)

ga.selector.set(Selectors.GRouletteWheel) # cambia la seleccion por ruleta

ga.setMinimax(Consts.minimaxType["minimize"]) # se minimiza para reducir la cantidad de ceros del tablero

ga.setMutationRate(mutacion) #asignamos el % de mutacion

ga.setGenerations(ngeneraciones) #asignamos la cantidad de genereaciones a crear

ga.setPopulationSize(81) #defiine el tamaño de la poblacion de nuestro tablero

ga.setCrossoverRate(cruza) # definimos el rango de cruza

ga.setElitismReplacement(1) # para eligir un solo individuo elite

ga.terminationCriteria.set(GSimpleGA.RawScoreCriteria)

ga.setDBAdapter(sqlite\_adapter) #definimos donde se guardaran los datos

ga.evolve(freq\_stats=100) #definimos la cantidad de frequencia a la que evolucionara

best = ga.bestIndividual() # generamos el individuo mas apto

print best

**Se permite que el usuario modifique los porcentajes de cruza, mutación y número máximo de generaciones.**

#Grafico interactivo para modificar aspectos del algoritmo

mutacion =widgets.FloatSlider(value =0.04,step = 0.01,min =0, max=1, description='% Mutacion:')

ngeneraciones = widgets.BoundedIntText(value=1000,min=0,max=500000,step=100,description='Generaciones:',disabled=False)

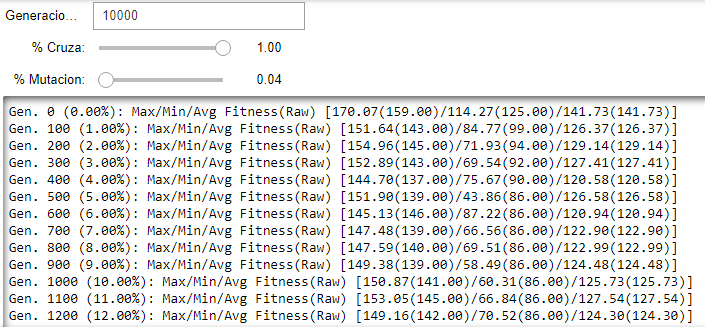
cruza = widgets.FloatSlider(value=1,min=0.1,max=1,step=0.05,description='% Cruza:')

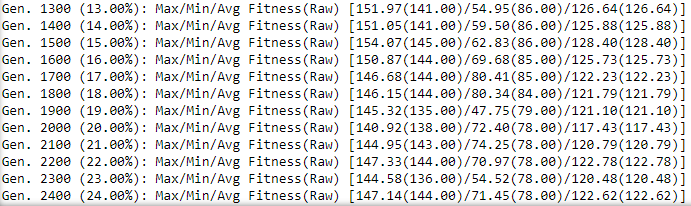
selecion = widgets.VBox([cruza,mutacion,ngeneraciones])

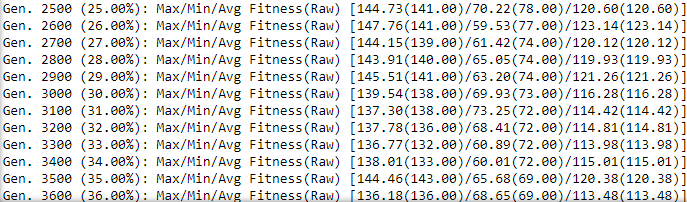
widgets.interact(data,ngeneraciones=ngeneraciones,cruza=cruza,mutacion=mutacion)

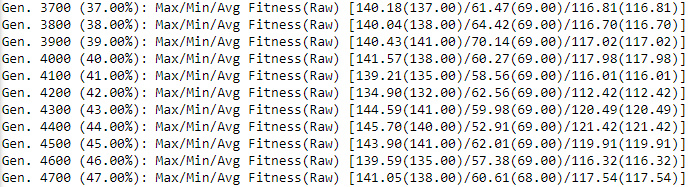
**Debes ejecutar al menos 10 veces el programa con los mismos parámetros (los antes mencionados, almacenaras la mejor solución de cada ejecución y la graficarás. esto te permitirá realizar un análisis a posteriori del algoritmo. sí en la mayoría de la ejecución obtenemos casi el mismo resultado está funcionando correctamente.**

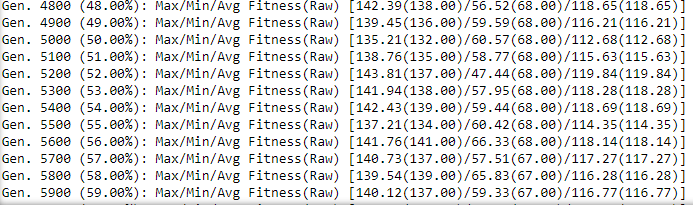
**Ejecución #1**

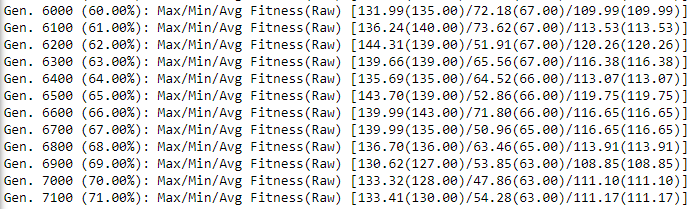


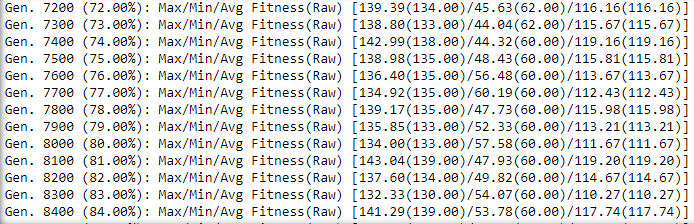


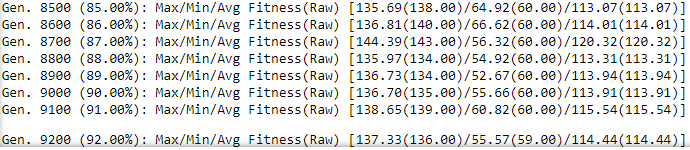


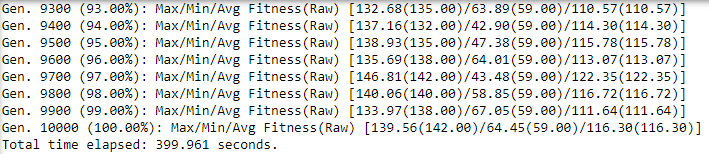


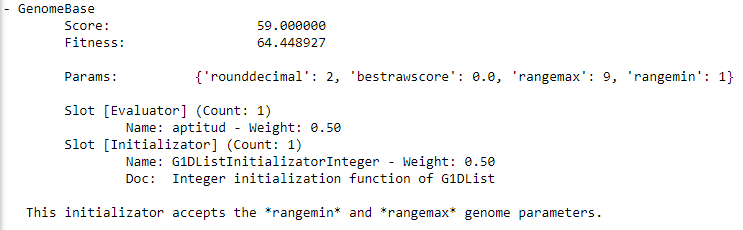


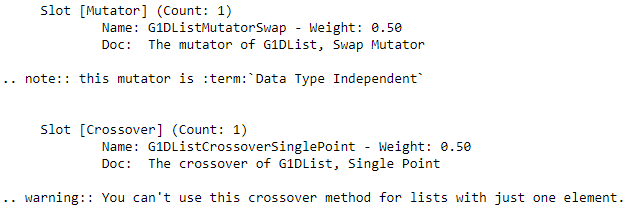


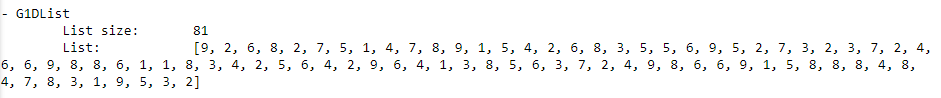


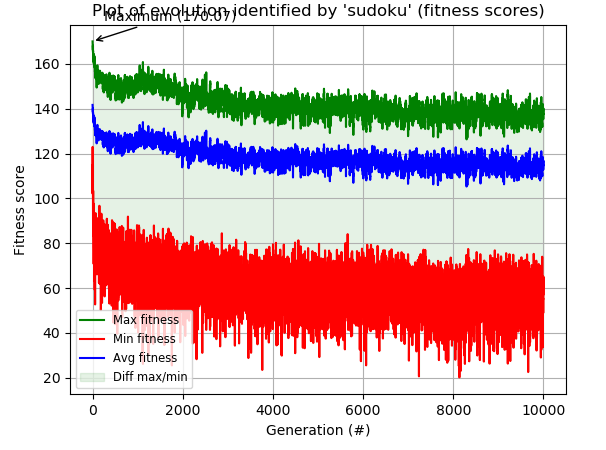




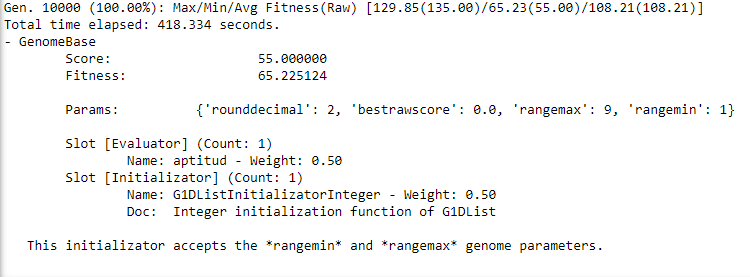


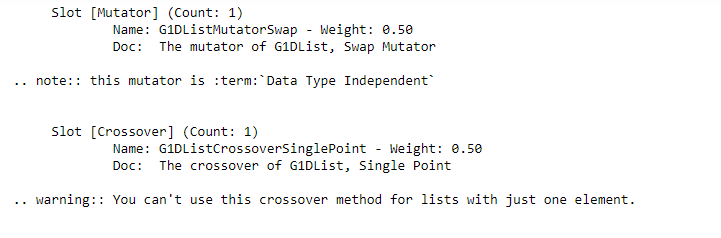
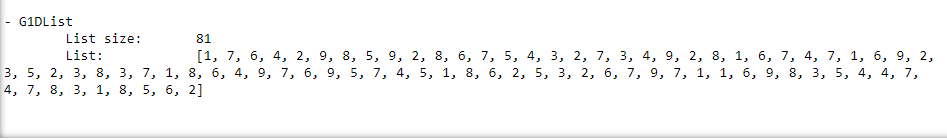


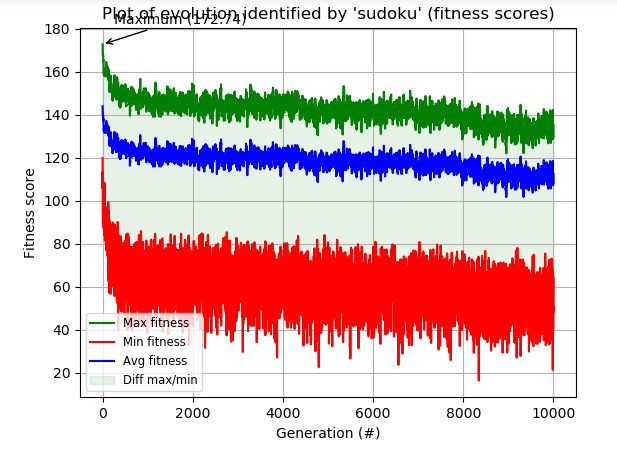




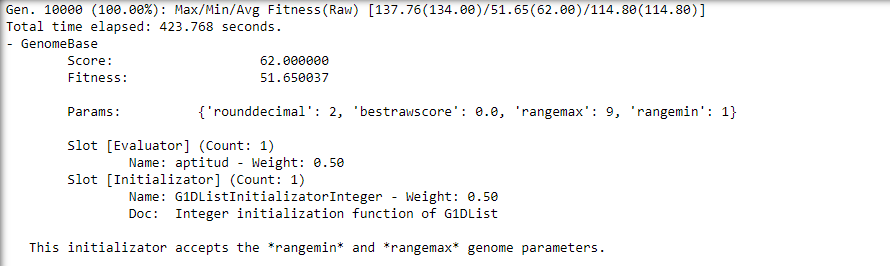
**Ejecución #2**

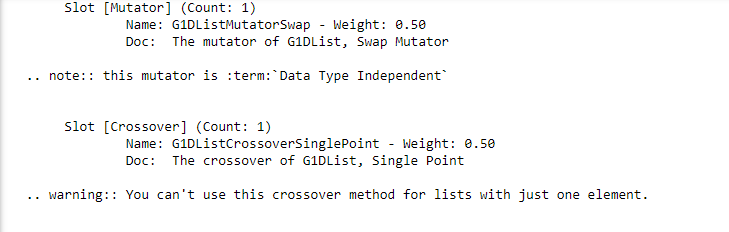


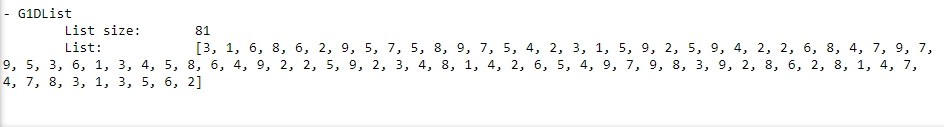
 

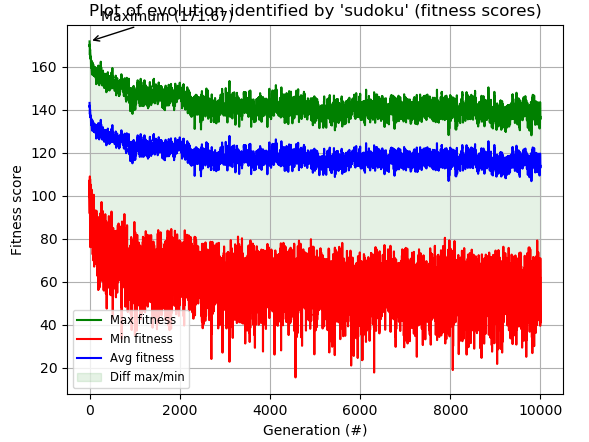


**Ejecución #3**

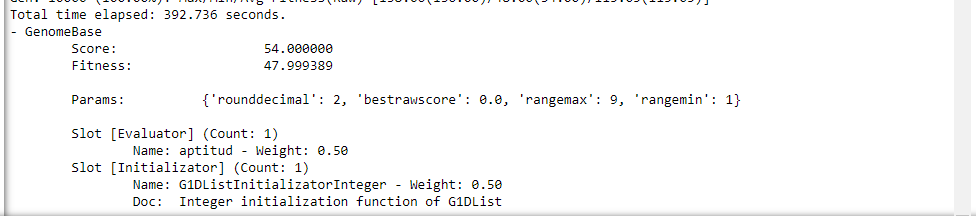


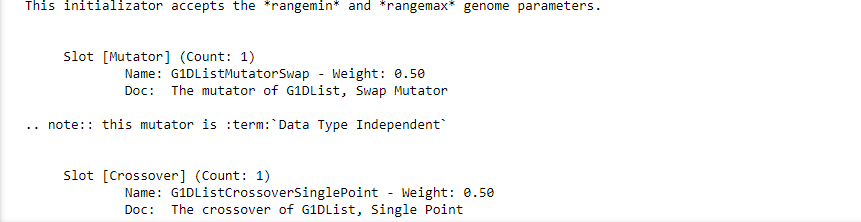


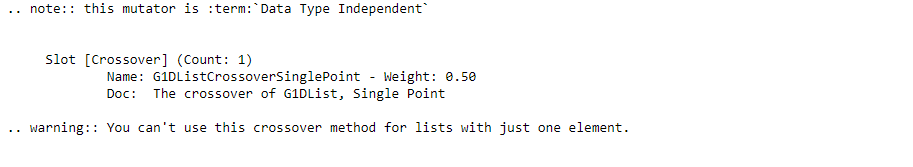


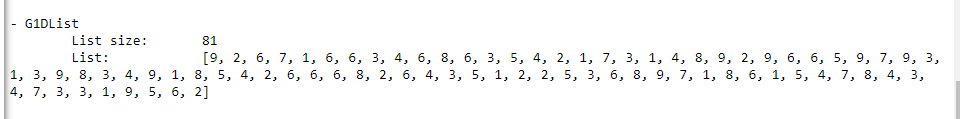


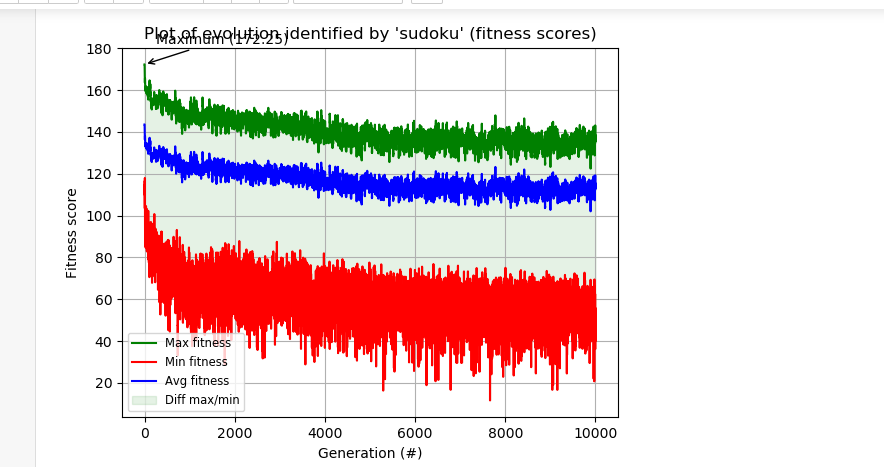
**Ejecución #4**



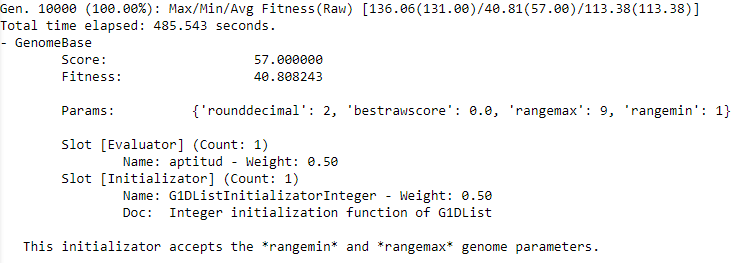


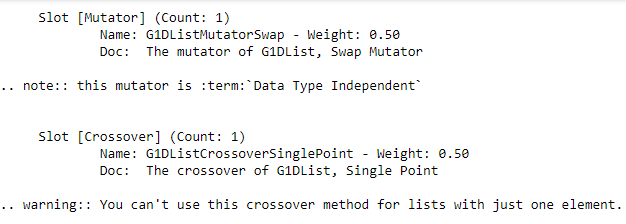


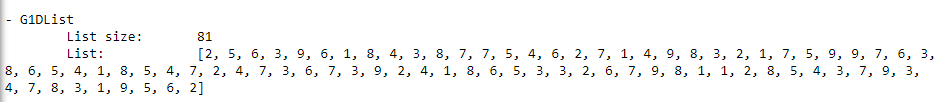


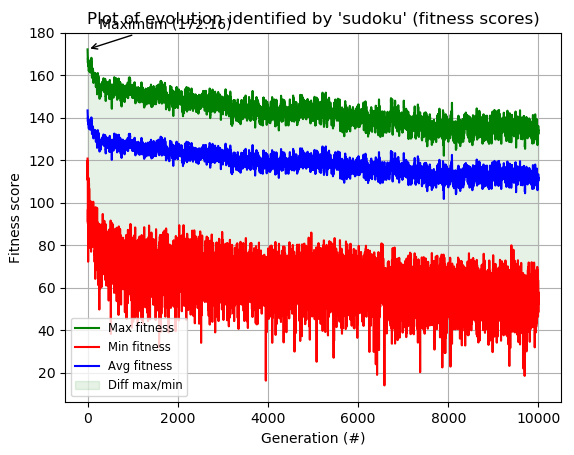


**Ejecución #5**

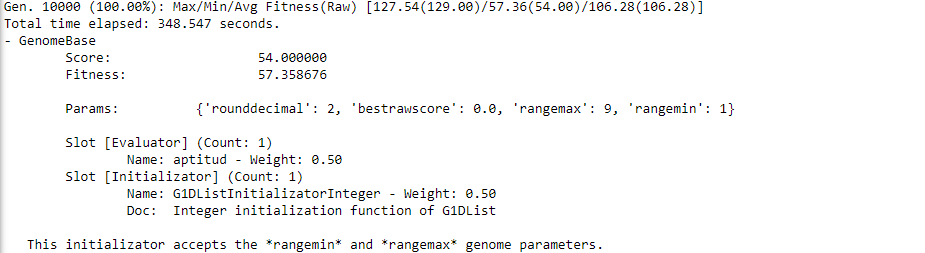


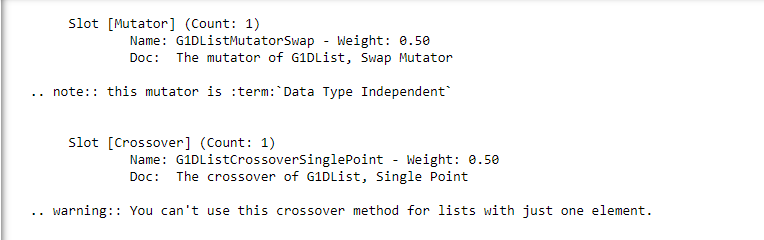


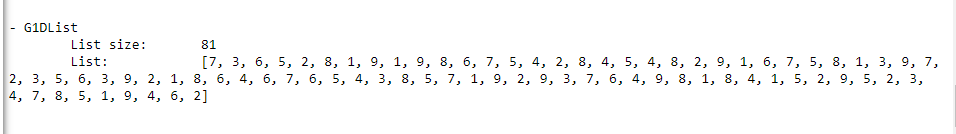


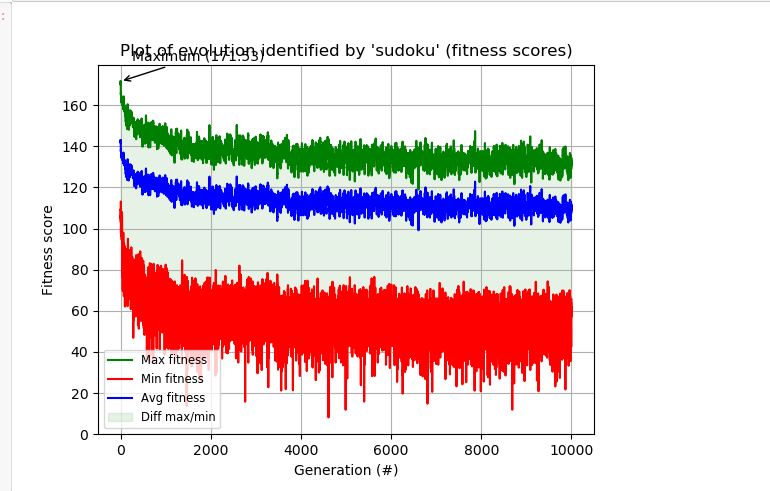


**Ejecución #6**

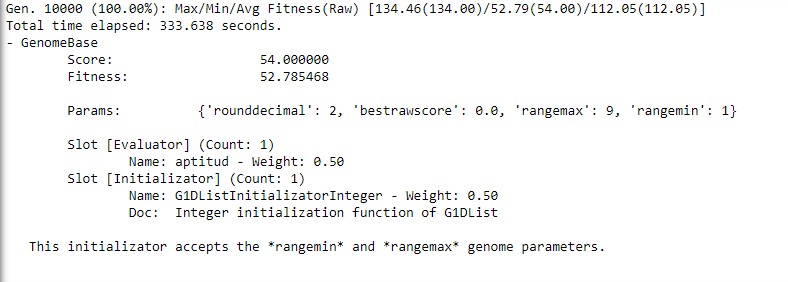


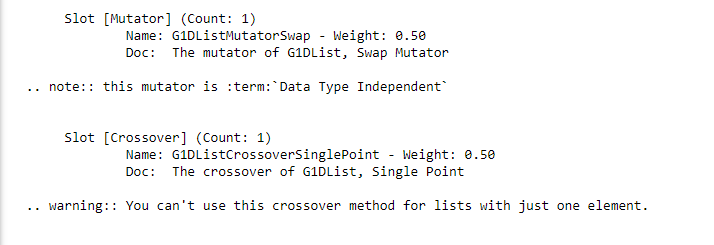


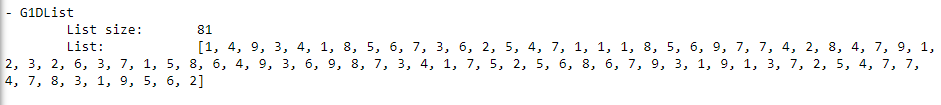


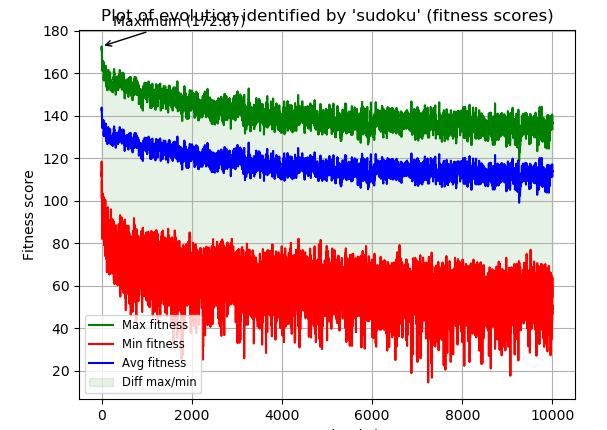


**Ejecución #7**

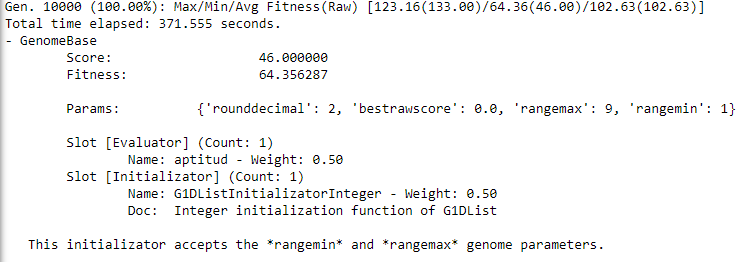


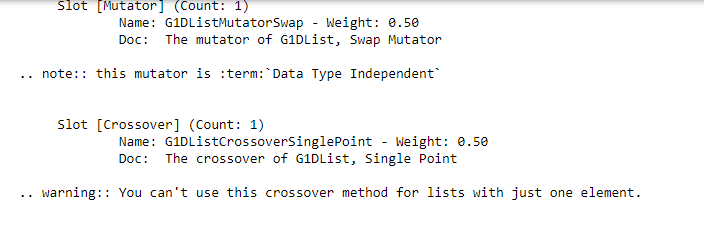


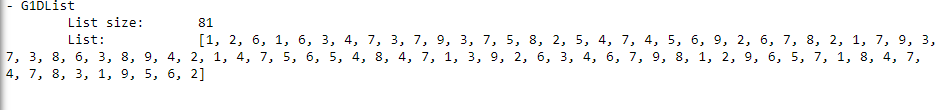


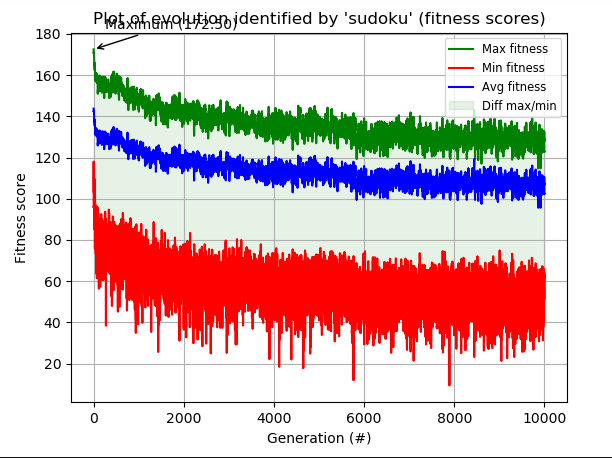


**Ejecución #8**

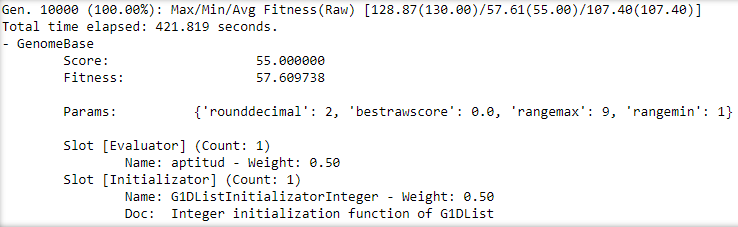


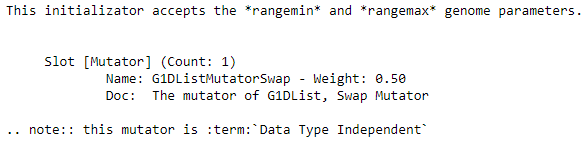


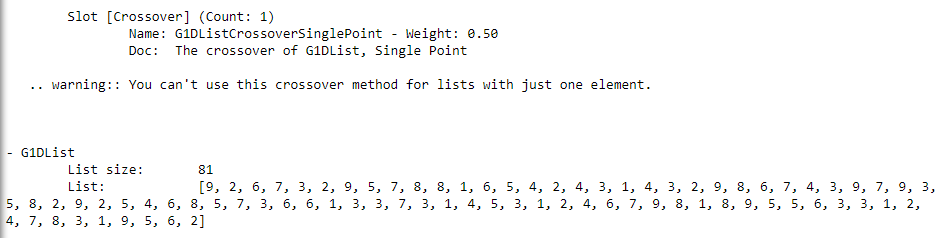


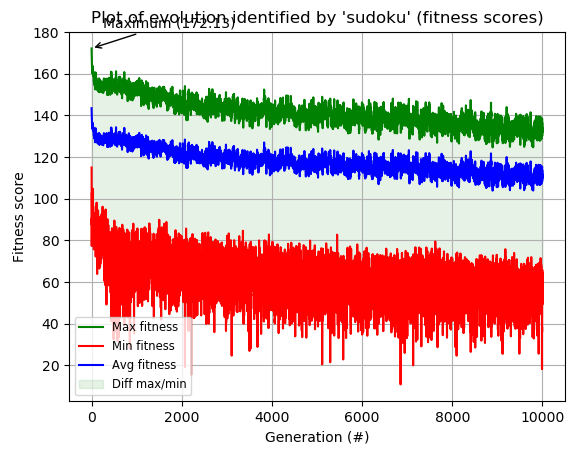


**Ejecución #9**

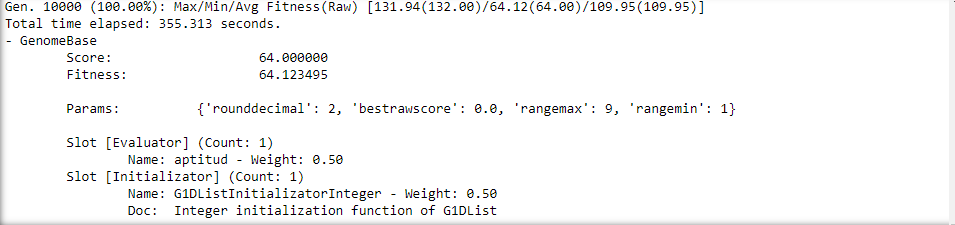


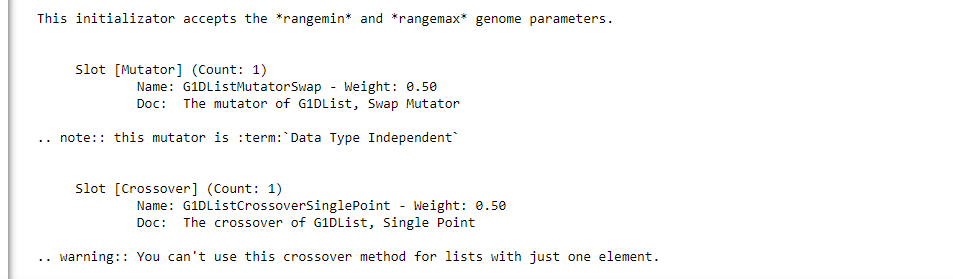


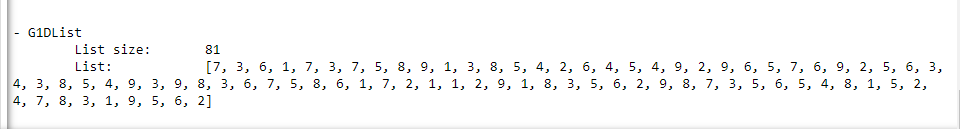


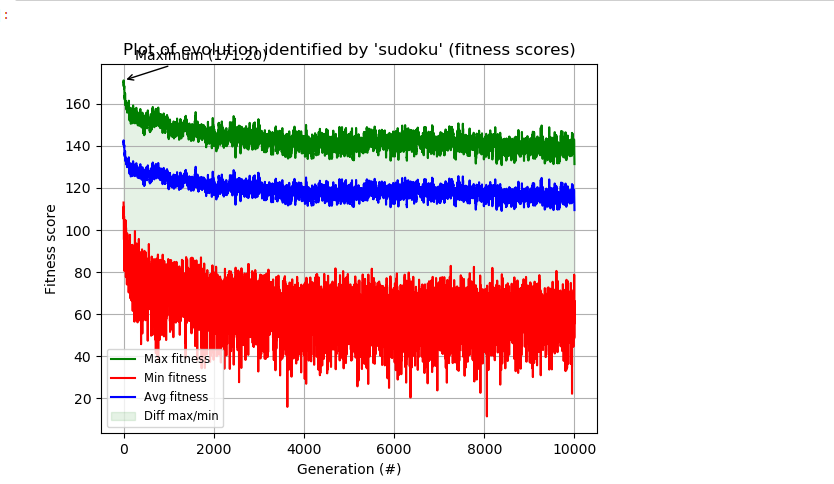


**Ejecución #10**



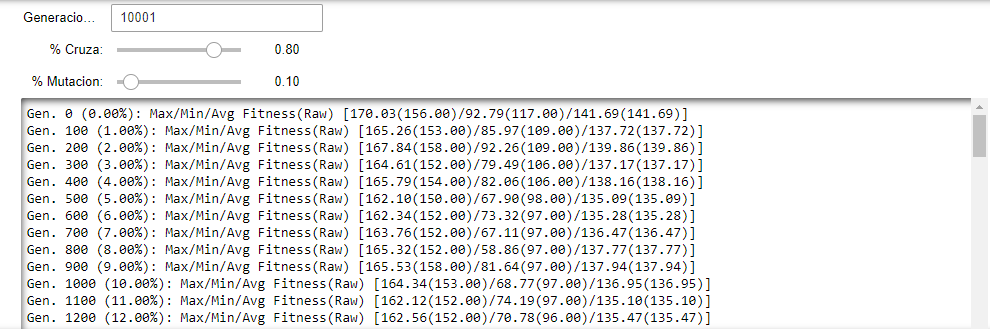


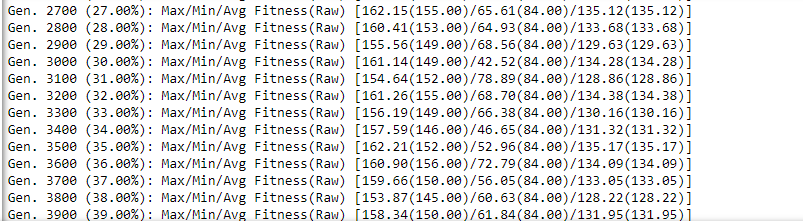
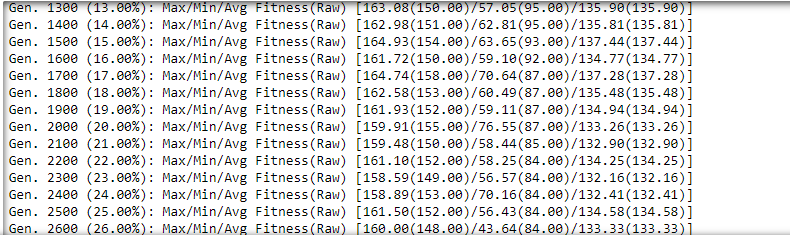


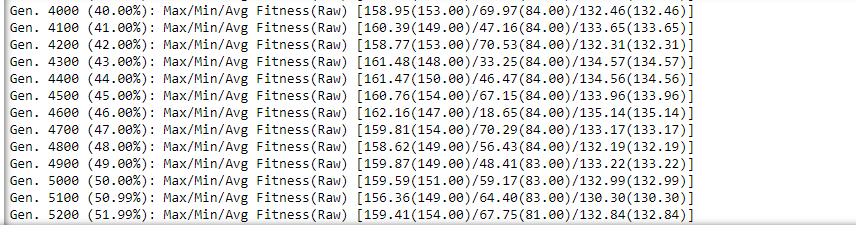


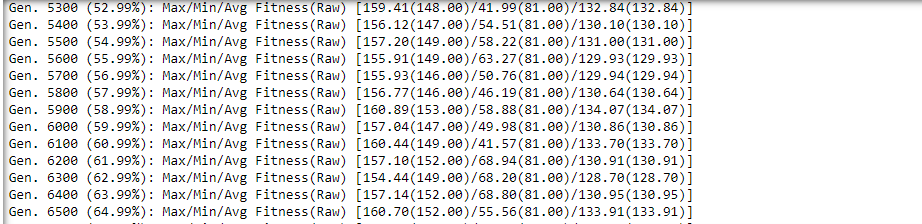
**Realizar ejecución modificando los parámetros, cruza, mutación, numero de generaciones y analizar el resultado obtenido en cada caso. Reflexiones sobre el impacto de cada operador en el desempeño del AG y escribe tus conclusiones.**

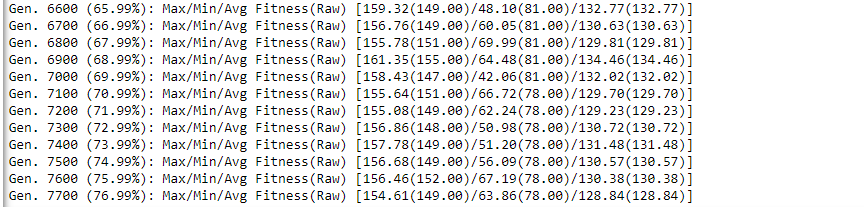
1. **Ejecución**

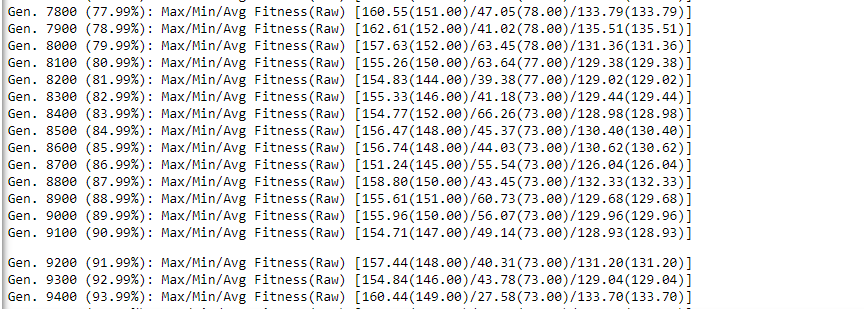


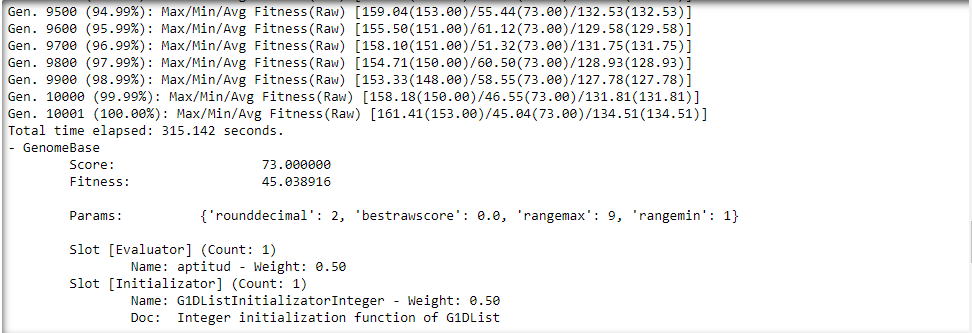


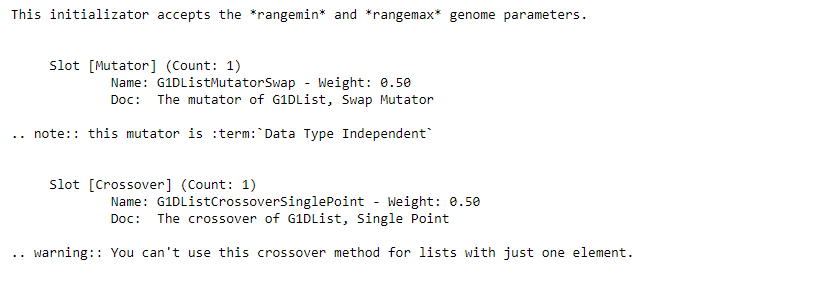


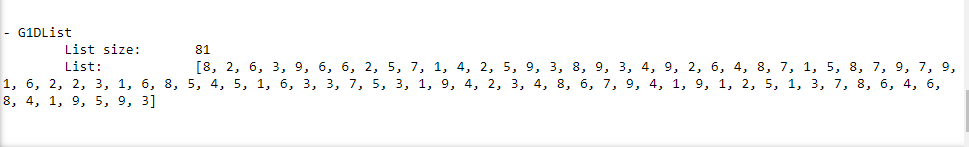


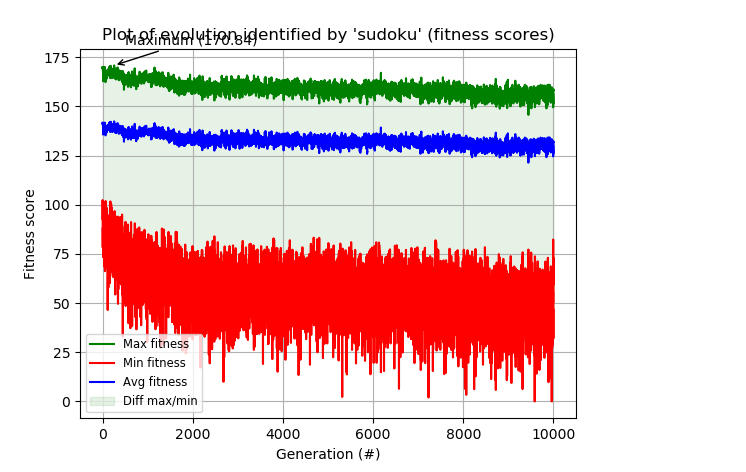










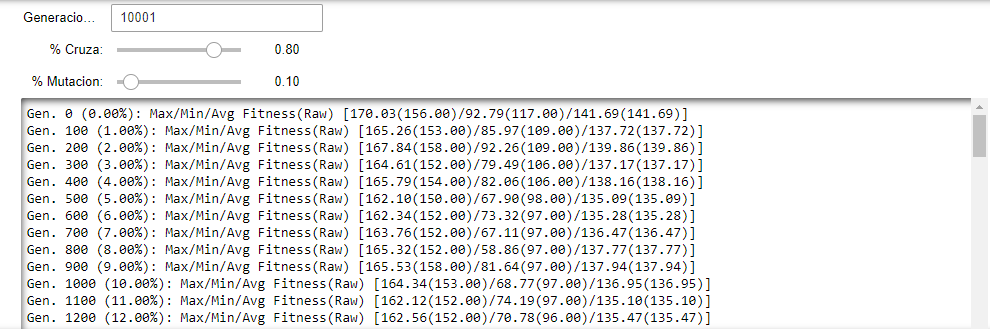


1. **Resultado**

Como resultado, obtuvimos que a como aumentaban las generaciones, disminuía el porcentaje de error, en este caso, la función de aptitud. Para ello, en la grafica del punto 1, muestra tres elementos fundamentales que debemos considerar para interpretar correctamente los datos, que son: el máximo, mínimo y el promedio que se esta teniendo en cada generación. Con un tiempo total de **315.142 segundos** ha terminado en resolver el algoritmo.

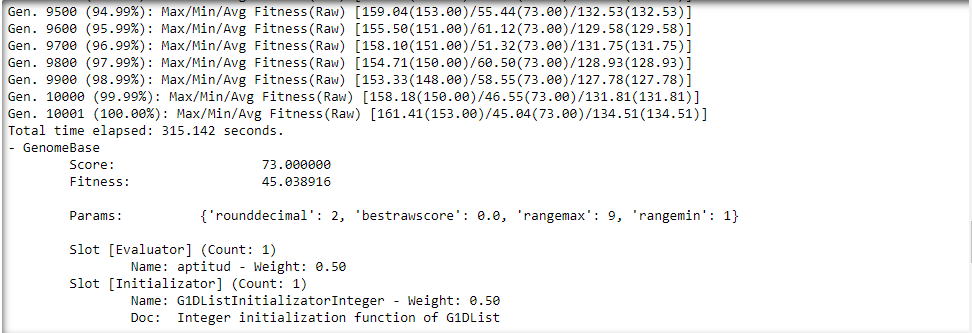
1. **Reflexión**

En las 10 ejecuciones que corrimos siempre llegaba a un **score de 64** aproximado en cada corrida, en este caso, nunca sobrepasaba el límite de 64 en cada una de las iteraciones. En cambio, en la **corrida 11**, en este caso, viene siendo la que nosotros modificamos los parámetros (cruza, mutación, numero de generaciones), que son los siguientes parámetros proporcionados:



A partir de esos parámetros, nos arrojó lo siguiente:





Cabe mencionar, que en la última iteración que llegó, no ha resuelto en 100% en su totalidad el sudoku.

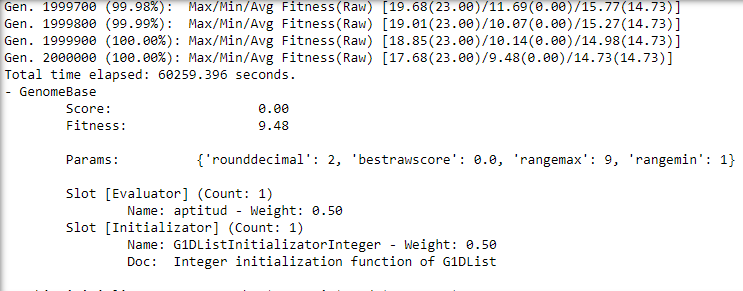
A diferencia de las **10 corridas** que se ejecutaron, se determinó que en la corrida 11, **el score fue 73**, quiere decir quetuvo más margen de error al ejecutarse. Además, la función de aptitud resulto con un **45.03.**

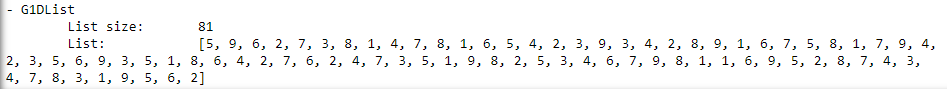
Por lo tanto, cuando reducimos la cruza y aumentamos la mutación, nos pudimos dar cuenta que aumenta el margen de error.

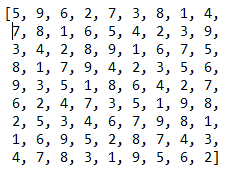
Después decidimos hacer dos pruebas más, una con 1 millón de generaciones y otra con 2 millones de generaciones y estos fueron los resultados:

1 millon(panocho):

2 millones:





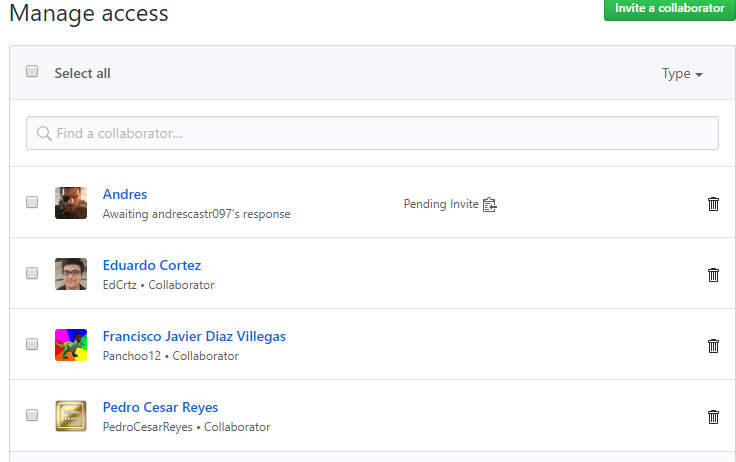


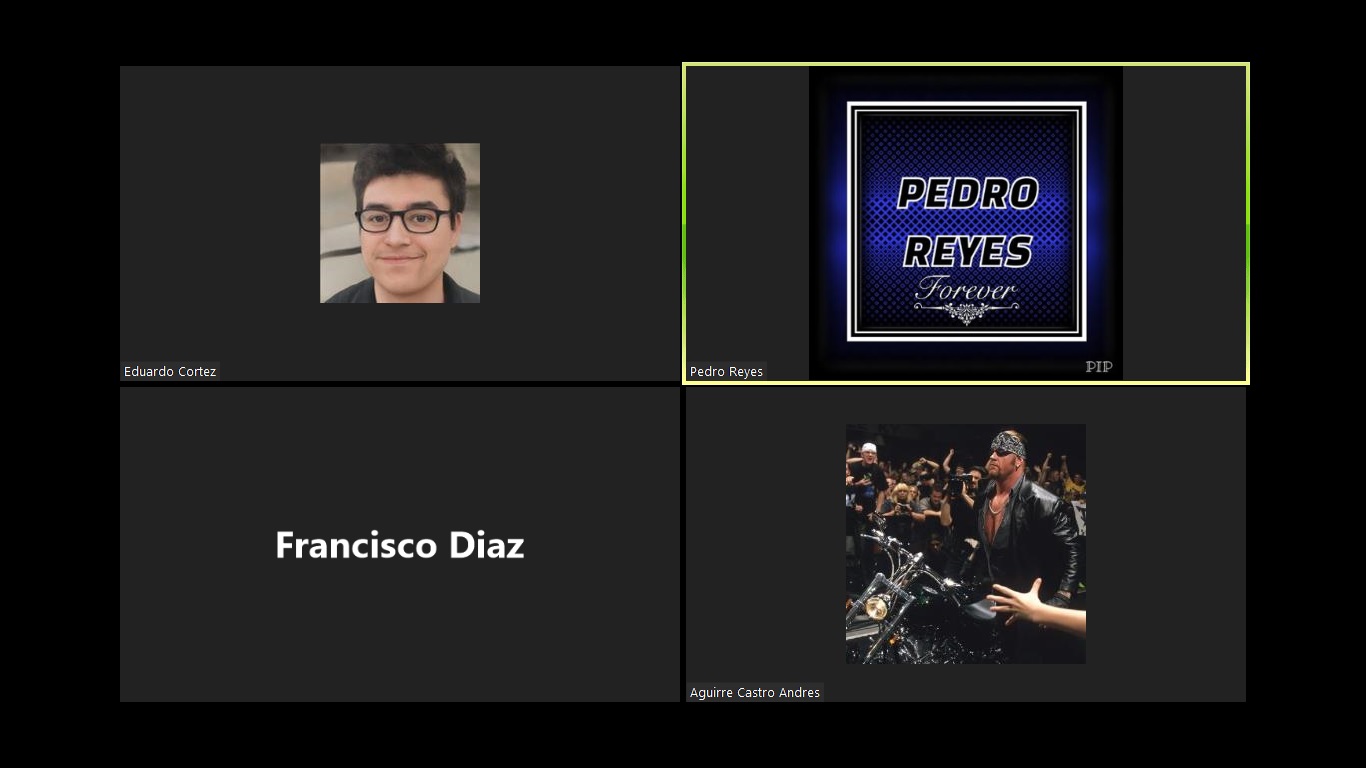
Aquí podemos ver que el sudoku se ha resuelto satisfactoriamente

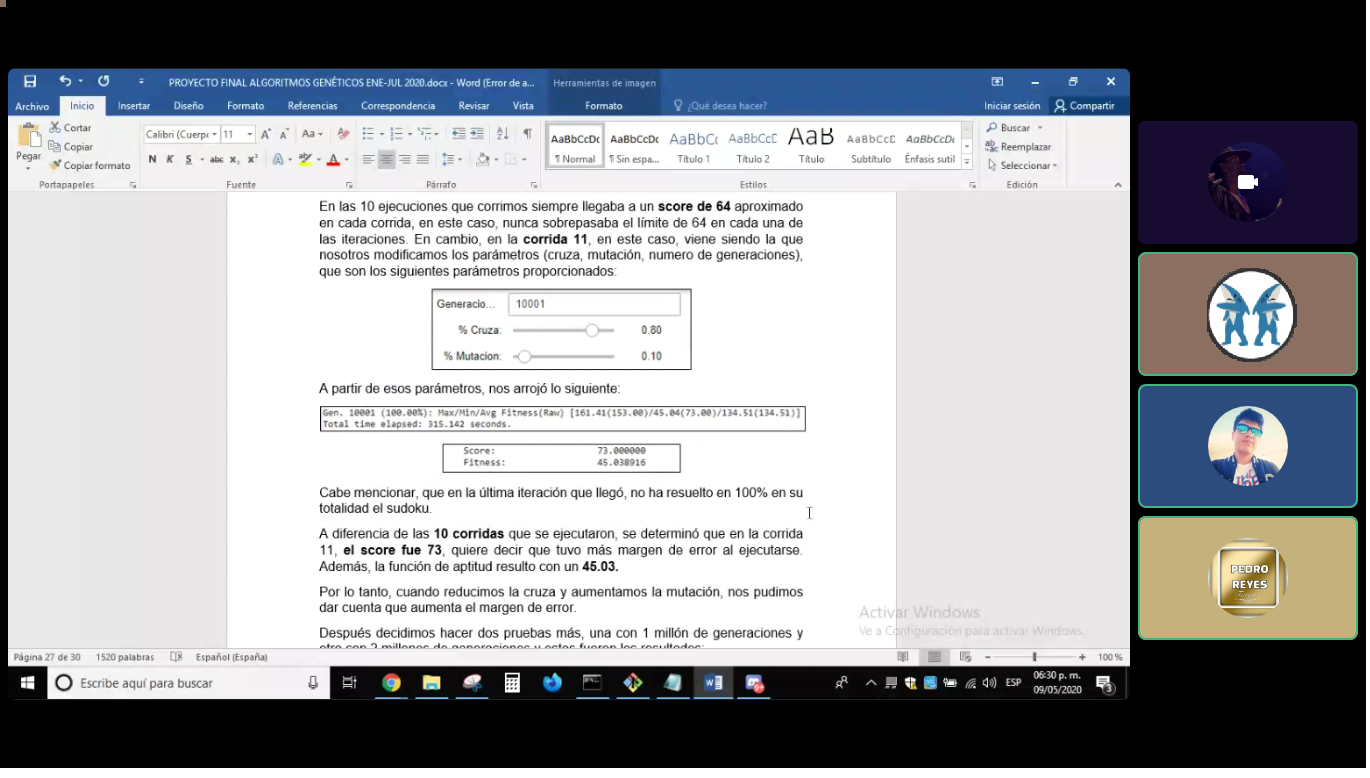
1. **Conclusión**

**Nos gusta la verga**

**ANEXOS**

Anexo 1: Colaboradores del control de versiones (git, github)

Anexo 2: Sala de conferencias para trabajar en equipo: Zoom

Anexo 3: Sala de conferencias para trabajar en equipo: Discord