**Instituto Politécnico Nacional**

Escuela Superior de Cómputo

**Inteligencia artificial**

Práctica 1

Integrantes:

* Vásquez Enríquez Amaury
* Ramírez Aguilar Marc Alexander
* Cornejo David

# Introducción

En la cotidianidad de las personas existen diversos problemas que pueden generar ciertas consecuencias en su entorno, desde generar algunas dudas con respecto a algún tema, evento o acción, hasta situaciones que se presentan como una dificultad para un proceso. Gracias a la programación es que se han logrado resolver varios de estos y han generado mejoras en el desarrollo de varias actividades. Sin embargo, existe la incógnita sobre si son las únicas soluciones que pueden darse, permitiendo idear más posibilidades de resolver un problema e incluso si hay la forma de resolverlo reduciendo a la vez recursos y haciéndolo más eficaz.

Es derivado de esto que surge la inteligencia artificial, una forma de resolver situaciones en donde máquinas automaticen sus soluciones a de manera óptima. Son algoritmos que parten desde un estado inicial como punto de referencia para realizar búsquedas a diversas soluciones posibles a la problemática en donde son planteados, localizando cuál de todas estas es la que mayormente beneficia a los usuarios y recursos implementados. A la solución que llegue a darse por este algoritmo se le denomina “objetivo”, el cual se arroja gracias a un modelo que implementa dicho algoritmo de búsqueda.

Dentro del desarrollo de estos modelos surgen algunas preguntas que permiten identificar qué tan útil es hacer uso de una inteligencia artificial para la solución de un problema. Éstas son: Si hay una solución ¿la encuentra?, ¿Es la solución óptima? ¿Qué tan compleja es el desarrollo para la solución?

Estas preguntas hacen referencia sobre si se puede hallar la solución al problema donde se plantea el modelo, sobre si la solución que se está encontrando es la que menos recursos utiliza y del tiempo y espacio que consume la búsqueda de la solución.

No necesariamente al computar un algoritmo se le reconoce como modelo de inteligencia artificial, ya que solo se está resolviendo de manera manual el mismo. El objetivo de la siguiente práctica es demostrar esa parte mediante la solución de problemas pequeños mediante programación, donde solo se realicen búsquedas sencillas y nada “inteligentes” por cuenta propia. Esto con el objetivo de comprender cómo sería la solución de diversas situaciones sin el uso de esta, así como el análisis de soluciones de búsqueda simples como una introducción a la rama de la programación.

# Desarrollo

## Planteamiento del problema

Se solicita realizar las siguientes actividades:

1. Desarrolla una versión tonta del juego del gato (tic tac toe)

* Deberá ofrecer la posibilidad de jugar humano vs computadora y computadora vs computadora.
* La selección de movimientos la hará de forma aleatoria y no se permitirá sobrescribir una jugada previa.
* Deberán validarse las combinaciones ganadoras y de empate.

1. Desarrolla un programa que mediante búsqueda aleatoria intente resolver el siguiente sistema de ecuaciones:

Texto, Carta

Descripción generada automáticamente

Lo anterior implementado en código en cualquier lenguaje de programación y entorno de desarrollo deseado.

## Ejercicio 1

Se solicita desarrolla un programa que genere una versión tonta del juego del gato, es decir, que no tenga la capacidad de analizar posibles siguientes movimientos del contrincante, haciendo las selecciones del lado del computador de manera aleatoria. Se crearán dos versiones de este, donde una sea jugador contra computadora y otra computadora contra computadora. Este programa debe ser capaz de detectar las 8 posibles maneras de ganar en el juego (2 cruzadas, tres en fila y tres en columna).

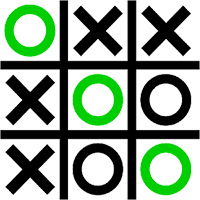


Ilustración El juego de Tic Tac Toe (o juego del gato) es uno de los juegos más populares a nivel global.

## Ejercicio 2

Se solicita desarrollar un programa que resuelva el sistema de ecuaciones de la **figura** mediante búsqueda aleatoria:

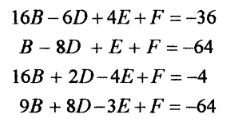


Ilustración Sistema de ecuaciones planteado.

La propuesta de solución fue la siguiente:

Captura de pantalla de computadora

Descripción generada automáticamente

Ilustración Código fuente parte 1.

Texto

Descripción generada automáticamente

Ilustración Código fuente parte 2.

Texto

Descripción generada automáticamente

Ilustración Código fuente parte 3.

El código anterior fue desarrollado con Python, en el editor de código ‘Visual Studio Code’, el cual posee extensiones para analizar con facilidad errores sintácticos que pueden presentarse en el programa.

### Modulo Random

Para la generación de números aleatorios se utilizó un módulo llamado ‘Random’, el cual contiene estructuras que pueden generar números enteros de manera aleatoria al definir un rango de limitación.

Captura de pantalla de computadora

Descripción generada automáticamente

Ilustración Importación del módulo Random de Python.

### generateRandom

Al ser números los que buscamos como datos de entrada para la solución, se requieren tener almacenadas en alguna parte. Es por ello por lo que se crea la función ‘generateRandom’ donde se declaran las 4 variables del sistema de ecuaciones a partir de la instancia a ‘randomint’ del módulo Random importado, retornando como resultado valores aleatorios almacenados en las variables B, D, E y F.

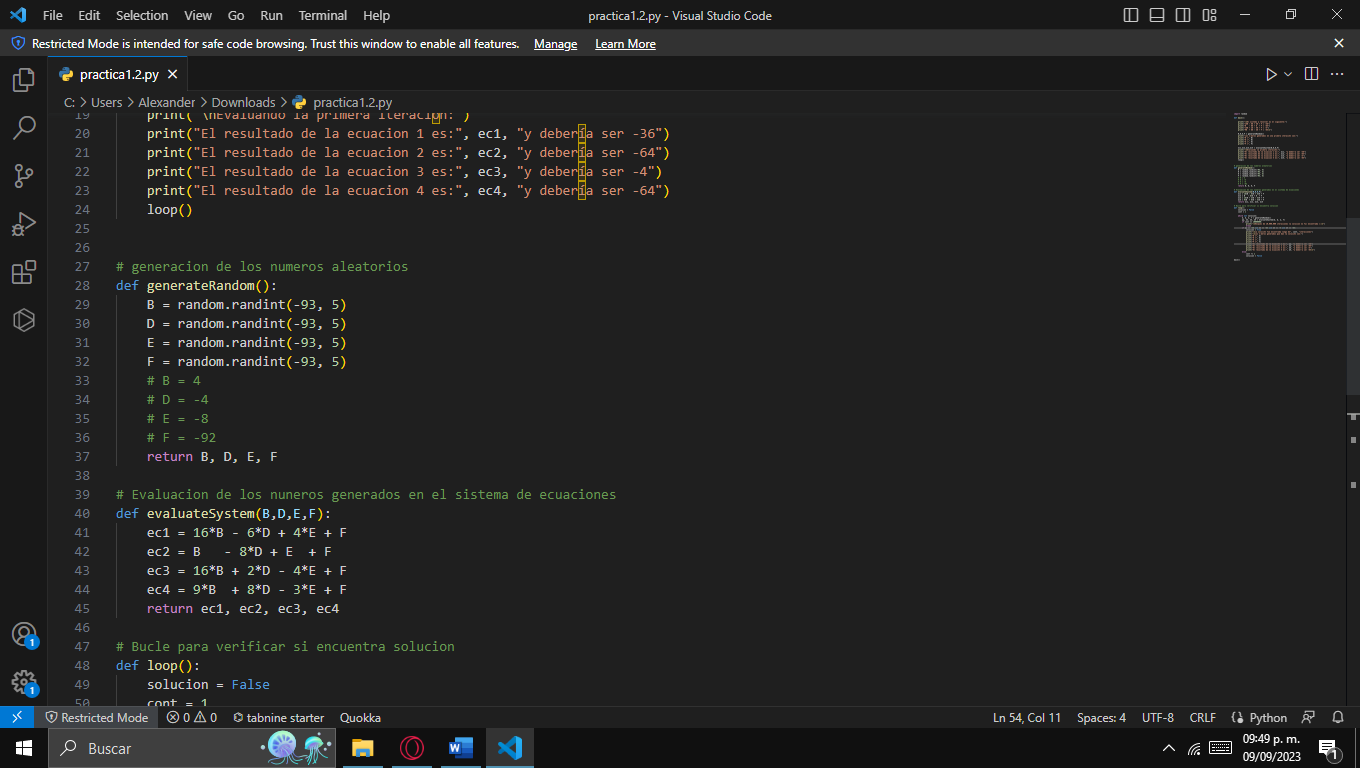


Ilustración Función generateRandom.

### evaluateSystem

El sistema de ecuaciones puede verse como un conjunto de igualdades, por ello se declaran dentro de una función llamada ‘evaluateSystem’, donde el resultado se guardará en variables que retornarán como salida.

Captura de pantalla de computadora

Descripción generada automáticamente

Ilustración función evaluateSystem.

### loop

Tomando en cuenta el caso en el que los números aleatorios que se requieren para el sistema de ecuaciones no se encuentren en la primera iteración que realice el programa, se considera que debe utilizarse un bucle que repita las iteraciones hasta hallar la solución al sistema de ecuaciones. Considerando el caso en el que no se encuentre la solución del problema, se determina un limite de 10 millones de iteraciones para encontrarla, de lo contrario, se detendrá este proceso de iteración y se retornará un mensaje que indica que no se encontró la solución. Para hacer el proceso de búsqueda se llaman a las funciones anteriormente descritas en generateRandom y evaluateSystem dentro de la función ‘loop’, que ejecutará las mismas hasta hallar una solución a las ecuaciones.

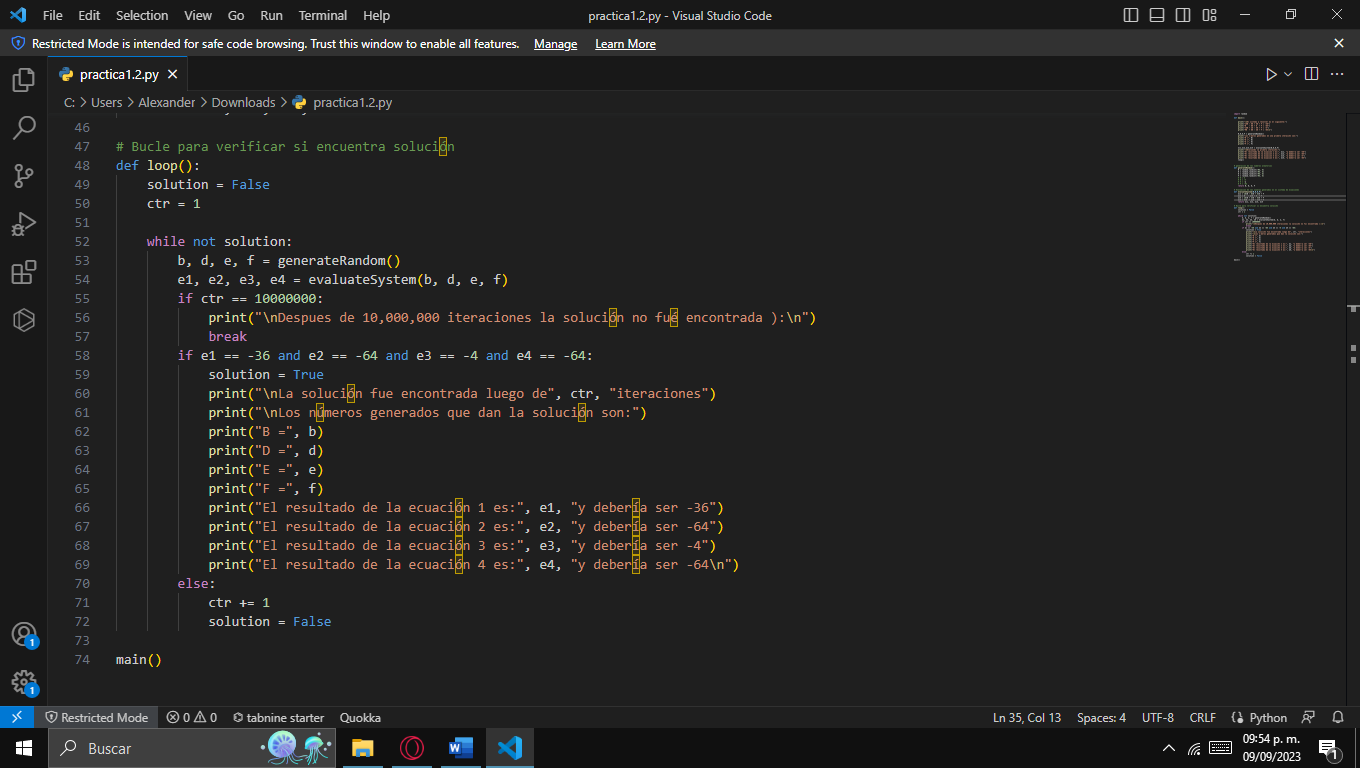


Ilustración función loop.

### main

Para el funcionamiento del programa se deben ejecutar las funciones por el intérprete, sin embargo, estas funciones se encuentran creadas de manera independiente de alguna estructura como métodos, por lo que no existe algún orden de cómo puedan ejecutarse. Para poder tener una correcta salida en el intérprete a la hora de depurar el código, se crea una función principal, donde se llaman las funciones escritas para resolver el problema. Esta función llamada ‘main’ comienza con una primera búsqueda aleatoria mediante variables, donde se pretende evaluar el sistema la primera ocasión. En caso de que este falle, se inicia el bucle de iteraciones.

Captura de pantalla de computadora

Descripción generada automáticamente

Ilustración función main.

### Resultado obtenido

Al realizar la compilación del programa se obtiene el siguiente resultado:

Texto

Descripción generada automáticamente

Ilustración Resultado de ejecución.

Se puede observar como en la primera iteración se arrojan valores que no otorgan la solución deseada del problema, pues no cumple con lo solicitado.

Texto

Descripción generada automáticamente

Ilustración Primera iteración realizada.

Dado este caso, se vuelve a realizar la iteración n veces hasta dar con el resultado que resuelva el sistema de ecuaciones. Tras la espera de 6 minutos en la ejecución del programa, se encontró una solución al sistema después de 81,406,835 iteraciones.

Texto

Descripción generada automáticamente

Ilustración Bucle de iteraciones.

# Conclusión

# Referencias