**Practica Final – Curso Python – Buscaminas  
Juan Esteban Salas Flórez**

Por medio de este documento, voy a describir como llegué a la solución del problema planteado que fue desarrollar un videojuego estilo buscaminas para la consola de Python, utilizando como entorno de desarrollo, PyCharm.

**Descripción del problema**

La práctica del buscaminas consiste en crear un videojuego cuyo objetivo es despejar un campo de minas sin detonar ninguna. El juego inicia con todo el campo desconocido. Al elegir una coordenada, se mostrará su información, la cual podrá ser una mina o un numero del 0 al 9 dependiendo de cuantas minas haya a su alrededor.

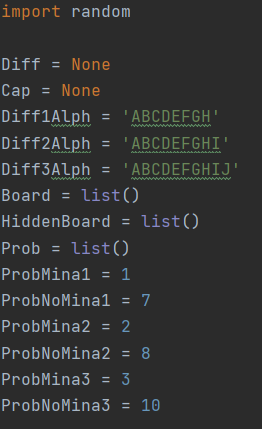
La terminación del juego consiste de dos maneras, si solo quedan por descubrir las casillas donde hay minas, es una victoria. Pero si se descubre una casilla donde hay mina, entonces es una derrota.

Importante:

* El juego tiene 3 niveles de dificultad: fácil, medio y difícil.
* El programa debe crear el tablero al azar asignándole sus minas y pistas desde que se elige la dificultad.
* Para jugar el jugador deberá introducir una coordenada al programa, la cual será la casilla que se mostrará. (ejemplo. A1, G8, G3)

**Solución implementada**

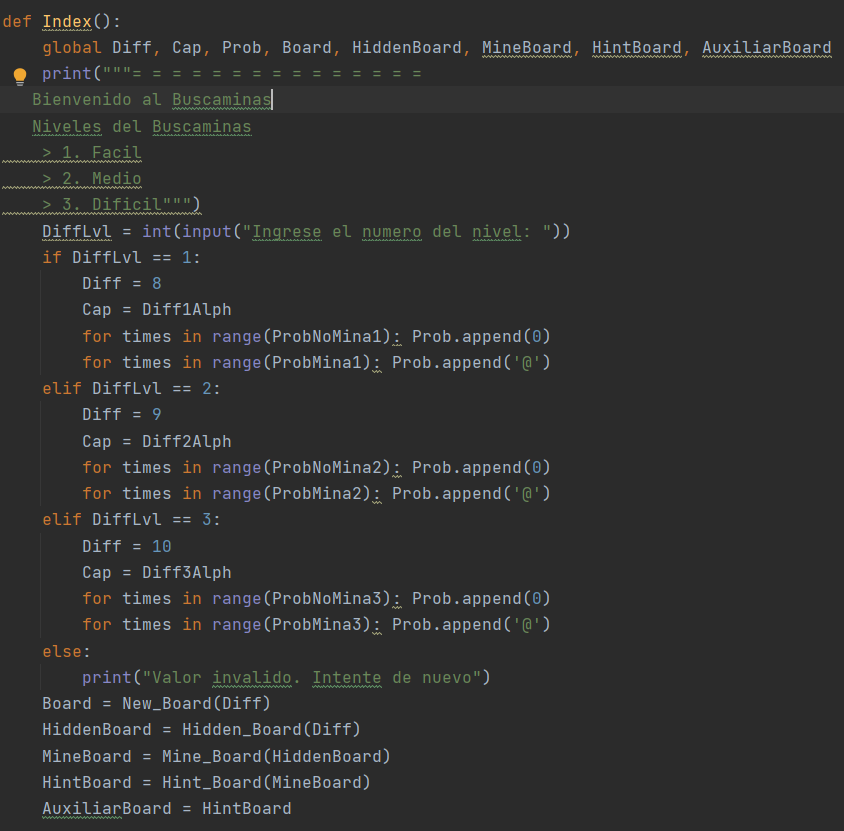
En esta parte voy a explicar cada método aplicado en el código y cual es su funcionamiento con el fin de dar un buen entendimiento a la solución implementada del problema.

****

**Declaración de variables e importación de librerías**

Al entrar a la aplicación primero vemos la importación de la librería random, la cual usaremos para poner las minas de forma aleatoria y también cambiar el numero de minas dependiendo del nivel, para que las partidas sean diferentes.

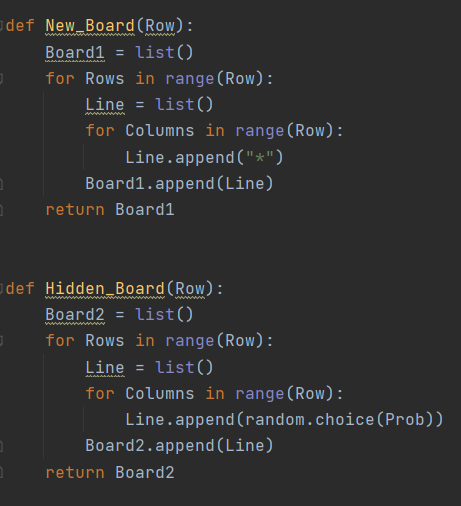
Tambien, vemos declaradas todas las variables globales que serán utilizadas para el funcionamiento del buscaminas, las mas importantes son las 3 de alfabeto, que son diferentes según la dificultad del tablero, las de probabilidad, que dan el numero de minas que tendrá cada nivel y la probabilidad de que no haya una mina en cierta casilla del tablero.

**Método Index()**

Este método tiene como finalidad, mostrar una pequeña interfaz de consola en la cual el usuario interactúa, lo primero que muestra es un mensaje de bienvenida junto con la lista de niveles y un lector de texto, el cual le permite al usuario ingresar la dificultad en la cual desea jugar.

Después de que el usuario haga su elección, se llenan las variables de probabilidad de bombas y no bombas, siendo 0 las no bombas y @ las bombas.

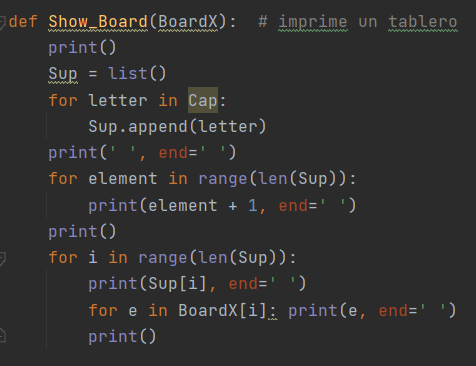
Por último, podemos ver que se llaman varios métodos que garantizan el inicio del juego, los cuales mostrare mas tarde en este mismo documento.



**New\_Board() y Hidden\_Board()**

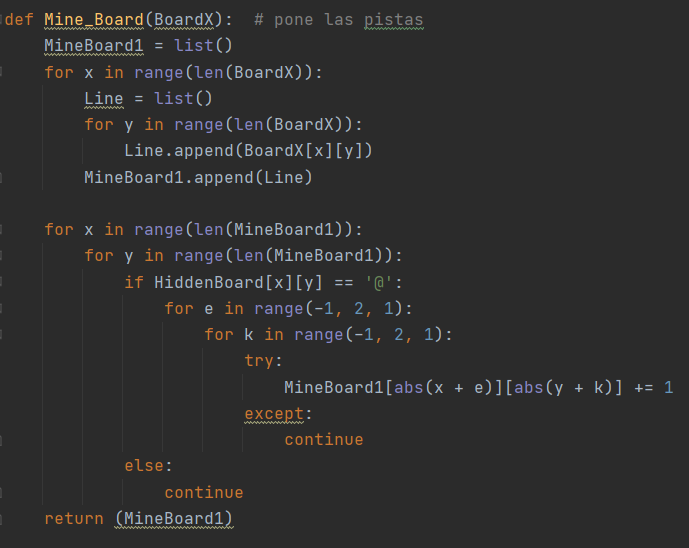
El método New\_Board() crea un tablero vacio, este lo hace por medio del uso de listas, primero crea una lista y luego en cada parte de esa lista pone nuevas listas, con el fin de que quede una matriz del tamaño que se desee dependiendo de la dificultad elegida en el index. Por último, devuelve el tablero lleno de \*’s.

El método Hidden\_Board() crea un tablero con las minas y no minas, es decir, un tablero aparte en el cual podremos verlo lleno de 0’s y @’s, siendo las @’s como dije anteriormente, las minas, usando este tablero, en este tablero se desarrollara el juego de la máquina, sin embargo, este no es mostrado en consola, aún.

****

**Show\_Board()**

Este método es usado para mostrar en pantalla cualquier tablero que sea necesario, para el desarrollo es necesario mostrar los tableros escondidos y verificar que la aplicación esta funcionando bien, sin embargo, en la solución final no es necesario, pero igual usamos el método para mostrar el tablero de juego cada que hacemos un movimiento, puesto que este se actualiza siempre.



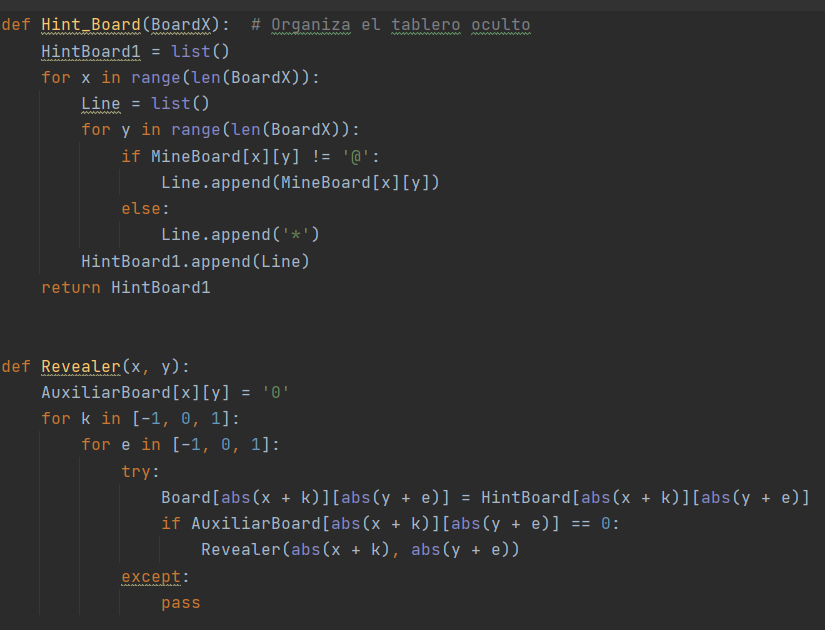
**Mine\_Board()**

Este método usar muchos siclos for, en los cuales se busca recorrer cada casilla del tablero y verificar sus 8 casillas adyacentes, si en una de estas hay una mina, entonces le sumamos un 1 al numero de la casilla que estamos verificando, para llegar a un máximo de 8 si es que la casilla cuenta con 8 minas adyacentes.

Tambien es importante controlar las excepciones, por ejemplo si revisamos las casillas de los lados y esquinas podemos ver que estas no tienen 8 casillas adyacentes, para esto hay otro método.

El método **Hint\_Board()** organiza el tablero que esta escondido, es decir, que usa el tablero normal y escondido, para que las coordenadas que usemos en uno sean las mismas que en el otro y así poder jugar en ambos tableros a la vez.

Y el método mas importante, el **Revealer()** es el encargado del juego, cuando damos una coordenada en la cual no hay minas, esta tiene que revisar si es un numero o un cero, si es un numero simplemente lo revela, pero si es un cero, debe revelar el mismo, junto con todos los ceros que tenga adyacentes, y los ceros adyacentes a esos ceros, generando islas de ceros en pantalla, y por último, mostrar los números que se encuentran cercanos a esos ceros. Es algo complicado por la abstracción del mismo, pero con el uso de dos siclos for y controlando excepciones, este se simplifica mucho, como podemos ver en la foto de abajo.

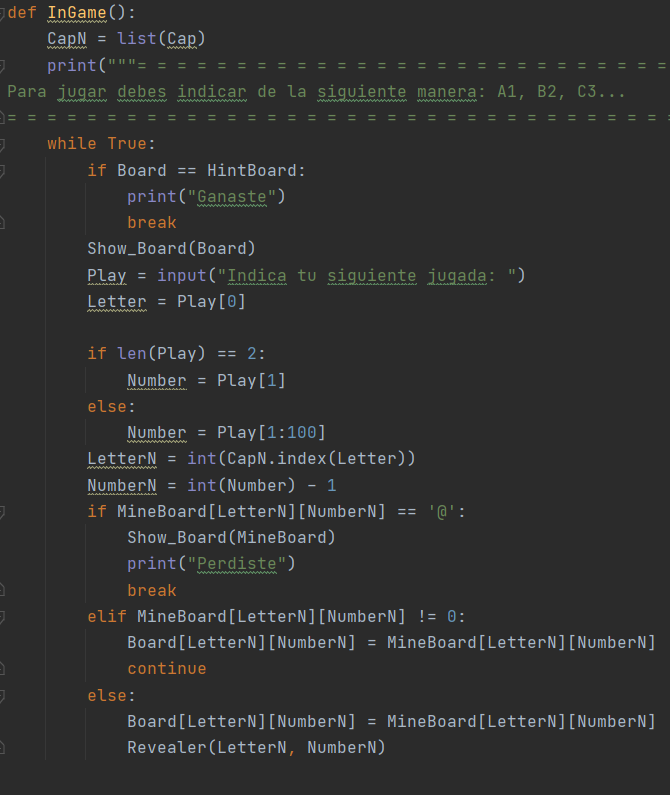
Por último, está el método del juego, **InGame()**.

Este método es el que lee las coordenadas en las cuales el usuario desea revelar la casilla, esto lo hace comparando la parte de la letra del string con la de la variable alfabeto declarada al inicio, y le da un numero dependiendo de su posición en el alfabeto, luego toma el numero del string que se ingreso y lo guarda, con estos dos números ya tenemos unas coordenadas para movernos en el tablero.

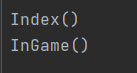
El siguiente paso es revisar si la casilla que se encuentra en esas coordenadas, tiene un numero o una mina, si es una mina, el método se cierra e imprime en pantalla un mensaje diciendo “Perdiste”, luego muestra el tablero completo.

Si la casilla no era una mina, este la revela si es un numero diferente a cero o ejecuta el método del revealer si es un cero.

Todo esto se repite en un ciclo while True, hasta que el usuario pierda o hasta que en el tablero solo queden minas sin revelar, si pasa esto, se imprime un mensaje diciendo “Ganaste”



Al final de todo el código, podemos ver que llamamos al método Index() y al método InGame() para que se del juego completo.



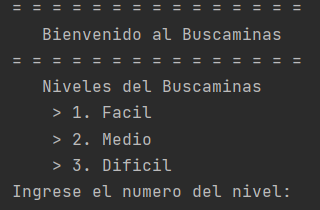
Así fue como solucioné el código, espero que la explicación de cada método haya sido clara y concisa.

**¡Juguemos!**

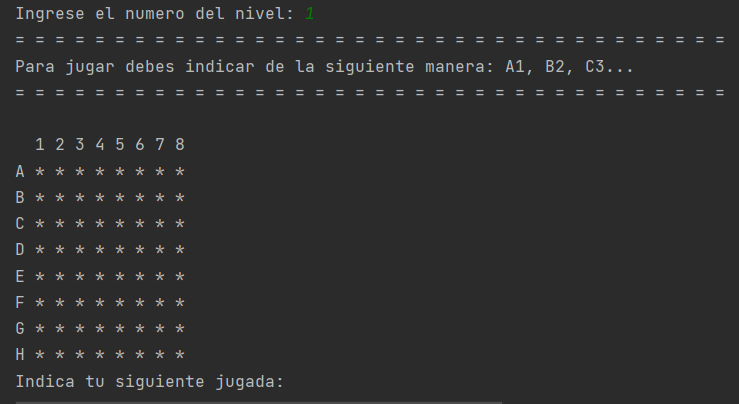
Ahora, voy a mostrar como es el funcionamiento en la consola, con una partida ganada y una en la que se toca una mina:

**Partida Ganada.**

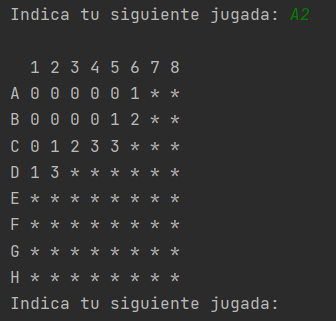
Al ejecutar el programa, en consola podemos ver esta interfaz:



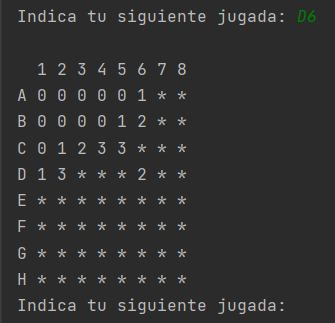
Seleccionamos la dificultad que queramos escribiendo el numero de esta, en este caso, será fácil.



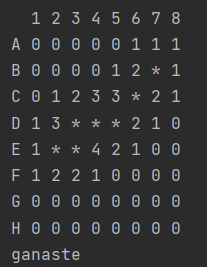
Se nos muestra un mensaje de como debemos poner las coordenadas, y luego aparece el tablero vacio que debemos ir despejando para ganar, la primera jugada que haremos será A2.



Vemos que se despejo una isla de ceros, la siguiente jugada será D6.

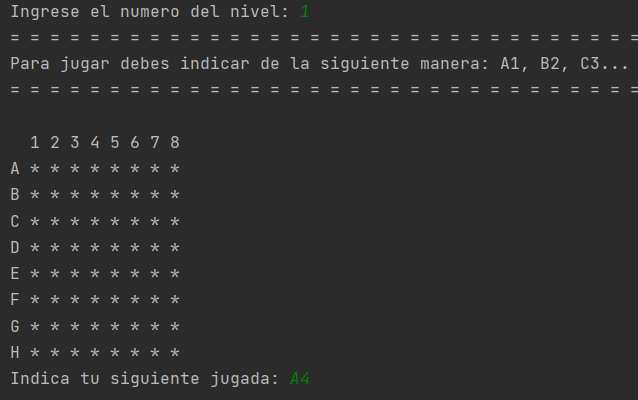


Y así sucesivamente hasta que despejamos todas las minas y tenemos un mensaje de victoria.



**Partida perdida.**

Ahora veamos un ejemplo en el cual perdemos, la primera jugada que haremos en este tablero es A4.



Como aquí había una mina, se muestra el tablero despejado, y un mensaje que dice “Perdiste”. Luego se acaba la ejecución.

