**Manual Técnico**

**INTRODUCCION:**

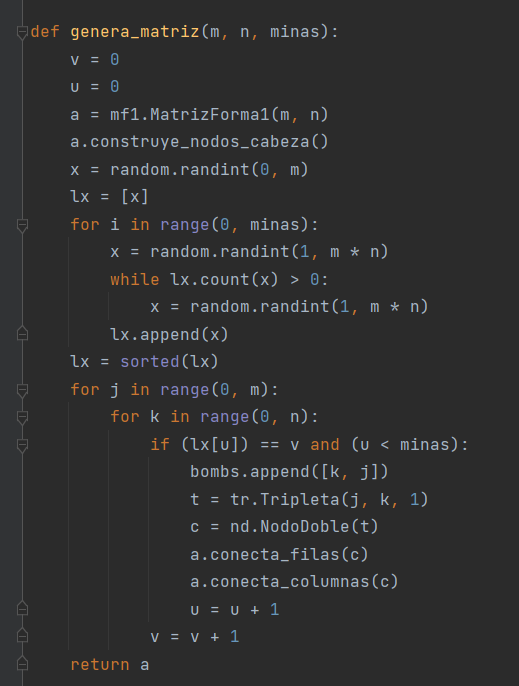
Nuestro Buscaminas es una aplicación diseñada para jugar el clásico juego en 4 dificultades diferentes que incluyen una personalizada. Buscaminas es perfecto para mostrar la utilidad que tiene manejar las matrices dispersas como listas ligadas usando tripletas para las posiciones de las mismas. Además, gracias a la interfaz gráfica de la aplicación, la jugabilidad es definitivamente más sencilla.

**DESARROLLO:**

Esta aplicación fue desarrollada principalmente en Python con el IDE PyCharm, para sentar las bases operativas de la misma, se crearon cuatro clases distintas con diferentes funcionalidades las cuales son **Programa, MatrizEnForma1, Tripleta, NodoDoble**. Para un mejor entendimiento vamos a definir cada una:

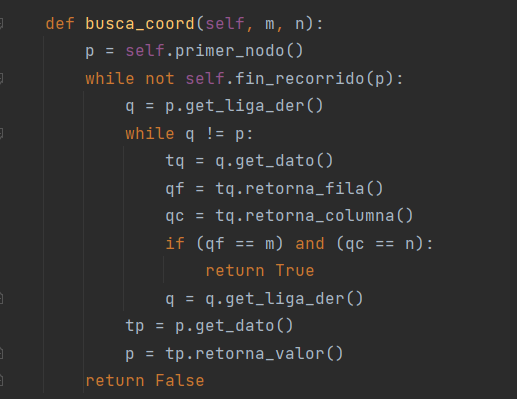
* **Clase Program:** es la clase principal en la que nos encontramos con una pequeña interfaz que se ejecuta en consola para saber cuales son las acciones que desea realizar el usuario, pero lo mas importante de esta clase es que al momento de ejecutar busca el archivo que contiene el laberinto el cual el usuario desea resolver y lo envía a la clase laberinto para poder ser usado.
* **Clase MatrizEnForma1:** esta clase es la que sirve para manejar la matriz dispersa que funciona como el mapa que indica en que casillas del tablero de juego se encuentran las minas. Contiene los métodos que recorren la matriz como lista ligada además de los que construyen la matriz para que esté representada de la forma 1 vista en clase con anterioridad.
* **Clase Tripleta:** en la clase tripleta se crean los datos que van a ser almacenados en los nodos dobles que conforman a las matrices en forma 1 que se utilizan para representar las coordenadas del tablero. Tal cómo su nombre indica, esta clase se compone de tres atributos que significan respectivamente, la fila, la columna y el valor contenido en una coordenada especifica de la matriz.
* **Clase NodoDoble:** la clase nodo doble, es la que permite la creación de los nodos que componen las matrices en forma 1. Sus atributos son simplemente liga izquierda, liga derecha y dato. Este ultimo se utilizará principalmente para almacenar las tripletas de las coordenadas de la matriz dispersa que no sean iguales a cero.

Para el comienzo de un nuevo juego, el programa primero se propone la creación de un tablero de juego con condiciones específicas que dependen de la elección del jugador por medio de los botones respectivos en los que se puede escoger la dificultad del juego o bien establecer una dificultad personalizada. Esto es posible gracias al método **genera\_matriz().**



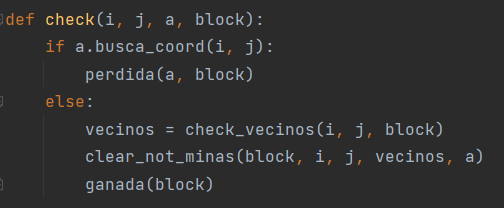
En este mismo método, que recibe como parámetros las dimensiones de la matriz o tablero y la cantidad de minas que serán plantadas, se genera una lista de números aleatorios diferentes que se encuentren dentro del límite que se consigue al multiplicar las dimensiones de la matriz. Una vez hecho esto, utilizando dos ciclos se crean los objetos tripleta que se guardarán en la matriz forma 1 en las posiciones de la lista aleatoria que se generó anteriormente.

Y a partir de ese método, los que se encargan de controlar las situaciones posibles son más sencillos de elaborar debido a que trabajan apoyandose en la matriz en forma 1 que ya se generó. Otro método que resulta sumamente importante para nuestro programa es el de **busca\_coord(),** que como explica su nombre, recibe como parametro una coordenada y busca si esta está almacenada en la matriz forma 1 que lo invocó:

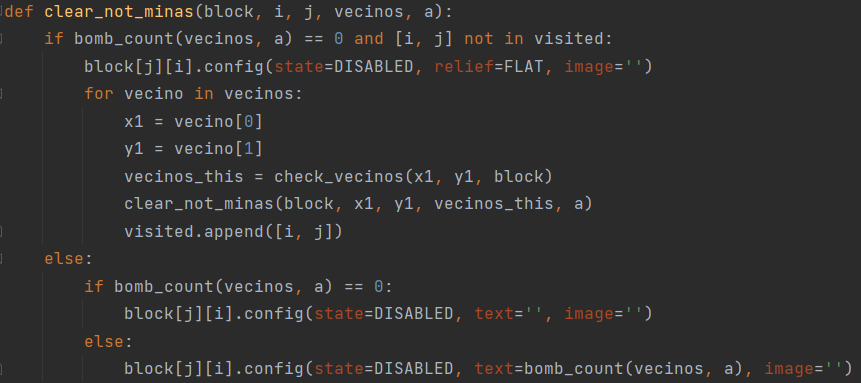


Sencillamente, este metodo recorre la matriz en forma 1 en busca de una tripleta que contenga las coordenadas ingresadas como parámetro para así confirmar si en esa posición se encuentra una mina detonable.

Luego, mediante el método **check()** que se activa cada vez que se presiona un botón del tablero que a su vez posee coordenadas específicas, sólo debe utilizar el metodo anterior para confirmar si el boton presionado llevaría a un game over o si el jugador puede seguir jugando su partida.

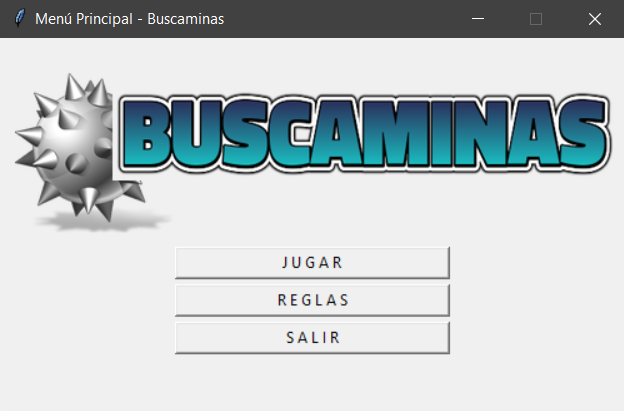


Es aquí donde el funcionamiento del programa se torna interesante. Por medio de los metodos **check\_vecinos()**, **clear\_not\_minas()** y **bomb\_count()** el programa se encarga de que en caso de hacer clic en una casilla del tablero que no tenga ninguna mina a sus alrededores. El primero de ellos, **check\_vecinos()**, sólo crea una lista con los vecinos de la casilla en cuestión, mientras que **bomb\_count()** es un valor entero que representa el numero de vecinos en la lista que contienen una mina asociada. Finalmente, **clear\_not\_minas()** es un método recursivo que se encarga de deshabilitar las casillas que no tienen minas en sus alrededores hasta encontrar los bordes o casillas que por otro lado si tengan al menos una mina a su alrededor.



**FUNCIONAMIENTO:**

Al ejecutar el programa tenemos esta interfaz de usuario, en primer lugar está la opción de jugar que más adelante habilitará otros botones que sirven para seleccionar la dificultad del juego, luego el botón reglas que lleva a otra ventana en la que se pueden encontrar breves instrucciones de acerca del funcionamiento del juego. Finalmente, el botón salir es el que permite salir del programa y terminar su ejecución.

****