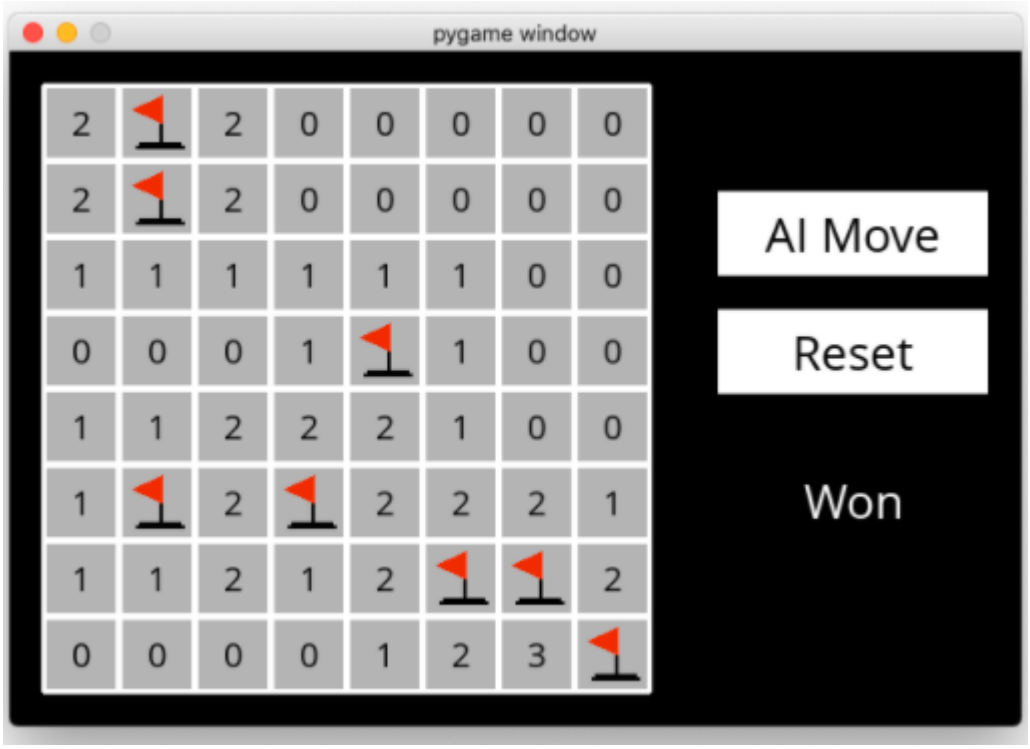
Minesweeper

Escriba un programa de inteligencia artificial para jugar minesweeper.



Contexto

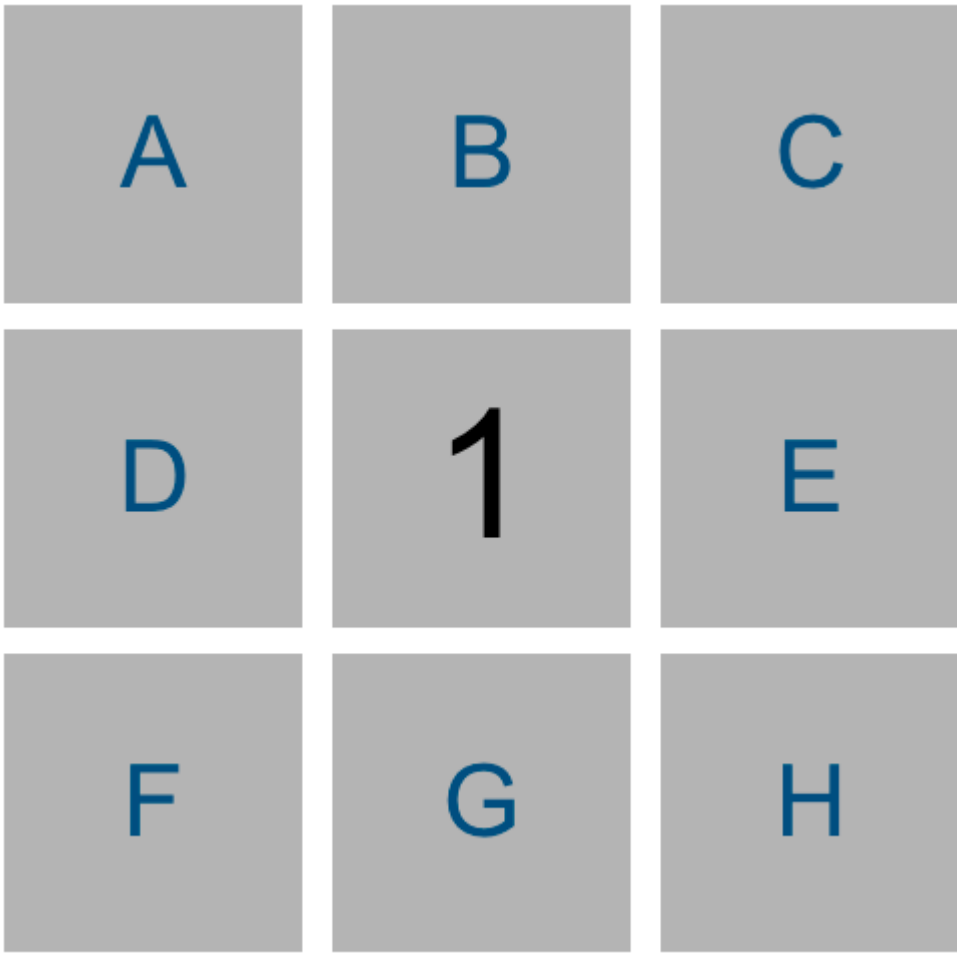
Minesweeper es un juego que consiste de una matriz de celdas, algunas de las cuales contienen minas ocultas. Hacer clic en una celda que contiene una mina hace que ésta se detone y el jugador pierde. Hacer clic en una celda segura revela un número que indica cuántas celdas vecinas tienen una mina. Las celdas vecinas son las que se encuentran a la izquierda, derecha, arriba, abajo o diagonal a una celda.

En este juego de 3x3, por ejemplo, los tres valores de 1 indican que cada una de estas celdas tienen un vecino que contiene una mina. Los cuatro 0 indican que esas celdas no tienen vecinos con minas.

Dada esta información un jugador puede concluir que hay una mina en la celda de la derecha en la parte baja y que no hay minas en la parte izquierda superior.

El objetivo del juego es identificar con una bandera cada una de las minas. En la implementación de este proyecto, el jugador puede poner una bandera con un clic derecho en una celda.

En este proyecto se usará lógica proposicional para crear una base del conocimiento sobre la que se realizarán inferencias. Cada celda es una variable proposicional que tiene valor de verdadero si una celda contiene una mina y es falsa en caso contrario. La IA conocerá cada vez que una celda segura es accedida y verá el número para esa celda. Considere el siguiente tablero de minesweeper, donde la celda del medio ha sido revelada y las otras han sido etiquetadas con una letra para identificarlas.



¿Qué información se tiene? Se conoce que una de las 8 celdas vecinas tiene una mina. Por lo tanto se puede escribir una expresión lógica de la siguiente forma:

```
Or(A, B, C, D, E, F, G, H)
```

Que se puede expresar de la siguiente forma:

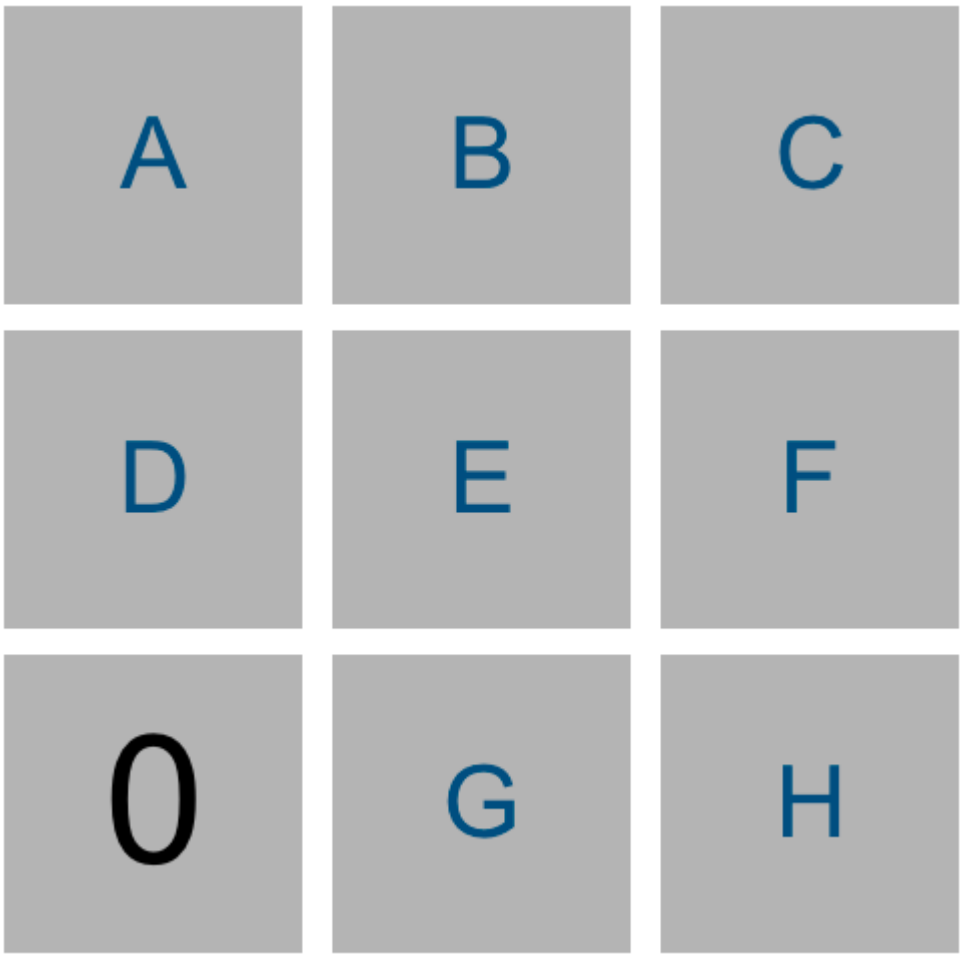
```
Or(
   And(A, Not(B), Not(C), Not(D), Not(E), Not(F), Not(G), Not(H)),
   And(Not(A), B, Not(C), Not(D), Not(E), Not(F), Not(G), Not(H)),
   And(Not(A), Not(B), C, Not(D), Not(E), Not(F), Not(G), Not(H)),
   And(Not(A), Not(B), Not(C), D, Not(E), Not(F), Not(G), Not(H)),
   And(Not(A), Not(B), Not(C), Not(D), E, Not(F), Not(G), Not(H)),
   And(Not(A), Not(B), Not(C), Not(D), Not(E), F, Not(G), Not(H)),
   And(Not(A), Not(B), Not(C), Not(D), Not(E), Not(F), G, Not(H)),
   And(Not(A), Not(B), Not(C), Not(D), Not(E), Not(F), Not(G), H)
)
```

Sin embargo, son expresiones muy complejas que pronto se pueden volver muy largas de comprobar computacionalmente. En lugar de esta representación se puede representar esta sentencia de la siguiente manera:

```
{A, B, C, D, E, F, G, H} = 1
```

Donde la representación tiene dos partes: un grupo de celdas y un conteo que representa cuántas de estas celdas tienen minas. Esta sentencia lógica dice que de las celdas A, B, C, D, E, F, G, and H una de ellas es una mina.

En el tablero:



Se tendría la expresión `{D, E, G} = 0` que dice que de las celdas D, E y G exactamente 0 son minas.

Si conocemos `{A, B, C} = 2`, aún no sabemos lo suficiente para concluir nada. Pero si nos dicen que C es segura, podemos remover C de la sentencia y tener: `{A, B} = 2`, que nos permite obtener nuevas conclusiones.

Igualmente si se dice que C es una mina, se puede decir que `{A, B} = 1`, y se puede inferir que si dos de A,B y C son minas y C es una mina entonces A o B son una mina.

Procedimiento

Descargar el código para este programa. Correr pip3 install -r requirements.txt para instalar pygame que es el paquete de Python que se requiere (si aún no lo ha instalado).

Hay dos archivos main para este proyecto runner.py y minesweeper.py. minesweeper.py contiene toda la lógica del juego y runner.py contiene el código para la interface gráfca. Se deben completar las funciones requeridas en minesweeper.py para poder jugar.

Abrir el código de este programa y poner comentarios para demostrar que se entiende el papel de cada clase. Desarrollar las funciones add_knowledge, make_safe_move, and make_random_move.