1. Descargar archivo tictactoe y descomprimirlo
2. Una vez en el directorio del proyecto, ejecute pip3 install -r requirements.txt para instalar el paquete Python requerido (pygame) para este proyecto.

Previo:

Hay dos archivos principales en este proyecto: runner.py y tictactoe.py. tictactoe.py contiene toda la lógica para jugar el juego y para realizar movimientos óptimos. runner.py ya se ha implementado y contiene todo el código para ejecutar la interfaz gráfica del juego. Una vez que hayas completado todas las funciones requeridas en tictactoe.py, ¡deberías poder ejecutar runner.py de Python para jugar contra tu IA!

Abramos tictactoe.py para comprender lo que se proporciona. Primero, definimos tres variables: X, O y EMPTY, para representar posibles movimientos del tablero.

La función ***initial\_state*** devuelve el estado inicial del tablero. Para este problema, se representar el tablero como una lista de tres listas (que representan las tres filas del tablero), donde cada lista interna contiene tres valores que son X, O o EMPTY. ¡Lo que sigue son funciones que debes implementar

Especificación

Completa las implementaciones de player, actions, result, winner, terminal, utility, and **minimax**.

* La función de player debe tomar un estado del board como entrada y devolver el turno de qué jugador es (X u O).
  + En el estado inicial del juego, X obtiene el primer movimiento. Posteriormente, el jugador se alterna con cada movimiento adicional.
  + Cualquier valor de retorno es aceptable si se proporciona un tablero terminal como entrada (es decir, el juego ya terminó).
* La ​​función de actions debe devolver un set de todas las acciones posibles que se pueden realizar en un tablero/ board determinado.
  + Cada acción debe representarse como una tupla **(i, j)** donde **i** corresponde a la fila del movimiento (0, 1 o 2) y **j** corresponde a qué celda de la fila corresponde al movimiento (también 0, 1 o 2).
  + Los movimientos posibles son todas las celdas del tablero que aún no tienen una X o una O.
  + Cualquier valor return es aceptable si se proporciona un tablero terminal / terminal board como entrada.
* La función result toma un board y una actions como entrada, y debe devolver un nuevo estado del tablero, sin modificar el tablero original.
  + Si actions no es una acción válida para el tablero, su programa debe generar una excepción. (<https://docs.python.org/3/tutorial/errors.html#raising-exceptions> )
  + El estado del tablero devuelto debe ser el tablero que resultaría de tomar el tablero de entrada original y dejar que el jugador a quien le toca hacer su movimiento en la celda indicada por la acción de entrada.
  + Es importante destacar que el tablero original debe dejarse sin modificar: dado que **Minimax** finalmente requerirá considerar muchos estados de tablero diferentes durante su cálculo. Esto significa que simplemente actualizar una celda en el tablero en sí no es una implementación correcta de la función result. Probablemente querrá hacer una **deepcopy** del tablero primero antes de realizar cualquier cambio. (<https://docs.python.org/3/library/copy.html#copy.deepcopy> )
* La función winner debe aceptar un board como entrada y devolver el ganador del tablero si lo hay.
  + Si el jugador X ganó el juego, su función debe devolver X. Si el jugador O ganó el juego, su función debe devolver O.
  + Se puede ganar el juego con tres de sus movimientos seguidos horizontal, vertical o diagonalmente.
  + Puede suponer que habrá como máximo un ganador (es decir, ningún tablero tendrá nunca a ambos jugadores con tres en raya, ya que ese sería un estado de tablero no válido).
  + Si no hay un ganador del juego (ya sea porque el juego está en curso o porque terminó en un empate), la función debe devolver **None**.
* La función terminal debe aceptar un board como entrada y devolver un valor booleano que indique si el juego terminó.
  + Si el juego terminó, ya sea porque alguien ganó el juego o porque se llenaron todas las celdas sin que nadie ganara, la función debe devolver ***True***.
  + De lo contrario, la función debe devolver ***False*** si el juego aún está en curso.
* La función de utility debe aceptar un terminal board /tablero terminal como entrada y generar la utilidad del tablero.
  + Si X ganó el juego, la utilidad es 1. Si O ganó el juego, la utilidad es -1. Si el juego terminó en un empate, la utilidad es 0.
  + Puede suponer que utility solo se llamará en un board si terminal(board) es ***True***.
* La función **minimax** debe tomar un board como entrada y devolver el movimiento óptimo que el jugador debe realizar en ese tablero.
  + El movimiento devuelto debe ser la acción óptima **(i, j)** que sea una de las acciones permitidas en el tablero. Si hay varios movimientos igualmente óptimos, cualquiera de ellos es aceptable.
  + Si el board es un terminal board, la función **minimax** debe devolver **None**.

Para todas las funciones que aceptan un board como entrada, puede asumir que es un board válido (es decir, que es una lista que contiene tres filas, cada una con tres valores de X, O o EMPTY). No debe modificar las declaraciones de función (el orden o la cantidad de argumentos para cada función) proporcionadas.

Una vez que todas las funciones se implementen correctamente, debería poder ejecutar python runner.py y jugar contra su IA. Y, dado que el tres en raya es un empate si ambos bandos juegan de forma óptima, nunca deberías poder vencer a la IA (aunque si no juegas de forma óptima, ¡puede que te gane!).

Sugerencias

* Si quieres probar tus funciones en un archivo Python diferente, puedes importarlas con líneas como **from** tictactoe **import** initial\_state.

NOTAS

A screenshot of a computer screen

AI-generated content may be incorrect.