Player()

* Takes a state as input
* Counts the number of x’s and o’s on the board
* If the number of x’s = the number of o’s, it’s x’s turn
* If the number of x’s is one more than the number of o’s, it’s o’s turn
* If the difference between the two, raise an exception

Actions()

* Takes a state as input
* Create an empty set that will contain all possible moves
* Loop through the 3 lists in the state one by one and check for empty spaces
  + For every empty space, return the index and combine it with the index of the list to create a set of coordinates
  + Format the coordinates as a tuple and append it to the set of all moves
* Return the set

Result()

* Takes a board state and an action
* Needs to know whose turn it is
* Using the indices given by the action, identify the item in the list that needs to be changed
* If the item is currently empty, change it to either x or o, depending on whose turn it is
* If the item is not empty, raise an exception

Terminal()

* Takes a state as input
* If none of the items in state is empty, return True, meaning the game is over
* If either x or o has gotten 3 in a row, return True, meaning the game is over
* If neither of these is true, return False, meaning the game is not over

Utility()

* Takes a state as input
* Set a variable called value
* If there are 3 x’s in a row, the value is 1
* If there are 3 o’s in a row, the value is -1
* If neither of these situations is true, the value is 0

Minimax

* Takes a state as input
* Calls the actions function to see all the possible actions
* Question: Which of the options is the best one?
  + The actions() function is called to list all the possible actions
  + The max player picks an action that leads to the highest possible value:
    - The maximum value of the minimum values of Result(s,a)
  + The min player picks an action that leads to the lowest possible value:
    - The minimum value of the maximum values of Result(s,a)
* max\_value(state):
  + If Terminal(s):
    - Result = utility(state)
    - Return result
  + else:
    - v = - infinity🡪 The goal is to get the highest value possible, so if the initial value is – infinity, any option is higher
    - Create a variable called actions and assign the set generated by actions() to it
    - For action in actions(state):
      * V = max(v, min\_value (result(state, action)))
      * Choose between v (is set as – infinity initially) and the minimum
      * If the result is not a terminal state, it is 0 for the time being
    - Return v
* min\_value(state):
  + if terminal(s):
    - result = utility(state)
    - return result
  + else:
    - v = infinity
    - For action in actions(state):
      * V = min(v, max\_value(result(state, action)))
    - Return v