Random Placement of Players

After checking the arguments for correct data, I create two lists that contain the number of empty spaces for each side of the field. After that I shuffle both sides to be random, and return a subset of the list depending on num\_players or spaces available. This was the most straight forward way I could think of meeting the requirements.  
  
I implemented all the tasks into smaller subroutines: sanitizing data, creating list of empty spaces, shuffling, and swapping elements. I chose to create two empty lists of left and right side when checking for available spaces instead of dividing later and decreasing efficiency. I also used lists to manipulate data (shuffling and adding) for ease of use. Because I convert once two list to a tuple once (O(2n)) it does not impact my time complexity.

The function to create\_empty\_lists has nested loops iterating over the field, which results in a time complexity of O(n^2), n being the shape of the field. All other functions or manipulations were of constant time. This implementation is O(n^2) as requested in the instructions.