```python
#Functions Challenge 34:  Head to Head Tic Tac Toe App

def draw_board(char_list):
    """Print a game board; either a number board or a tic tac toe board."""
    print("\n\t   Tic-Tac-Toe")
    print("\t~~~~~~~~~~~~~~~~~")
    print("\t|| " + char_list[0] + " || " + char_list[1] + " || " + char_list[2] + " ||")
    print("\t~~~~~~~~~~~~~~~~~")
    print("\t|| " + char_list[3] + " || " + char_list[4] + " || " + char_list[5] + " ||")
    print("\t~~~~~~~~~~~~~~~~~")
    print("\t|| " + char_list[6] + " || " + char_list[7] + " || " + char_list[8] + " ||")
    print("\t~~~~~~~~~~~~~~~~~")


def get_player_input(player_char, char_list):
    """Get a players move until it is a valid move on the board with no piece currently there."""
    while True:
        #Get user input
        player_move = int(input(player_char + ": Where would you like to place your piece (1-9): "))
        #Move is on board
        if player_move > 0 and player_move < 10:
            #Move is an empty spot
            if char_list[player_move - 1] == '_':
                return player_move
            else:
                print("That spot has already been chosen.  Try again.")
        else:
            print("That is not a spot on the board.  Try again.")


def place_char_on_board(player_char, player_move, char_list):
    """Put a players character at the correct spot on the board."""
    char_list[player_move - 1] = player_char


def is_winner(pC, cL):
    """Return a Bool if the given player is a winner."""
    return ((cL[0] == pC and cL[1] == pC and cL[2] == pC) or #victory in first row
            (cL[3] == pC and cL[4] == pC and cL[5] == pC) or #victory in second row
            (cL[6] == pC and cL[7] == pC and cL[8] == pC) or #victory in last row
            (cL[0] == pC and cL[3] == pC and cL[6] == pC) or #victory in first column
            (cL[1] == pC and cL[4] == pC and cL[7] == pC) or #victory in second column
            (cL[2] == pC and cL[5] == pC and cL[8] == pC) or #victory in last column
            (cL[0] == pC and cL[4] == pC and cL[8] == pC) or #victory in diagonal 1
            (cL[2] == pC and cL[4] == pC and cL[6] == pC)) #victory in diagonal 2


#The main code
#Define variables
player_1 = 'X'
player_2 = 'O'
c_list = ['_']*9
n_list = ['1', '2', '3', '4', '5', '6', '7', '8', '9']

```

```python
#Draw the initial state of the game board
draw_board(n_list)
draw_board(c_list)

while True:
    #Player 1 turn
    #Get the players move
    move = get_player_input(player_1, c_list)
    #Put move on board
    place_char_on_board(player_1, move, c_list)
    #Re-draw game boards
    draw_board(n_list)
    draw_board(c_list)
    #Check to see if player 1 won
    if is_winner(player_1, c_list):
        print("Player 1 wins!")
        break
    #Check if there is a tie
    elif "_" not in c_list:
        print("The game was a tie!")
        break

    #Player 2 turn
    #Get the players move
    move = get_player_input(player_2, c_list)
    #Put move on board
    place_char_on_board(player_2, move, c_list)
    #Re-draw game boards
    draw_board(n_list)
    draw_board(c_list)
    #Check to see if player 1 won
    if is_winner(player_2, c_list):
        print("Player 2 wins!")
        break
```