* Use uniform to select location of start then use gamma to add direction left and right since will likely move within a closer distance
* Small board size, 7x7 or 5x5 with 4 ships. Could allow for player to select multiple.
* Allow user to input ships locations at start. During game, user can choose ship to move or to fire on a ship (mix of battleship and like strategy 🡪 Could increase this to also buying/placing a new ship or moving multiple spaces in a turn)
* Have one AI be MDP based with decision move one space (is it separate decisions one space up, down, left, right?) vs firing. (Future would include buying + placing).
* Have other AI be just straight decisions by uniform distribution for everything.
* If cell is taken, redo decision until valid move. Illegal moves get penalty to prevent lazy learners as taken from here https://towardsdatascience.com/an-artificial-intelligence-learns-to-play-battleship-ebd2cf9adb01
* If decide to move and it goes off the board, take modulo so it is still on the board
* Start with single cell ships, increase to multi-cell ships
* Have 2 simultaneous games at once, human vs. MDP and human vs. chance to see which performs better.
* Can get win percentages by having AI and MDP play against each other. Bootstrap/multi sims for this
* Display boards for all at first, switch to hidden when doing an actual gameplay example
* Game ends when ship are all gone

1. **Define game rules and board. Display these at start of program**
   1. **Set board size and ship number. Initial thought is 7x7 with 6 single square ships. Time permitting, this would be changed slightly to allow the user to select from a set number of board sizes and ship amounts.**
   2. **Default each users**
   3. **To start, there will be a user input board playing against (1) an MDP based AI board and (2) a Uniform distribution/random chance played board**
   4. **Establish turn order with current\_player variable**
2. **Begin game with inputs**
   1. **Set both AI games to random positions on the board. Will utilize pulling from a random uniform for the x and y coordinate. Will label each ship and maintain position in a Json-like array or a label on a dataframe position. If already a ship in that position, reroll for a free space.**
   2. **Allow user to input (x,y) coordinate for each ship. Will label each ship and maintain position in a Json-like array or a label on a dataframe position. If already a ship in that position, ask for new input at a free space.**
   3. **Show starting board. For non-playing and research tasks, will show all 3 boards.**
3. **Set inputs for MDPtoolbox function from paper.** 
   1. **Set rewards for each decision as 0 for choosing to move a space, -1 for moving to a space already with a ship, 1 for hitting a ship. This will change to a 1 for hitting and 2 for sinking if time permits a version with ships of multiple lengths. Also considering adding -1 for choosing off the board.**
   2. **Set decisions as either move 1 space or fire on a space. Could allow for a do nothing option, but this will likely be useless.** 
      1. **May need to set the move space as separate decisions but only allowing for one (one left, right, up, or down).**
4. **Begin game play loop**
   1. **Set loop to break once player 1 or player 2 have ship count of 0**
   2. **Set player 1 (the human player) to play first if human in game. Else, split player 2 below into player 1 and player 2 with same code adjusted. Will have a flag for human or AI.**
      1. **Ask for input decision**
      2. **If decision is to move, ask for coordinate and ship to move. Check if more than 1 space from current location. Ask for new input if it is.** 
         1. **Check if already a ship on the space. If not, clear the old space and set new space to chosen. Otherwise, ask for new coordinate.**
      3. **If decision is to fire, ask for coordinate. Check against opponent board if it is a hit and display if it is a hit or miss. If hit, decrease player 2 ship counter by 1 and remove ship from board.**
      4. **May allow player one to both move and fire or fire twice if playing against both types of AI at once.**
      5. **End player turn and set current player to player 2**
   3. **Let player 2 (both the random choice and the MDP) go**
      1. **For random choice player use a uniform random number generator for 0 to 2 and a floor to decide whether to move (0) or fire (1/2)**
         1. **If move, use runif(0,6) and floor to decide the ship. Use runif(0,4) and floor to decide direction to move in (left, right, up, down) a single space with modulo 7 to prevent overshooting board**
         2. **If fire, use runif and floor to decide (x,y) to fire on. Display if hit or miss. If hit, decrease player 1 ship counter by 1 and remove ship from board.**
      2. **For MDP let it decide optimal decision to make (move or fire)**
         1. **See if way to let it decide location and ship for move, or location for fire. If not, use runif method**
   4. **Display all boards to get current state. TBD what this will look like.** 
      1. **In actual gameplay, only display players current board and their hit/miss counter for each space on the opponents board.**
   5. **Once a player reaches 0 ships, end game and declare other player as winner**
5. **Code to evaluate performance**
   1. **Run above code for multiple games to get performance idea**
   2. **Adjust above code to set player 1 or 2 to RNG based AI and the other to the MDP based AI**
      1. **Run games against each other and record winner**
      2. **Run enough simulations to get an average win count for the MDP based solution vs. the control rng**
      3. **Using the output of these simulations, run bootstrap against them to see the win distribution and if there is a significance to the MDP approach**
         1. **Visualize**