* scenarios:
  1. Initializing the board
     + - Initialized as a list of lists in the init method. Making it easier for me to create and visualize the step processes.
  2. Determining how to represent multiple pieces at a given location on the board
     + - When \_num\_pieces at \_start\_loc or \_end\_loc are greater than 5 during the move\_piece method. Will store pieces into the location strings and parse through them the way you parse through a string for the return values
  3. Determining how to make single move, multiple move and reserve move.
     + - Single/Multiple = In the \_move\_pieces method, the number of pieces to be moved is defined.
       - Reserve = Pull from the reserve string to place the number of reserve “pieces” on the given place on the board.
  4. Remembering captured and reserved pieces for each player.
     + - Will store them into the created empty strings at the beginning of the class “reserved” and “captured”. And add and remove as needed until a captured string has 6 or more elements. Then a winner is declared.
  5. Determining how to track which player's turn is it to play right now.
     + - Not fully sure yet, still trying to figure this out. Currently only have a rule of if a player tries to move the wrong piece it gives the error message
  6. Determining how to handle the scenario when a move results in more than 5 pieces at a location.
     + - Cut the string after the 5th element, creating two strings, and parse through the newly created string to add to the captured and reserve lists.

Your scenario descriptions should be high-level (not code-level) pseudocode for all the methods involved, how they're interacting, and what they're each doing.  **They should convey detailed actionable plans for the given scenarios.**

# Author: Khrystian Clark  
# Date: 11/24/2020  
# Description: A FocusGame program that has two players playing the Focus Game  
  
class FocusGame:  
 reserved\_red = ""  
 reserved\_green = ""  
 captured\_red = ""  
 captured\_green = ""  
  
 def \_\_init\_\_(self, \_player1, \_player2):  
 *"""Initialize the game board with a list of lists"""* self.\_gameboard = [['R', 'R', 'G', 'G', 'R', 'R'],  
 ['G', 'G', 'R', 'R', 'G', 'G'],  
 ['R', 'R', 'G', 'G', 'R', 'R'],  
 ['G', 'G', 'R', 'R', 'G', 'G'],  
 ['R', 'R', 'G', 'G', 'R', 'R'],  
 ['G', 'G', 'R', 'R', 'G', 'G']]  
 self.\_player1 = ("Player1", 'R')  
 self.\_player2 = ("Player2", 'G')  
 pass  
  
 def move\_piece(self, \_player, \_start\_loc, \_end\_loc, \_num\_pieces):  
 *"""It returns an error or proper message"""* # if a player is trying to make a move out of turn, return 'not your turn'  
 if \_player[1] != \_start\_loc[0]:  
 return "not your turn"  
 # if the player provides invalid locations (source or destination), return 'invalid location'  
 if \_start\_loc or \_end\_loc > (5, 5):  
 return "invalid location"  
 if \_start\_loc or \_end\_loc < (0, 0):  
 return "invalid location"  
 # if the player is trying to move invalid number of pieces, return 'invalid number of pieces'  
 if len(\_start\_loc) < \_num\_pieces:  
 return "invalid number of pieces"  
 else:  
 if len(captured) >= 6:  
 # Return the winner  
 # return "successfully moved" message  
 pass  
  
 def show\_pieces(self, \_board\_loc):  
 *"""takes a position on the board and returns a list showing the pieces that are present at that location with  
 the bottom-most pieces at the 0th index of the array and other pieces on it in the order."""* """Communicates with the updated board value at the init method to find what is at the input location"""  
 pass  
  
 def show\_reserve(self, \_player):  
 *"""takes the player name as the parameter and shows the count of pieces that are in reserve for the player.  
 If no pieces are in reserve, return 0."""* """Communicates with the reserve strings to show which elements are reserved"""  
 pass  
  
 def show\_captured(self, \_player):  
 *"""takes the player name as the parameter and shows the number of pieces captured by that player.  
 If no pieces have been captured, return 0."""* """Communicates with the captured strings to show which elements are captured"""  
 pass  
  
 def reserved\_move(self, \_player, \_board\_loc):  
 *"""Takes the player name and the location on the board as the parameters. It places the piece from the reserve  
 to the location. It should reduce the reserve pieces of that player by one and make appropriate adjustments to  
 pieces at the location. If there are no pieces in reserve, return 'no pieces in reserve'"""* """Communicates with reserve string, and init method to place pieces on the board"""  
 pass