Backtracking and MRV Heuristic and Forward Checking Trace

1. Command in terminal: ./homework2.py inputFileName.txt
2. The program starts off in homework2.py where it checks if the input file provided meets necessary requirements
3. Prompts user for which heuristic they would like to test. User input is 2 (for backtracking and mrv heuristic).
4. Goes to heuristic.py, and a function called `backtracking(inputArray, nmkArray, additional="None")`. Function takes in sudoku as an array (inputArray) and list of nmk values (nmkArray). Additional is equal to "FC" for forward checking.
5. Creates a list (`stackOfSuduko`) that will hold possible renditions of the sudoku puzzle
6. Goes into loop that will continue until this list is empty, or if the answer is found
7. Pops out the last value in the list and checks if it is the answer with `isAnswer(inputArray)`.
8. If answer, exits loop. If not, goes to function `mrv(inputArray, nmkArray, additional = "None")`
9. Mrv function looks for the location of the sudoku puzzle that has the least possible inputs. Variable called "RemainingValues" holds possible values for each location.
10. First checks each row. Gathers all numbers in a row, possible numbers that aren't in this list are then added to RemainingValues for each location.
11. Then checks each column. Anytime a number is found, it removes it from the corresponding RemainingValues column.
12. Then checks each box. Creates lists corresponding to box values present. If these values are in the corresponding RemainingValues box, it removes them.
13. Checks if constraint propagation needs to be done. Answer = no.
14. Goes through RemainingValues and finds location which has minimum values. Location cannot equal [-1], representing a number that has been already placed or [], where there are no possible remaining values.
15. Checks if forward checking needs to be done. Answer = yes.
16. Goes to `forwardChecking(remainingValues)`
17. Checks if any index in the array is equal to [], representing that there are no possible values at that location. Returns true or false, false representing that [] is present.
18. If false, mrv function returns [], which will not be added to stackOfSudoku
19. If true, goes to a function called `generateNewPuzzle(inputArray, location, values)`. This function takes the inputArray and produces new arrays with an altered value, from a list (values), at a location. Returns a list of new arrays.
20. Given that this returned list is not empty, appends it stackOfSudoku
21. Repeats steps 6 through 20 until answer is found or all possible routes have been explored.