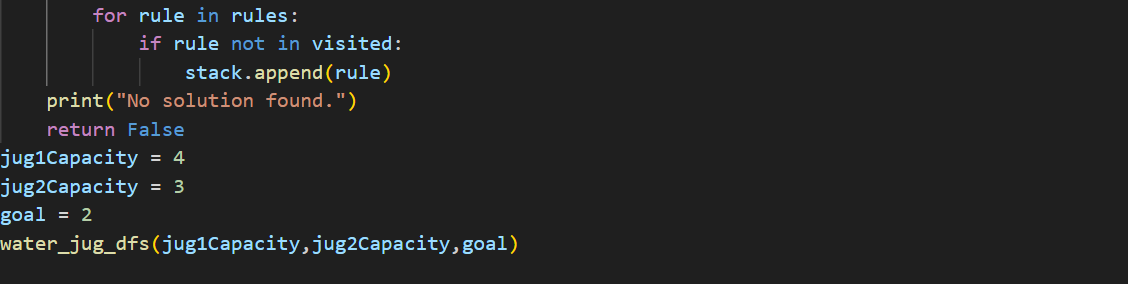
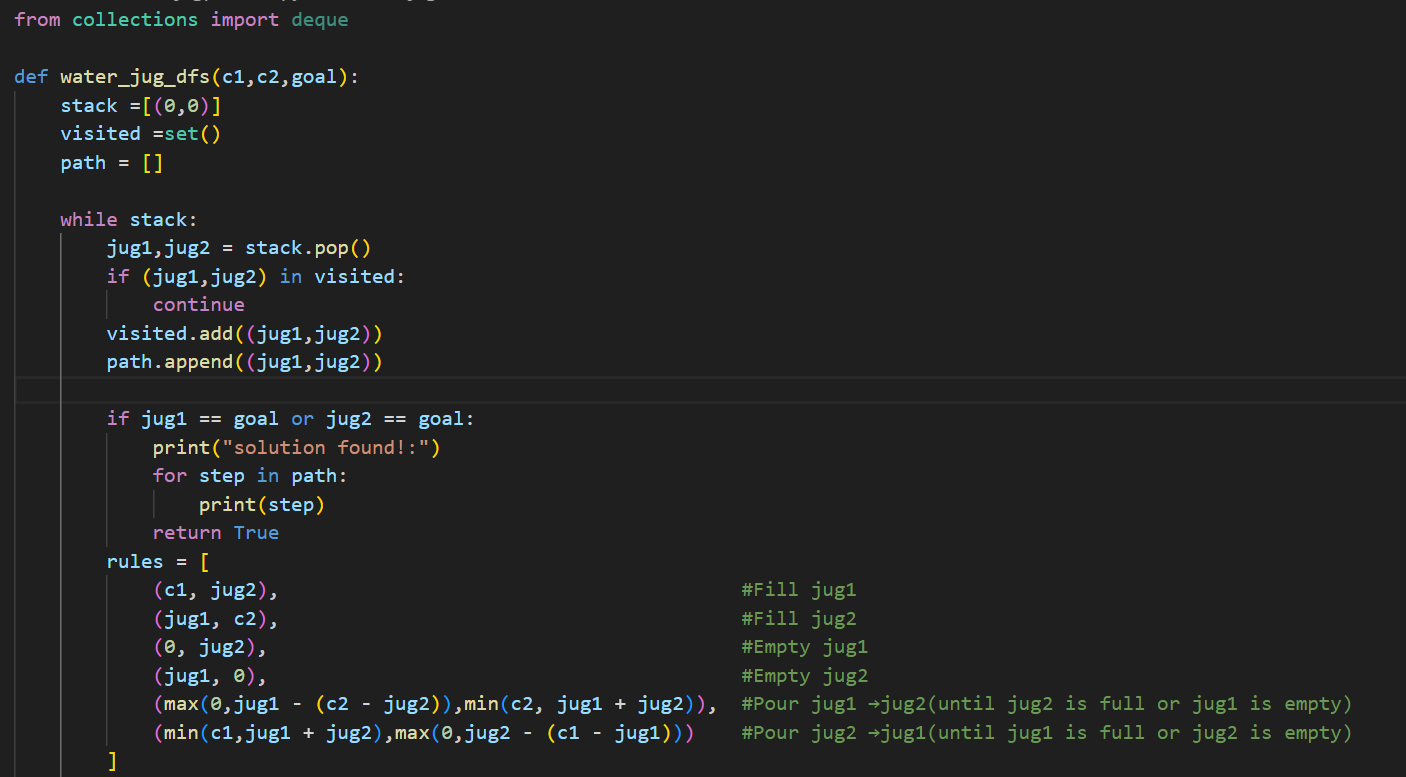
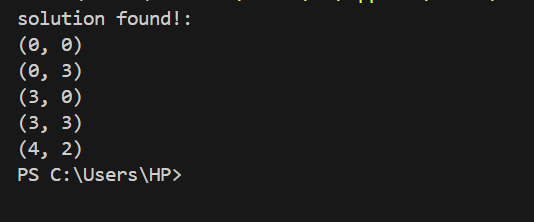
**Lab 3**

**Task: WaterJug with DFS & printing rules (also correct the rule 5 & 6)**



**OUTPUT:**



**How the Code Works**

The code starts by initializing a stack with the initial state (0, 0), representing both jugs being empty. It also initializes a visited set to keep track of explored states and a path list to store the sequence of states leading to the solution. The DFS algorithm proceeds by popping the current state from the stack, checking if it has been visited, and marking it as visited to avoid cycles. If the current state matches the goal (i.e., either jug contains the desired amount of water), the solution is printed, and the function terminates. If not, the code generates new states by applying a set of rules: filling either jug to its capacity, emptying either jug, or pouring water from one jug to the other until the source jug is empty or the destination jug is full. These new states are added to the stack if they haven't been visited yet. The process repeats until the stack is empty or a solution is found. If no solution exists, the function prints "No solution found."