Lab 3

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

Here's a step-by-step explanation of the code:

1. Function `dfs(x, y, target, X, Y, visited)`

- **Purpose**: This function implements the **Depth-First Search (DFS)** algorithm to solve the Water Jug Problem.
- **Parameters**:
  - `x`: Current amount of water in Jug A.
  - `y`: Current amount of water in Jug B.
  - `target`: The desired amount of water we want to measure in either of the jugs.
  - `X`: Capacity of Jug A.
  - `Y`: Capacity of Jug B.
  - `visited`: A set used to track visited states to avoid revisiting and prevent infinite loops.
- **Steps**:
  - **Check if the state is already visited**:
    - If `(x, y)` has been visited before, return `False` to avoid revisiting this state.
  - **Mark the current state as visited**:
    - Add the current state `(x, y)` to the `visited` set.
  - **Print the current state**:
    - Display the amount of water in both jugs (`Jug A: x liters, Jug B: y liters`).
  - **Check if the target is reached**:
    - If either Jug A or Jug B contains the target amount, print that the target has been reached and return `True`.
  - **Define possible actions**:
    - List of actions that can be performed on the jugs:
      1. **Fill Jug A** to its full capacity `X`.
      2. **Fill Jug B** to its full capacity `Y`.
      3. **Empty Jug A** completely.
      4. **Empty Jug B** completely.
      5. **Pour water from Jug A to Jug B** (until Jug B is full or Jug A is empty).
      6. **Pour water from Jug B to Jug A** (until Jug A is full or Jug B is empty).
  - **Explore all possible actions recursively**:
    - For each possible action, the function recursively explores the resulting state `(new_x, new_y).`
    - If the recursive call leads to a solution, return `True`.
  - **If no solution is found after all possible actions**, return `False`.

2. Function `water_jug_dfs(X, Y, target)`

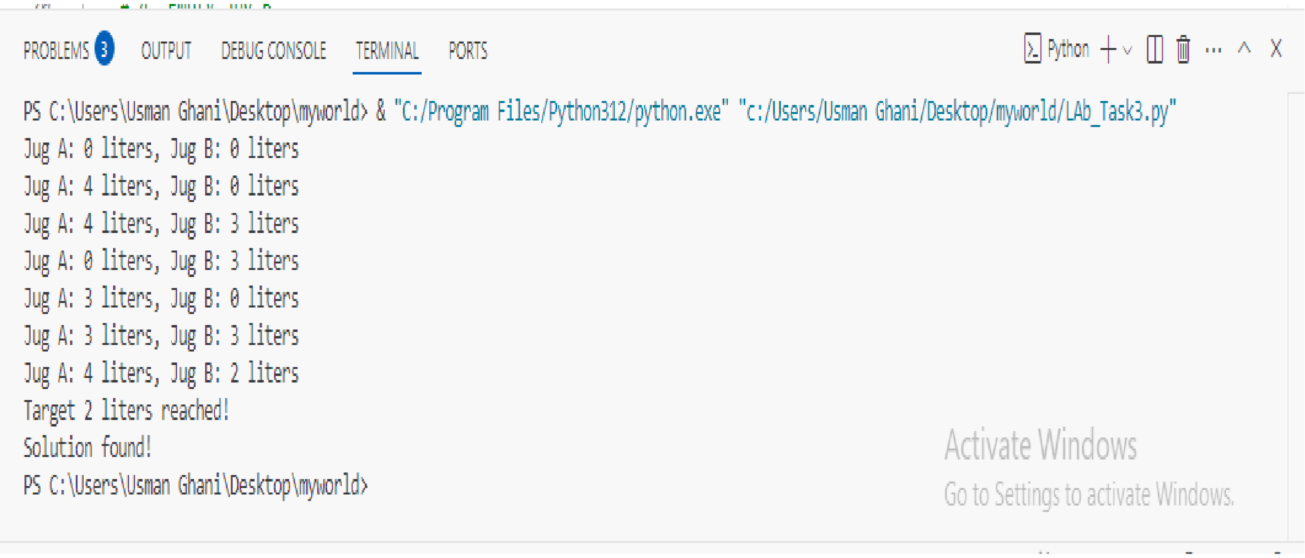
- **Purpose**: This function initializes the DFS search for the Water Jug Problem.
- **Parameters**:
  - `X`: Capacity of Jug A.
  - `Y`: Capacity of Jug B.
  - `target`: The target amount of water to measure.
- **Steps**:
  - **Initialize the `visited` set**: Used to keep track of previously visited states.
  - **Call `dfs` function**: Start the search with both jugs empty (initial state `(0, 0)`) by calling `dfs(0, 0, target, X, Y, visited)`.
  - **Output result**:
    - If `dfs` returns `True` (i.e., a solution is found), print "Solution found!".
    - If `dfs` returns `False`, print "No solution found!".

3. Example Execution (`water_jug_dfs(4, 3, 2)`)

- **Problem Setup**:
  - Jug A has a capacity of 4 liters.
  - Jug B has a capacity of 3 liters.
  - The target is to measure exactly 2 liters in either Jug A or Jug B.
- **Expected Behavior**:
  - The `water_jug_dfs(4, 3, 2)` function is called, and the DFS algorithm explores all possible actions to find a solution for measuring exactly 2 liters.
  - If the target is reached, the solution path will be printed, and "Solution found!" will be displayed.

4. Actions/Operations in DFS

- The function simulates the possible actions with the jugs and recursively explores the resulting states:
  - **Fill Jug A completely**: This action sets Jug A's water amount to `X` (full).
  - **Fill Jug B completely**: This action sets Jug B's water amount to `Y` (full).
  - **Empty Jug A**: This sets Jug A's water amount to 0.
  - **Empty Jug B**: This sets Jug B's water amount to 0.
  - **Pour from Jug A to Jug B**: This action moves as much water from Jug A to Jug B as possible, either until Jug B is full or Jug A is empty.
  - **Pour from Jug B to Jug A**: This action moves as much water from Jug B to Jug A as possible, either until Jug A is full or Jug B is empty.



```
PS C:\Users\Usman Ghani\Desktop\myworld> & "C:/Program Files/Python312/python.exe" "c:/Users/Usman Ghani/Desktop/myworld/LAb_Task3.py"
Jug A: 0 liters, Jug B: 0 liters
Jug A: 4 liters, Jug B: 0 liters
Jug A: 4 liters, Jug B: 3 liters
Jug A: 0 liters, Jug B: 3 liters
Jug A: 3 liters, Jug B: 0 liters
Jug A: 3 liters, Jug B: 3 liters
Jug A: 4 liters, Jug B: 2 liters
Target 2 liters reached!
Solution found!
PS C:\Users\Usman Ghani\Desktop\myworld>
```