## TOY PROBLEM: Water Jug Problem

**Problem Statement:**

You are given two jugs with
capacities x and y liters. There are no markings on the jugs, so you can't measure exact quantities directly. The goal is to measure exactly z liters of water using these two jugs.

**CODE:-**

```
from collections import deque


def water_jug_bfs(x, y, z):
    """
    Solve the Water Jug problem using BFS.

    :param x: Capacity of jug 1
    :param y: Capacity of jug 2
    :param z: Target amount of water to measure
    :return: Sequence of steps to achieve the target, or "Not Possible"
    """
```

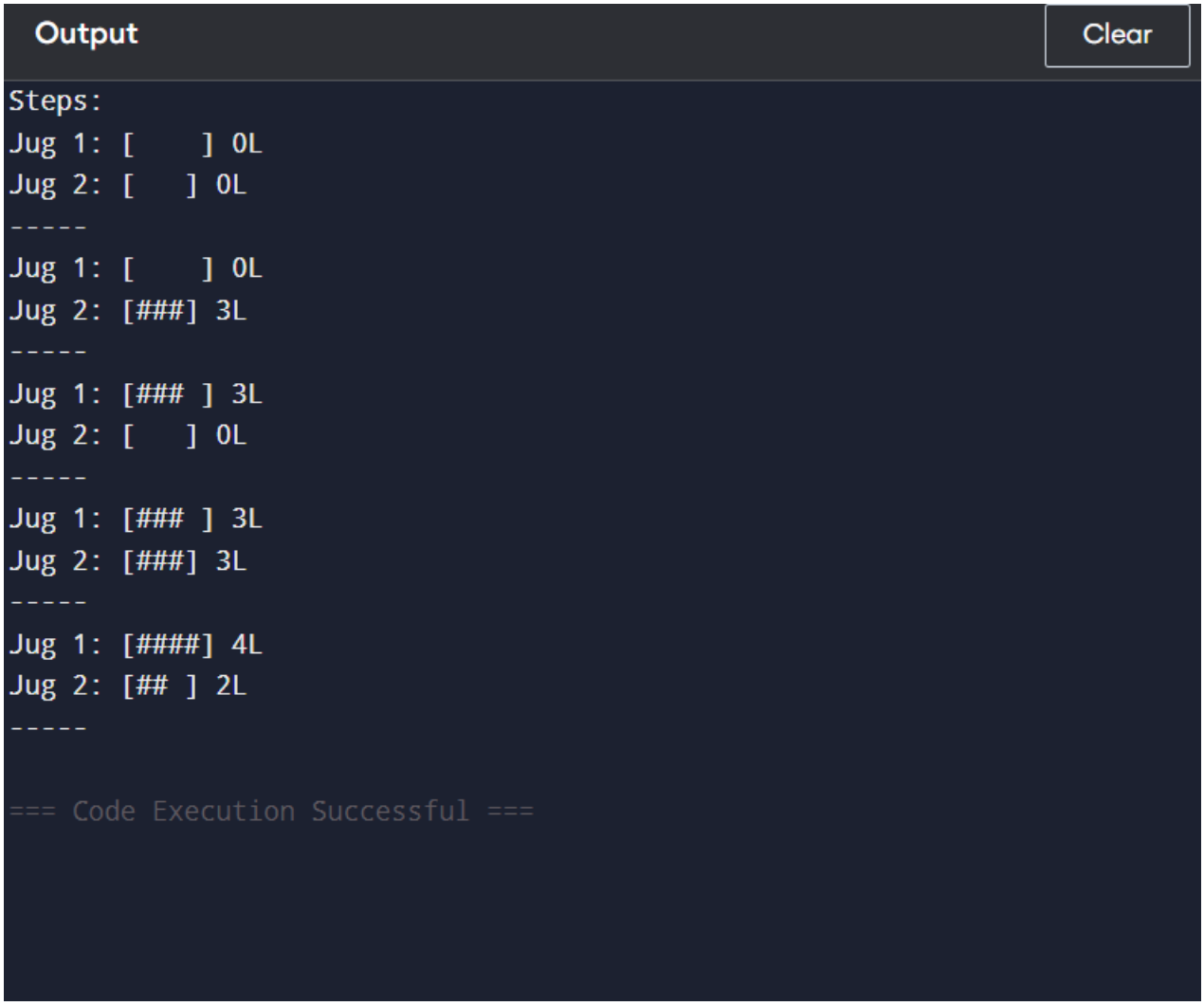
```
    queue = deque([(0, 0)])  # Start with both jugs empty
    visited = {(0, 0): None}  # Track visited states and their parents

    while queue:
        jug1, jug2 = queue.popleft()

        # If target is achieved, return the path
        if jug1 == z or jug2 == z:
            path = []
            current = (jug1, jug2)
            while current is not None:
                path.append(current)
                current = visited[current]
            return path[::-1]  # Reverse to get the correct order

        # Generate all possible states
```

```
        states = [
            (x, jug2),  # Fill jug1
            (jug1, y),  # Fill jug2
            (0, jug2),  # Empty jug1
            (jug1, 0),  # Empty jug2
            (jug1 - min(jug1, y - jug2), jug2 + min(jug1, y - jug2)),  # Pour jug1 to jug2
            (jug1 + min(jug2, x - jug1), jug2 - min(jug2, x - jug1)),  # Pour jug2 to jug1
        ]

        # Explore each state
        for state in states:
            if state not in visited:
                visited[state] = (jug1, jug2)
                queue.append(state)

    return "Not Possible"
```

```
def draw_jug(amount, capacity):
    """
    Draw a simple jug with water level using ASCII art.

    :param amount: Current amount of water in the jug
    :param capacity: Total capacity of the jug
    """
    print(f"[{'#' * amount}{' ' * (capacity - amount)}] {amount}L")

# Input (hardcoded for simplicity)
x, y, z = 4, 3, 2  # Capacities of jugs and target amount

# Solve
result = water_jug_bfs(x, y, z)

# Output
```

```
if result == "Not Possible":
    print("Not Possible")
else:
    print("Steps:")
    for step in result:
        print(f"Jug 1: ", end="")
        draw_jug(step[0], x)
        print(f"Jug 2: ", end="")
        draw_jug(step[1], y)
        print("-----")
```

**OUTPUT:-**




**Explaination:-**

This code solves the **Water Jug Problem** using **Breadth-First Search (BFS)**. You have two jugs with capacities x and y. The goal is to measure exactly z liters of water by filling, emptying, or pouring water between the jugs. The code finds the shortest sequence of steps to achieve this.

**How It Works**

1. **Start**: Both jugs are empty (0, 0).
2. **Explore**: The code tries all possible actions (fill, empty, pour) and keeps track of visited states to avoid loops.
3. **Goal**: If either jug contains exactly z liters, the solution is found.
4. **Output**: The sequence of steps is displayed, showing the state of both jugs at each step.

**Novelty:-**

1. **Compact Visualization:**

- The state of each jug is shown using simple ASCII art ([####] for water and [ ] for empty space).

2. **Minimal Input and Output Explanation**

**Input**: The capacities of the jugs and the target amount are hardcoded as x, y, z = 4, 3, 2, making it straightforward to test the solution.

**Output**: The output is concise and displays only the essential steps, showing the state of both jugs at each step.

3. **Efficient and Clear**:

- Uses BFS to ensure the shortest solution.
- Simple and easy to understand, even for beginners.