Codefest #3

The FrozenLake Problem

The "deterministic" FrozenLake is a toy problem from the so called "grid world" category of problems. In this problem the agent lives in a square grid and can move in 4 directions, "up", "down", "left" and "right". The agent always starts in the top-left position and its goal is to reach the bottom right position on the grid (see image below).

Part 1: Create the Frozen Lake

1. Using Python, create a 5x5 grid sized Frozen Lake, with a start state at the top left corner and a goal state at the bottom right corner.
2. Place four holes at the following grid positions in the Frozen Lake. **(2,0)**, **(4,1)**, **(2,2)**, **(3,3)**
3. The reward for reaching the goal state is **+10.0**. The reward for falling into a hole is **-5.0** (because you die!) and the rewards for each transition to a non-terminal state is **-1.0**.
4. The episode ends if the agent falls into a hole or reaches the goal state.
5. The actions are **"up"**, **"down"**, **"left"** and **"right"**.

	0	1	2	3	4
0	Start (0,0)				
1					
2	Hole (2,0)		Hole (2,2)		
3				Hole (3,3)	
4		Hole (4,1)			Goal (4,4)

```
import numpy as np


class FrozenLake:
  def __init__(self):
    self.size = 5
```

```
        self.start_pos = (0, 0)
        self.goal_pos = (4, 4)
        self.holes = {(2, 0), (4, 1), (2, 2), (3, 3)}
        self.actions = ['up', 'down', 'left', 'right']
        self.reset()

    def reset(self):
        self.agent_pos = self.start_pos
        return self.agent_pos

    def step(self, action):
        x, y = self.agent_pos

        if action == 'up':
            x = max(0, x - 1)
        elif action == 'down':
            x = min(self.size - 1, x + 1)
        elif action == 'left':
            y = max(0, y - 1)
        elif action == 'right':
            y = min(self.size - 1, y + 1)
        else:
            raise ValueError(f"Invalid action: {action}")

        self.agent_pos = (x, y)

        # Check if agent is in a hole
        if self.agent_pos in self.holes:
            reward = -5.0
            done = True
        # Check if agent reached the goal
        elif self.agent_pos == self.goal_pos:
            reward = +10.0
            done = True
        else:
            reward = -1.0
            done = False

        return self.agent_pos, reward, done

    def render(self):
        grid = [['.' for _ in range(self.size)] for _ in range(self.size)]

        for hx, hy in self.holes:
            grid[hx][hy] = 'H'

        gx, gy = self.goal_pos
        grid[gx][gy] = 'G'

        ax, ay = self.agent_pos
        grid[ax][ay] = 'A'

        for row in grid:
            print(' '.join(row))
        print()

# Example run
env = FrozenLake()
env.render()

state, reward, done = env.step('right')
env.render()
print(f"State: {state}, Reward: {reward}, Done: {done}")
```

Key Features: • 5×5 grid

- Start: `(0, 0)`
- Goal: `(4, 4)` with `+10.0`
- Holes at `(2,0), (4,1), (2,2), (3,3)` with `-5.0`
- Movement: `up`, `down`, `left`, `right.`
- Transition reward: `-1.0`
- Episode ends on hole or goal.
- `render()` method to print grid visually.