DOCUMENTATION VACUUM CLEANER SIMULATION

OVERVIEW

This program (*scripts/vacuum cleaner3.py*) simulates a table-driven reflex agent (vacuum cleaner) operating in a two-room environment "A" and "B" and it contains an additional background character (evilMan) continuously dirtying the environment, creating an adversarial setting.

1. Purpose

- Demonstrates a **basic AI agent** using a lookup table for decision-making.
- Shows how concurrency (threading) can simulate environmental changes.

2. Environment Model

- **Rooms**: 'A' and 'B'
- **States**: "Clean" or "dirty"
- **Global environment** stored in dictionary ***env***

```
env = {
'A': [],
'B': []
 }
```

3.Agent Table

```
agent_table = {
  ('Clean', 'A'): 'MoveRight',
  ('Clean', 'B'): 'MoveLeft',
  ('Dirty', 'A'): 'Suck',
  ('Dirty', 'B'): 'Suck',
   }
```

- If dirty → **Suck**
- If clean → **Move to other room**

4. VacuumCleaner Class

It represents the agent in this case.

Attributes

- location: Current room ('A' or 'B')

- status is Perceived cleanliness ('Clean' or 'Dirty')

Methods

- percept(): Returns current status.
- act(action): Executes chosen action.

5. EvilMan Thread

Defined `def evilMan(interval, vacuum)`

Its a background thread that dirties rooms every few seconds.

- Runs 5 times, sleeping interval seconds each.
- Almost always sets vacuum.status = "Dirty".
- Overwrites room states inconsistently.

6. Main Simulation Loop

Runs agent for 10 steps:

1. Get percept (vacuum.percept())
2. Lookup action (table_driven_agent())
3. Execute action (vacuum.act())
4. Print state