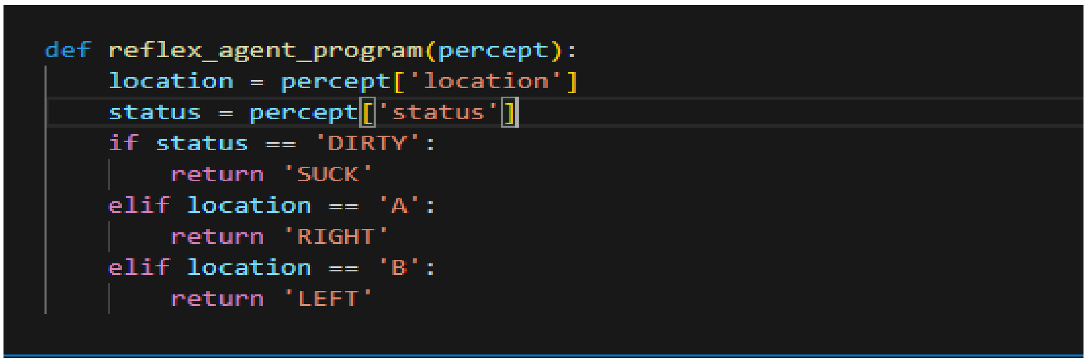
Lab #4: Intelligent Agents

The main aim of the lab is to implement a reflex agent.

Simple Reflex Agents: These agents select actions on the basis of the current percept, often using condition-action rules (if-then rules). They operate by looking at the world state as it is right now and choosing an action that seems appropriate to the current situation. For example, a simple reflex agent in a vacuum cleaner robot might have a rule that says, "if the current location is dirty, then clean.

Task 1: Implement an agent program for **reflex agent** working in the 2 squares [A,B] environment:



```python
def reflex_agent_program(percept):
    location = percept['location']
    status = percept['status']
    if status == 'DIRTY':
        return 'SUCK'
    elif location == 'A':
        return 'RIGHT'
    elif location == 'B':
        return 'LEFT'
```

Then, implement the following methods in Enviroment: Environment has an EnvironmentState object to track the states of locations in the environment.

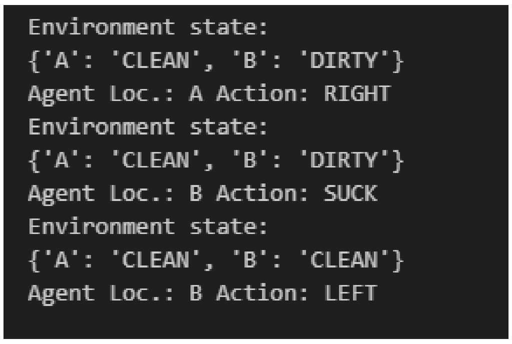
```
class Environment:
    def __init__(self, location_a_state, location_b_state):
        self.state = {'A': location_a_state, 'B': location_b_state}

def execute_action(self, agent, action):
if action == 'LEFT':
// location of agent
elif action == 'RIGHT':
// location of agent
 def step(self, n):
```

Test/Main:

```
envir = Environment('CLEAN', 'DIRTY')
agent = Agent(reflex_agent_program)
agent.location = 'A'
envir.step(3)
```

The output is as follows:

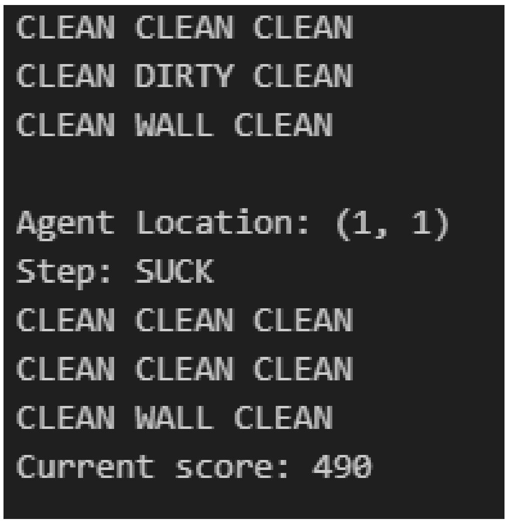


```
Environment state:
{'A': 'CLEAN', 'B': 'DIRTY'}
Agent Loc.: A Action: RIGHT
Environment state:
{'A': 'CLEAN', 'B': 'DIRTY'}
Agent Loc.: B Action: SUCK
Environment state:
{'A': 'CLEAN', 'B': 'CLEAN'}
Agent Loc.: B Action: LEFT
```

Task 2: Expand the reflex agent according to the following requirements:

- Environment is an **m × n grid** (the room is divided into a discrete number of cells)
- There exist a number of dirt and obstacles in the environment. Dirt and obstacles (walls) are **randomly placed** in the cells with a given rate. Therefore, the number of obstacles will be m*n*WALL_RATE (suppose `DIRT_RATE = 0.2; WALL_RATE = 0.1;`)
- At each step:
  - if cell is **DIRTY**, then action **SUCK** is invoked
  - if cell is **CLEAN**, then pick a random **direction to move** (UP, DOWN, LEFT, RIGHT), and pick move action (if can't move there (i.e, because of obstacle), then will remain in same cell).
  - For example, if the direction is UP, then the move action will move up 1 square.
- Performance measure (score):
  - For action **SUCK**, + 500 points;
  - If agent can't move (because of obstacle) - 100 points;
  - For each action taken: - 10 points;
- Develop the GUI for vacuum agent in the grid environment as suggested (optional):

Output



```
CLEAN CLEAN CLEAN
CLEAN DIRTY CLEAN
CLEAN WALL CLEAN

Agent Location: (1, 1)
Step: SUCK
CLEAN CLEAN CLEAN
CLEAN CLEAN CLEAN
CLEAN WALL CLEAN
Current score: 490
```

```
m, n = 3, 3
dirt_rate = 0.2
wall_rate = 0.1
envir = Environment(m, n, dirt_rate, wall_rate)
envir.print_grid()
print("Agent Location:", envir.agent_location)
    print("Step:", action)
    envir.execute_action(action)
    envir.print_grid()
    envir.display_performance()
```