Ayesha Zamurd
49733
BSCS-06

Artificial Intelligence
Lab: 09

<u>Lab Tasks</u>

Question 01:

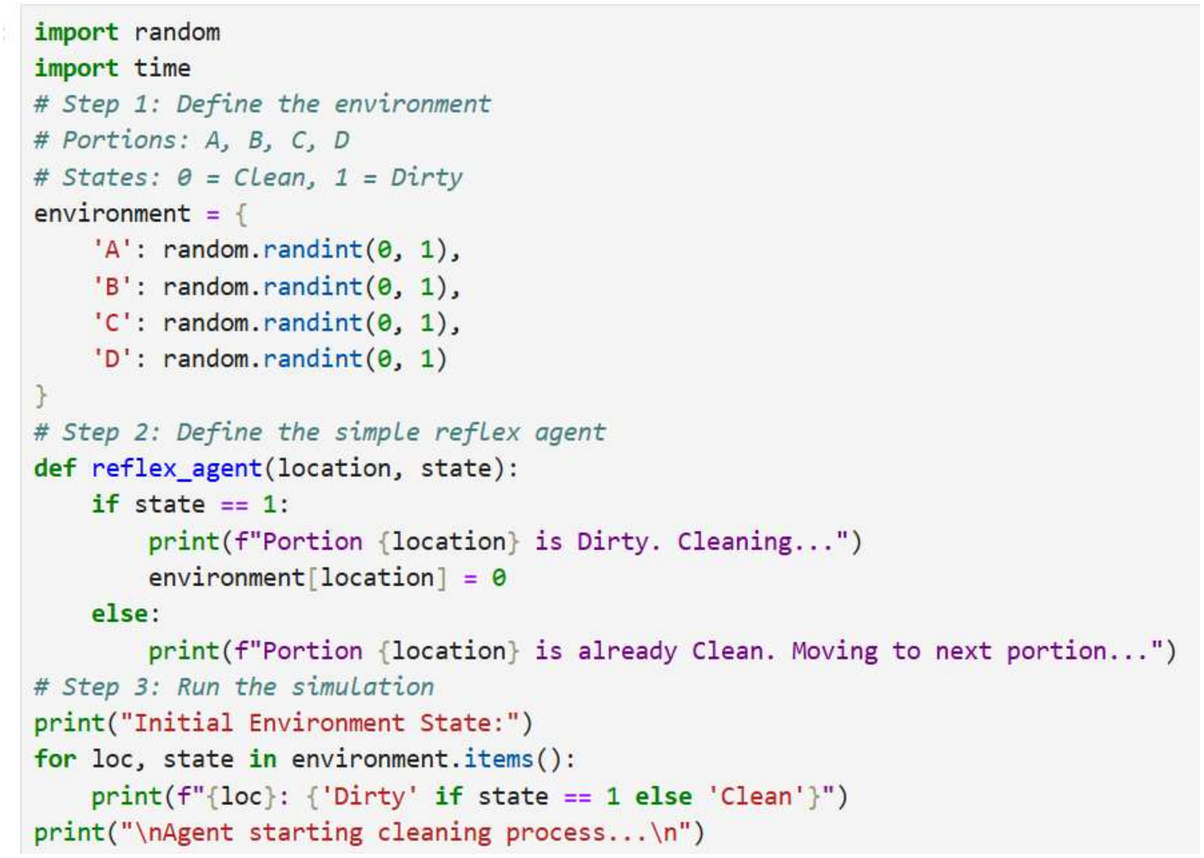
Write a program for a simple reflex agent. The agent will act as a vacuum cleaner. In the first activity, wewill create an environment for the agent.

• The environment is divided into 4 portions A,B,C and D.

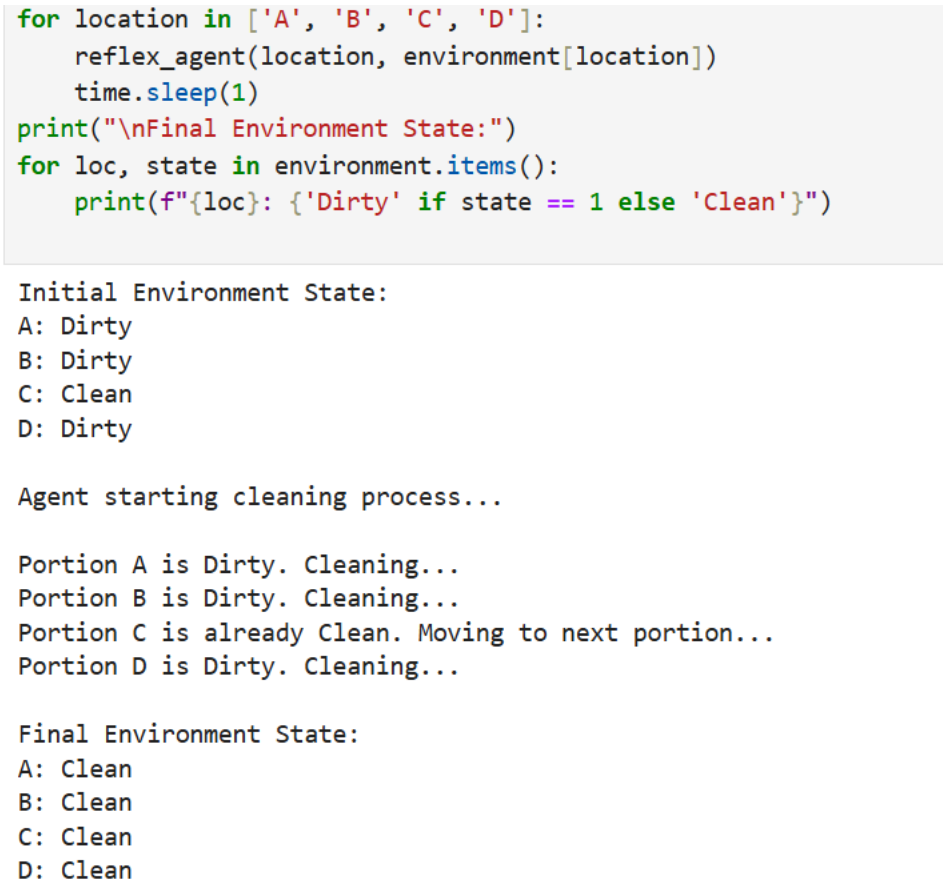
• Then define two states for each portion.

• 0 indicates the cleaned state and 1 indicates the dirty state.

• We will initialize each portion with a random state that would be either 0 or 1.



```python
import random
import time
# Step 1: Define the environment
# Portions: A, B, C, D
# States: 0 = Clean, 1 = Dirty
environment = {
    'A': random.randint(0, 1),
    'B': random.randint(0, 1),
    'C': random.randint(0, 1),
    'D': random.randint(0, 1)
}
# Step 2: Define the simple reflex agent
def reflex_agent(location, state):
    if state == 1:
        print(f"Portion {location} is Dirty. Cleaning...")
        environment[location] = 0
    else:
        print(f"Portion {location} is already Clean. Moving to next portion...")
# Step 3: Run the simulation
print("Initial Environment State:")
for loc, state in environment.items():
    print(f"{loc}: {'Dirty' if state == 1 else 'Clean'}")
print("\nAgent starting cleaning process...\n")
```



```python
for location in ['A', 'B', 'C', 'D']:
    reflex_agent(location, environment[location])
    time.sleep(1)
print("\nFinal Environment State:")
for loc, state in environment.items():
    print(f"{loc}: {'Dirty' if state == 1 else 'Clean'}")
```

```
Initial Environment State:
A: Dirty
B: Dirty
C: Clean
D: Dirty

Agent starting cleaning process...

Portion A is Dirty. Cleaning...
Portion B is Dirty. Cleaning...
Portion C is already Clean. Moving to next portion...
Portion D is Dirty. Cleaning...

Final Environment State:
A: Clean
B: Clean
C: Clean
D: Clean
```

Question 02:

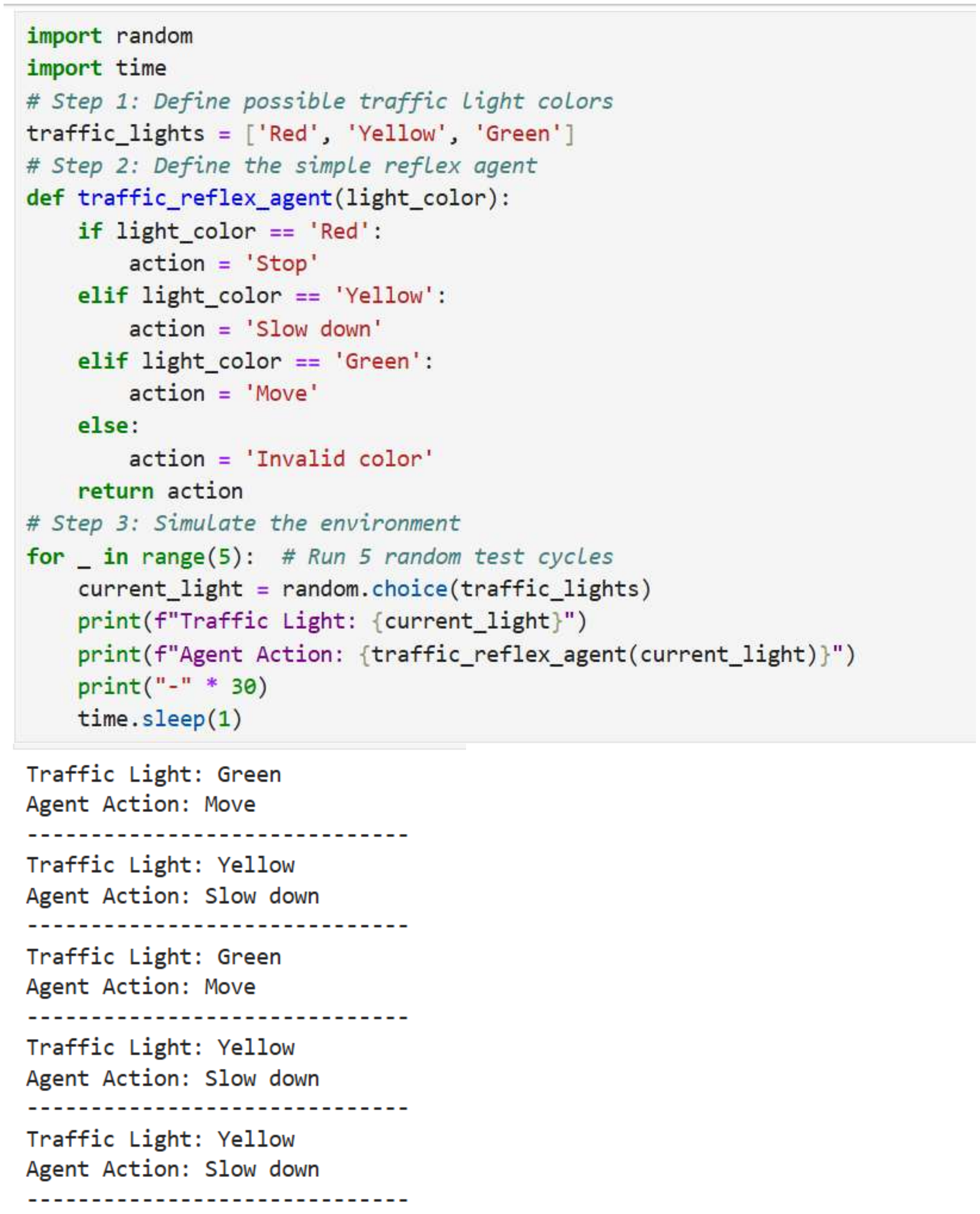
Create a Simple Reflex Agent that:

- Observes traffic light color (red, yellow, green).
- Takes an action based on the light:

  Red → Stop

  Yellow → Slow down

  Green → Move



```python
import random
import time
# Step 1: Define possible traffic light colors
traffic_lights = ['Red', 'Yellow', 'Green']
# Step 2: Define the simple reflex agent
def traffic_reflex_agent(light_color):
    if light_color == 'Red':
        action = 'Stop'
    elif light_color == 'Yellow':
        action = 'Slow down'
    elif light_color == 'Green':
        action = 'Move'
    else:
        action = 'Invalid color'
    return action
# Step 3: Simulate the environment
for _ in range(5):  # Run 5 random test cycles
    current_light = random.choice(traffic_lights)
    print(f"Traffic Light: {current_light}")
    print(f"Agent Action: {traffic_reflex_agent(current_light)}")
    print("-" * 30)
    time.sleep(1)
```

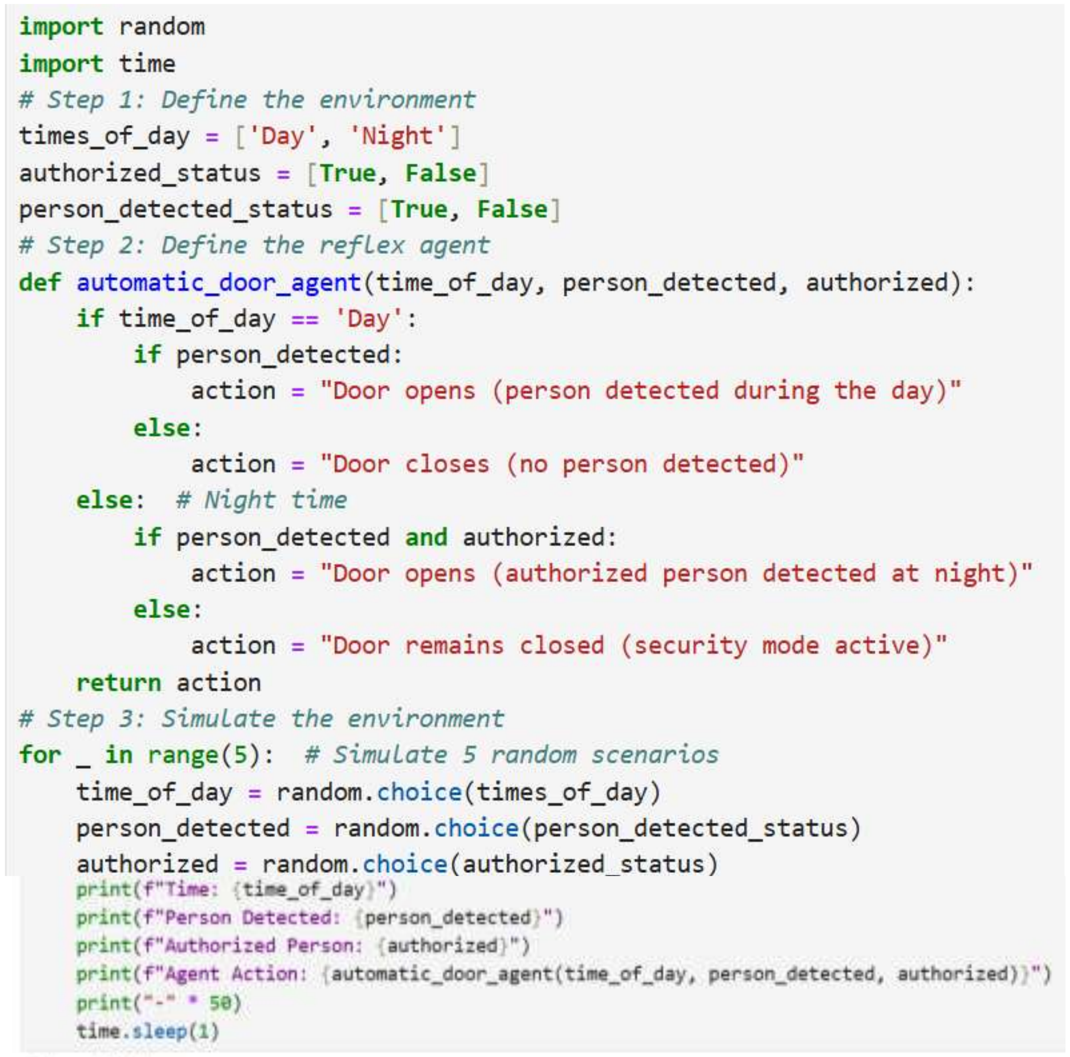
```
Traffic Light: Green
Agent Action: Move
------------------------------
Traffic Light: Yellow
Agent Action: Slow down
------------------------------
Traffic Light: Green
Agent Action: Move
------------------------------
Traffic Light: Yellow
Agent Action: Slow down
------------------------------
Traffic Light: Yellow
Agent Action: Slow down
------------------------------
```

Question 03:

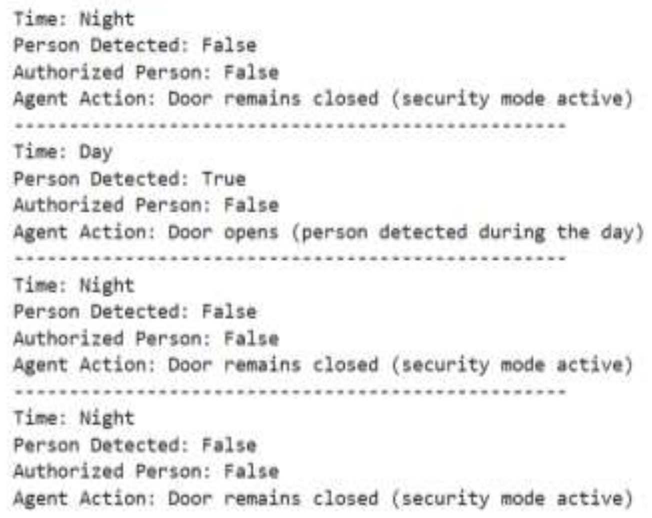
Implement an automatic door agent that:

- Opens if it detects a person near the door.
- Closes if no person is detected.

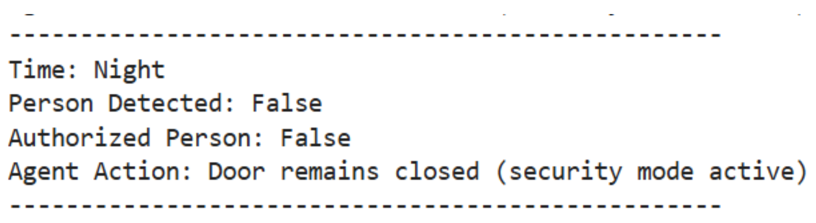
Add a security feature where the door stays closed at night unless an authorized person is detected.



```python
import random
import time
# Step 1: Define the environment
times_of_day = ['Day', 'Night']
authorized_status = [True, False]
person_detected_status = [True, False]
# Step 2: Define the reflex agent
def automatic_door_agent(time_of_day, person_detected, authorized):
    if time_of_day == 'Day':
        if person_detected:
            action = "Door opens (person detected during the day)"
        else:
            action = "Door closes (no person detected)"
    else:  # Night time
        if person_detected and authorized:
            action = "Door opens (authorized person detected at night)"
        else:
            action = "Door remains closed (security mode active)"
    return action
# Step 3: Simulate the environment
for _ in range(5):  # Simulate 5 random scenarios
    time_of_day = random.choice(times_of_day)
    person_detected = random.choice(person_detected_status)
    authorized = random.choice(authorized_status)
    print(f"Time: {time_of_day}")
    print(f"Person Detected: {person_detected}")
    print(f"Authorized Person: {authorized}")
    print(f"Agent Action: {automatic_door_agent(time_of_day, person_detected, authorized)}")
    print("-" * 50)
    time.sleep(1)
```



```
Time: Night
Person Detected: False
Authorized Person: False
Agent Action: Door remains closed (security mode active)
--------------------------------------------------
Time: Day
Person Detected: True
Authorized Person: False
Agent Action: Door opens (person detected during the day)
--------------------------------------------------
Time: Night
Person Detected: False
Authorized Person: False
Agent Action: Door remains closed (security mode active)
--------------------------------------------------
Time: Night
Person Detected: False
Authorized Person: False
Agent Action: Door remains closed (security mode active)
```



```
--------------------------------------------------
Time: Night
Person Detected: False
Authorized Person: False
Agent Action: Door remains closed (security mode active)
--------------------------------------------------
```

The end…