MuhammadHassanShah_20P-0025_C_Lab02

February 7, 2023

[10]:
```
#INSTRUCTIONS
#Enter LOCATION A/B in captial letters
#Enter Status 0/1 accordingly where 0 means CLEAN and 1 means DIRTY

def vacuum_world():
    # Initialize goal state with both rooms as clean

#     goal_state = {'A': '0', 'B': '0'}
    cost = 0

    # Get user input for vacuum location and room status
    location = input("Enter location of vacuum (A/B): ").upper()
    status = input(f"Enter status of room {location} (0/1): ")
    other_room = 'A' if location == 'B' else 'B'
    other_status = input(f"Enter status of room {other_room} (0/1): ")

    goal_state = {location: status, other_room: other_status}

    print("location condition:", goal_state)

    if status == '1':
        # Vacuum is in a dirty room
        goal_state[location] = '0'
        cost += 1
        print(f"Location {location} cleaned. Cost: {cost}")


        if other_status == '1':
            # Other room is dirty
            print(f"Moving to room {other_room}")
            cost += 1
            goal_state[other_room] = '0'
            cost += 1
            print(f"Location {other_room} cleaned. Cost: {cost}")
        else:
            print(f"Room {other_room} is already clean")
    else:
```

```
        print(f"Location {location} is already clean")

        if other_status == '1':
            # Other room is dirty
            print(f"Moving to room {other_room}")
            cost += 1
            goal_state[other_room] = '0'
            cost += 1
            print(f"Location {other_room} cleaned. Cost: {cost}")
        else:
            print(f"Room {other_room} is already clean")

    print("\nGoal state:", goal_state)
    print("Performance measurement:", cost)

vacuum_world()
```

```
Enter location of vacuum (A/B):  b
Enter status of room B (0/1):  1
Enter status of room A (0/1):  1

location condition: {'B': '1', 'A': '1'}
Location B cleaned. Cost: 1
Moving to room A
Location A cleaned. Cost: 3

Goal state: {'B': '0', 'A': '0'}
Performance measurement: 3
```

1 Task 1

[18]:
```
def water_system(status,water_status):
    if status:
        if water_status == False:                          # If this is printed␣
 ↪once it won't print everytime
            print("Water sprinkling")
        return True
    else:
        return False

def call_fire_dept(status,call_status):
    if status:
        if call_status == False:                          # If this is printed once␣
 ↪it won't print everytime
            print("Called Fire Department")
        return True
    else:
```

```
        return False

def smoke_status(smoke):
    if(smoke == "Y"):
        return True
    else:
        return False

def temp_check(temp):
    if(temp >= 45):
        return True
    else:
        return False

def fire_alarm():

    water_status = False                                                    # These are␣
 ↪the states that changes

    call_status = False

    while(1):

        smoke = input("Is there any smoke? (Y/N): ").upper()

        temp = int(input("What is temprature: "))

        water_status = water_system(smoke_status(smoke),water_status)

        call_status = call_fire_dept(temp_check(temp),call_status)

        if water_status == False and call_status == False:
            return

fire_alarm()
```

```
Is there any smoke? (Y/N):  y
What is temprature:  48

Water sprinkling
Called Fire Department

Is there any smoke? (Y/N):  n
What is temprature:  48
Is there any smoke? (Y/N):  n
What is temprature:  44
```

2 Task 2

[5]:
```
import logging

class WateringSystem:
    def __init__(self, dry_limit, moist_limit):
        self.dry_limit = dry_limit
        self.moist_limit = moist_limit
        self.is_on = False
        self.active = True

    def turn_on(self):
        if self.active:
            self.is_on = True
            logging.info("Watering system turned on.")
        else:
            logging.info("Watering system is not active.")

    def turn_off(self):
        if self.active:
            self.is_on = False
            logging.info("Watering system turned off.")
        else:
            logging.info("Watering system is not active.")

    def deactivate(self):
        self.active = False

    def activate(self):
        self.active = True

    def control_water(self,moisture_level):
        self.activate()
        if moisture_level <= dry_limit and self.active:
            self.turn_on()
        elif moisture_level <= moist_limit and self.active:
            self.turn_off()
        else:
            logging.info("Watering system Deactivated.")
            self.deactivate()


logging.basicConfig(filename="watering_system.log", level=logging.INFO)
try:
    dry_limit = float(input("Enter limit of water for dry"))
    moist_limit = float(input("Enter limit of water for moist"))
    moist_level = float(input("Enter Moisture Level"))
```

```
    w_system = WateringSystem(dry_limit,moist_limit)
    w_system.control_water(moist_level)
except:
    logging.exception("Invalid Inputs.")
```

```
Enter limit of water for dry 20
Enter limit of water for moist 30
Enter Moisture Level 25
```

[ ]: