**LAB 04**

**AGENTS AND ENVIRONMENTS**

**1. Run the two room vacuum cleaner agent program and understand it. Convert the program to a Three room environment.**

**Two Room Environment:**

from abc import abstractmethod

# Environment Class

class Environment(object):

    @abstractmethod

    def \_\_init\_\_(self, n):

        self.n = n

    def executeStep(self, n=1):

        raise NotImplementedError('action must be defined!')

    def executeAll(self):

        raise NotImplementedError('action must be defined!')

    def delay(self, n=100):

        self.delay = n

# Room Class

class Room:

    def \_\_init\_\_(self, location, status="dirty"):

        self.location = location

        self.status = status

# Abstract Agent Class

class Agent(object):

    @abstractmethod

    def \_\_init\_\_(self):

        pass

    @abstractmethod

    def sense(self, environment):

        pass

    @abstractmethod

    def act(self):

        pass

# Vaccum Cleaner Agent Class

class VaccumAgent(Agent):

    def \_\_init\_\_(self):

        pass

    def sense(self, env):

        self.environment = env

    def act(self):

        if self.environment.currentRoom.status == 'dirty':

            return 'clean'

        elif self.environment.currentRoom.location == 'A':

            return 'right'

        else:

            return 'left'

# Environment Class

class TwoRoomVaccumCleanerEnvironment(Environment):

    def \_\_init\_\_(self, agent):

        # Constructor

        self.r1 = Room('A', 'dirty')

        self.r2 = Room('B', 'dirty')

        self.agent = agent

        self.currentRoom = self.r1

        self.delay = 1000

        self.step = 1

        self.action = ""

    def executeStep(self, n=1):

        for \_ in range(0, n):

            self.displayPerception()

            self.agent.sense(self)

            res = self.agent.act()

            self.action = res

            if res == 'clean':

                self.currentRoom.status = 'clean'

            elif res == 'right':

                self.currentRoom = self.r2

            else:

                self.currentRoom = self.r1

            self.displayAction()

            self.step += 1

    def executeAll(self):

        raise NotImplementedError('action must be defined!')

    def displayPerception(self):

        print("Perception at step %d is [%s,%s]" % (

            self.step, self.currentRoom.status, self.currentRoom.location))

    def displayAction(self):

        print(

            "------- Action taken at step %d is [%s]" % (self.step, self.action))

    def delay(self, n=100):

        self.delay = n

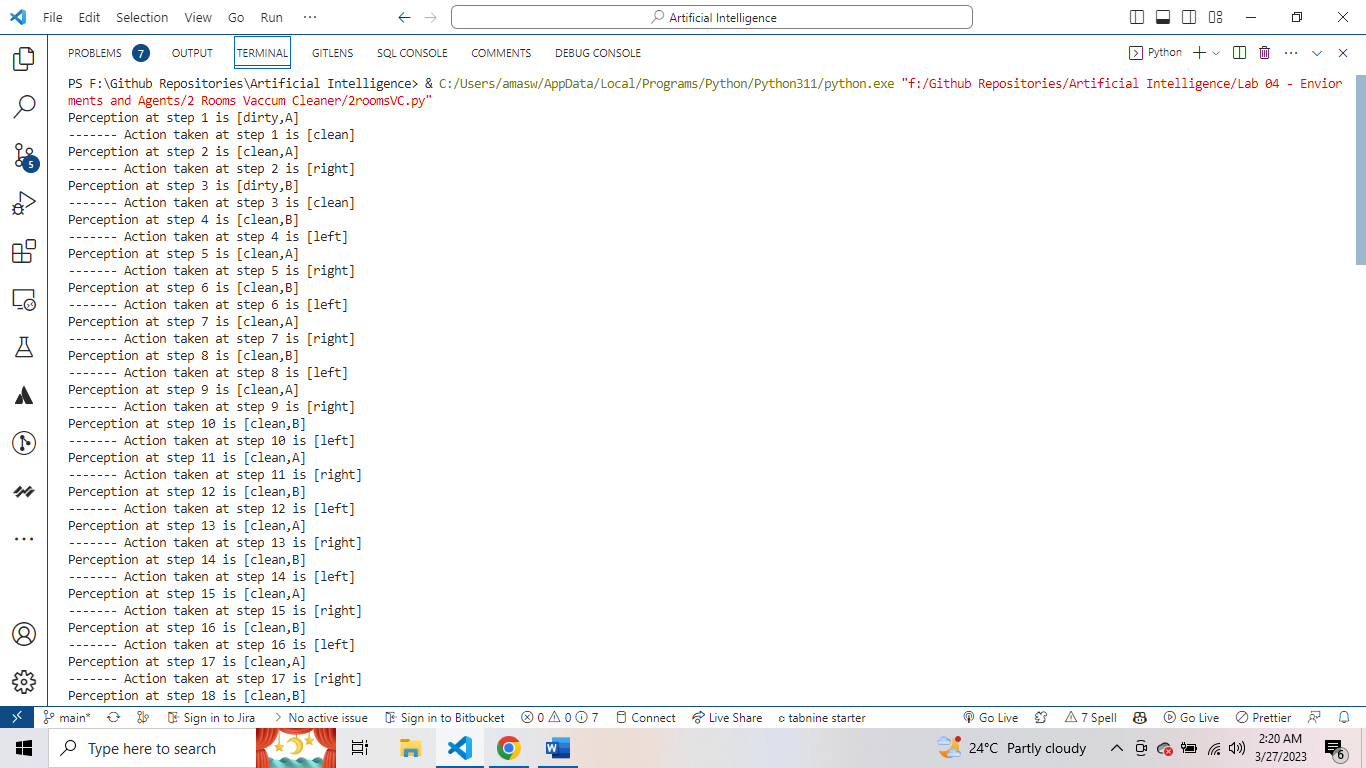
# Test Program

if \_\_name\_\_ == '\_\_main\_\_':

    vcagent = VaccumAgent()

    env = TwoRoomVaccumCleanerEnvironment(vcagent)

    env.executeStep(50)



**Three Room Environment:**

from abc import abstractmethod

# Environment Class

class Environment(object):

    @abstractmethod

    def \_\_init\_\_(self, n):

        self.n = n

    def executeStep(self, n=1):

        raise NotImplementedError('action must be defined!')

    def executeAll(self):

        raise NotImplementedError('action must be defined!')

    def delay(self, n=100):

        self.delay = n

# Room Class

class Room:

    def \_\_init\_\_(self, location, status="dirty"):

        self.location = location

        self.status = status

# Abstract Agent Class

class Agent(object):

    @abstractmethod

    def \_\_init\_\_(self):

        pass

    @abstractmethod

    def sense(self, environment):

        pass

    @abstractmethod

    def act(self):

        pass

# Vaccum Cleaner Agent Class

class VaccumAgent(Agent):

    def \_\_init\_\_(self):

        pass

    def sense(self, env):

        self.environment = env

    def act(self):

        if self.environment.currentRoom.status == 'dirty':

            if self.environment.currentRoom.location == 'A':

                return 'right'

            elif self.environment.currentRoom.location == 'B':

                return 'middle'

            elif self.environment.currentRoom.location == 'C':

                return 'left'

        else:

            return 'clean'

# Environment Class

class TwoRoomVaccumCleanerEnvironment(Environment):

    def \_\_init\_\_(self, agent):

        # Constructor

        self.r1 = Room('A', 'dirty')

        self.r2 = Room('B', 'dirty')

        self.r3 = Room('C', 'dirty')

        self.agent = agent

        self.currentRoom = self.r1

        self.delay = 1000

        self.step = 1

        self.action = ""

    def executeStep(self, n=1):

        for \_ in range(0, n):

            self.displayPerception()

            self.agent.sense(self)

            res = self.agent.act()

            self.action = res

            if res == 'clean':

                self.currentRoom.status = 'clean'

            elif res == 'right':

                self.currentRoom = self.r2

            elif res == 'middle':

                self.currentRoom = self.r3

            elif res == 'left':

                self.currentRoom = self.r1

            self.displayAction()

            self.step += 1

    def executeAll(self):

        raise NotImplementedError('action must be defined!')

    def displayPerception(self):

        print("Perception at step %d is [%s,%s]" % (

            self.step, self.currentRoom.status, self.currentRoom.location))

    def displayAction(self):

        print(

            "------- Action taken at step %d is [%s]" % (self.step, self.action))

    def delay(self, n=100):

        self.delay = n

# Test Program

if \_\_name\_\_ == '\_\_main\_\_':

    vcagent = VaccumAgent()

    env = TwoRoomVaccumCleanerEnvironment(vcagent)

    env.executeStep(5)

**2. Convert the environment to a ‘n’ room environment where n >= 2**

from abc import abstractmethod

class Environment(object):

    @abstractmethod

    def \_\_init\_\_(self, n):

        self.n = n

    def executeStep(self, n=1):

        raise NotImplementedError('action must be defined!')

    def executeAll(self):

        raise NotImplementedError('action must be defined!')

    def delay(self, n=100):

        self.delay = n

class TwoRoomVaccumCleanerEnvironment(Environment):

    def \_\_init\_\_(self, agent):

        self.r1 = Room('A', 'dirty')

        self.r2 = Room('B', 'dirty')

        self.agent = agent

        self.currentRoom = self.r1

        self.delay = 1000

        self.step = 1

        self.action = ""

    def executeStep(self, n=1):

        for \_ in range(0, n):

            self.displayPerception()

            self.agent.sense(self)

            res = self.agent.act()

            self.action = res

            if res == 'clean':

                self.currentRoom.status = 'clean'

            elif res == 'right':

                self.currentRoom = self.r2

            else:

                self.currentRoom = self.r1

            self.displayAction()

            self.step += 1

    def executeAll(self):

        raise NotImplementedError('action must be defined!')

    def displayPerception(self):

        print("Perception at step %d is [%s,%s]" % (

            self.step, self.currentRoom.status, self.currentRoom.location))

    def displayAction(self):

        print(

            "------- Action taken at step %d is [%s]" % (self.step, self.action))

    def delay(self, n=100):

        self.delay = n

class Room:

    def \_\_init\_\_(self, location, status="dirty"):

        self.location = location

        self.status = status

class Agent(object):

    @abstractmethod

    def \_\_init\_\_(self): pass

    @abstractmethod

    def sense(self, environment):

        pass

    @abstractmethod

    def act(self):

        pass

class VaccumAgent(Agent):

    def \_\_init\_\_(self):

        pass

    def sense(self, env):

        self.environment = env

    def act(self):

        if self.environment.currentRoom.status == 'dirty':

            return 'clean'

        if self.environment.currentRoom.location == 'A':

            return 'right'

        return 'left'

if \_\_name\_\_ == '\_\_main\_\_':

    vcagent = VaccumAgent()

    env = NRoomVaccumCleanerEnvironment(vcagent, 5)

    env.executeStep(50)

**3- Does the agent ever stop? If no, can you make it stop? Is your program rational?**

No, the agent does not stop until all the rooms are clean. We can make it stop by modifying the executeAll() method by adding a condition to check if all the rooms are clean before terminating the loop. The program is rational as it cleans the dirty rooms while minimizing the total score.

**4- Score your agent, -1 points for moving from a room, +25 points to clean a room that is dirty, and -10 points if a room is dirty. The scoring will take place after every 1 second.**

We can modify the executeStep() method to score the agent after every second based on the action it takes. The scoring is -1 point for moving from a room, +25 points for cleaning a room that is dirty, and -10 points if a room is dirty.

**5- Convert the agent to a reflex-based agent with a model. Afterwards, take the sensors away from the agents, i.e., now the agent cannot perceive anything. Does your agent still work? If so, then why?**

The reflex-based agent with a model can be created by modifying the act() method to include a model that maps the current room state to an action. The model-based agent will work even if the sensors are taken away because it uses a model to determine the actions based on the current room state. However, if the environment changes, and the model is not updated accordingly, the agent may not perform optimally.