## AI\_LAB\_2

Kola Deepti (120cs0151)

### Vacuum Cleaner Problem with two rooms A and B:

### Utility Based

Code:

import random

class SimpleVacuumEnvironment:

    def \_\_init\_\_(self):

        self.room\_status = {

            'A': random.choice(['Clean', 'Dirty']),

            'B': random.choice(['Clean', 'Dirty'])

        }

        self.agent\_location = random.choice(['A', 'B'])

    def is\_dirty(self, room):

        return self.room\_status[room] == 'Dirty'

    def clean(self, room):

        self.room\_status[room] = 'Clean'

    def move\_agent(self, room):

        self.agent\_location = room

    def display(self):

        print("Room A:", self.room\_status['A'])

        print("Room B:", self.room\_status['B'])

        print("Agent Location:", self.agent\_location)

        print()

class UtilityBasedVacuumAgent:

    def \_\_init\_\_(self, environment):

        self.environment = environment

        self.utilities = {'A': 0, 'B': 0}

    def calculate\_utilities(self):

        for room in self.utilities:

            if self.environment.is\_dirty(room):

                self.utilities[room] = -1  # Negative utility for dirty rooms

            else:

                self.utilities[room] = 1   # Positive utility for clean rooms

    def decide\_action(self):

        current\_room = self.environment.agent\_location

        other\_room = 'B' if current\_room == 'A' else 'A'

        if self.utilities[current\_room] < self.utilities[other\_room]:

            return "Clean" if self.environment.is\_dirty(current\_room) else "Move"

        else:

            return "Move"

    def act(self):

        current\_room, dirt\_status = self.environment.agent\_location, self.environment.is\_dirty(self.environment.agent\_location)

        action = self.decide\_action()

        if action == "Clean":

            self.environment.clean(current\_room)

            print("Agent cleans", current\_room)

        elif action == "Move":

            target\_room = 'B' if current\_room == 'A' else 'A'

            self.environment.move\_agent(target\_room)

            print("Agent moves to", target\_room)

        self.environment.display()

# Example usage

env = SimpleVacuumEnvironment()

agent = UtilityBasedVacuumAgent(env)

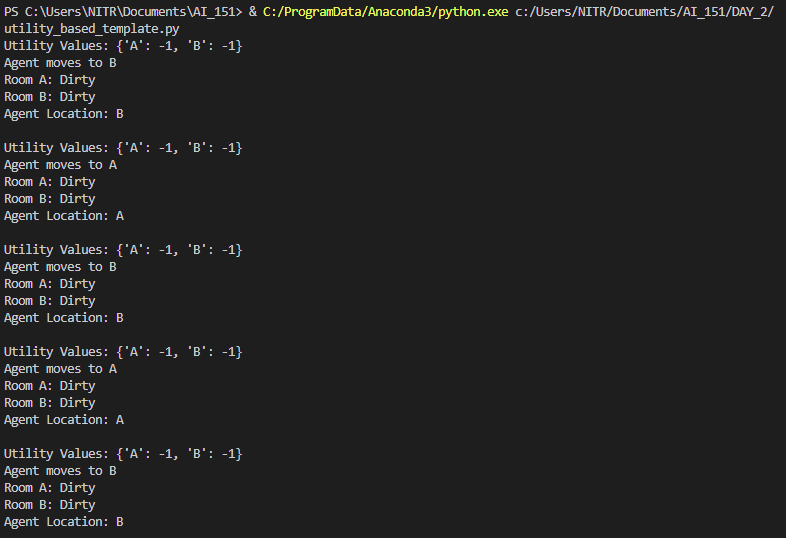
for \_ in range(5):

    agent.calculate\_utilities()

    print(f"Utility Values: {agent.utilities}")

    agent.act()

Output:



### Simple reflex

Code:

import random

class Environment(object):

    def \_\_init\_\_(self):

        self.locationCondition = {'A': '0', 'B': '0'}

        self.locationCondition['A'] = random.randint(0, 1)

        self.locationCondition['B'] = random.randint(0, 1)

class SimpleReflexVacuumAgent(Environment):

    def \_\_init\_\_(self, Environment):

        print (Environment.locationCondition)

        Score = 0

        vacuumLocation = random.randint(0, 1)

        if vacuumLocation == 0:

            print ("Vacuum is randomly placed at Location A.")

            print ("Location A is Dirty.") if Environment.locationCondition['A'] == 1 else print("Location A is Clean.")

            if Environment.locationCondition['A'] == 1:

                Environment.locationCondition['A'] = 0

                Score += 1

                print ("Location A has been Cleaned.")

            print ("Moving to Location B...")

            print ("Location B is Dirty.") if Environment.locationCondition['B'] == 1 else print("Location B is Clean.")

            if Environment.locationCondition['B'] == 1:

                Environment.locationCondition['B'] = 0

                Score += 1

                print ("Location B has been Cleaned.")

            print("Environment is Clean.")

        elif vacuumLocation == 1:

            print ("Vacuum randomly placed at Location B.")

            print ("Location B is Dirty.") if Environment.locationCondition['B'] == 1 else print("Location B is Clean.")

            if Environment.locationCondition['B'] == 1:

                Environment.locationCondition['B'] = 0

                Score += 1

                print ("Location B has been Cleaned.")

            print ("Moving to Location A...")

            print ("Location A is Dirty.") if Environment.locationCondition['A'] == 1 else print("Location A is Clean.")

            if Environment.locationCondition['A'] == 1:

                Environment.locationCondition['A'] = 0

                Score += 1

                print ("Location A has been Cleaned.")

            print("Environment is Clean.")

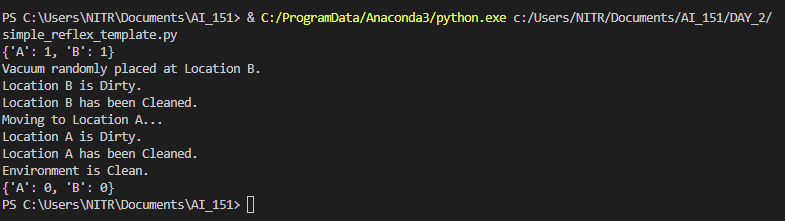
        print (Environment.locationCondition)

        #print ("Performance Measurement: " + str(Score))

theEnvironment = Environment()

theVacuum = SimpleReflexVacuumAgent(theEnvironment)

Output:



### Model Based

Code:

import random

class SimpleVacuumEnvironment:

    def \_\_init\_\_(self):

        # Initialize rooms, room status, and agent location

        self.rooms = ['Room A', 'Room B']

        self.room\_status = {room: 'Dirty' for room in self.rooms}

        self.agent\_location = random.choice(self.rooms)

    def is\_dirty(self, room):

        # Returns if the room is dirty or not

        return self.room\_status[room] == 'Dirty'

    def clean(self, room):

        # Marks the room as clean

        self.room\_status[room] = 'Not Dirty'

    def move\_agent(self, room):

        # Moves the agent to the specified room

        self.agent\_location = room

    def display(self):

        # Display the current state of rooms and agent location

        print("Current state:")

        for room in self.rooms:

            print(f"{room} is {self.room\_status[room]}")

        print(f"Agent is in {self.agent\_location}\n")

class ModelBasedReflexVacuumAgent:

    def \_\_init\_\_(self, environment):

        # Initialize agent with the environment and an internal model

        self.environment = environment

        self.model = {room: 'Dirty' for room in self.environment.rooms}

    def perceive(self):

        # Perceive and return the current room and dirt status

        current\_room = self.environment.agent\_location

        dirt\_status = self.environment.is\_dirty(current\_room)

        return current\_room, dirt\_status

    def update\_model(self, room):

        # Update the internal model when the agent cleans a room

        self.model[room] = 'Not Dirty'

    def decide\_action(self, dirt\_status):

        # Decide whether to clean or move based on perception and model

        current\_room = self.environment.agent\_location

        if dirt\_status:

            return "Clean"

        elif self.model[current\_room] == 'Dirty':

            return "Clean"

        else:

            return "Move"

    def act(self):

        # Perform the chosen action and update environment and model

        current\_room, dirt\_status = self.perceive()

        action = self.decide\_action(dirt\_status)

        if action == "Clean":

            print(f"Agent cleans {current\_room}")

            self.environment.clean(current\_room)

            self.update\_model(current\_room)

        elif action == "Move":

            other\_room = [room for room in self.environment.rooms if room != current\_room][0]

            print(f"Agent moves from {current\_room} to {other\_room}")

            self.environment.move\_agent(other\_room)

# Example usage

env = SimpleVacuumEnvironment()

env.display()

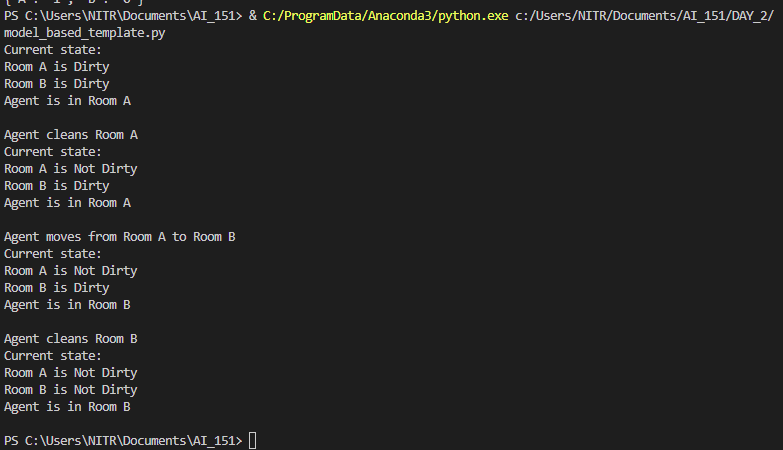
agent = ModelBasedReflexVacuumAgent(env)

for \_ in range(3):

    agent.act()

    env.display()

Output:



### Goal Based

Code:

def vacuum\_world():

        # initializing goal\_state

        # 0 indicates Clean and 1 indicates Dirty

    goal\_state = {'A': '1', 'B': '0'}

    cost = 0

    print("INSTRUCTIONS\nEnter LOCATION A/B in captial letters\nEnter Status O/1 accordingly where 0 means CLEAN and 1 means DIRTY")

    location\_input = input("Enter Location of Vacuum :\n") #user\_input of location vacuum is placed

    status\_input = input("Enter status of " + location\_input +":\n") #user\_input if location is dirty or clean

    status\_input\_complement = input("Enter status of other room :\n")

    #print("Initial Location Condition" + str(goal\_state))

    if location\_input == 'A':

        # Location A is Dirty.

        print("Vacuum is placed in Location A")

        if status\_input == '1':

            print("Location A is Dirty.")

            # suck the dirt  and mark it as clean

            goal\_state['A'] = '1'

            # cost += 1                      #cost for suck

            # print("Cost for CLEANING A " + str(cost))

            print("Location A has been Cleaned.")

            if status\_input\_complement == '1':

                # if B is Dirty

                print("Location B is Dirty.")

                print("Moving right to the Location B. ")

                # cost += 1                       #cost for moving right

                # print("COST for moving RIGHT" + str(cost))

                # suck the dirt and mark it as clean

                goal\_state['B'] = '0'

                # cost += 1                       #cost for suck

                # print("COST for SUCK " + str(cost))

                print("Location B has been Cleaned. ")

            else:

                # print("No action" + str(cost))

                # suck and mark clean

                print("Location B is already clean.")

        if status\_input == '0':

            print("Location A is already clean ")

            if status\_input\_complement == '1':# if B is Dirty

                print("Location B is Dirty.")

                print("Moving RIGHT to the Location B. ")

                # cost += 1                       #cost for moving right

                # print("COST for moving RIGHT " + str(cost))

                # suck the dirt and mark it as clean

                goal\_state['B'] = '0'

                # cost += 1                       #cost for suck

                # print("Cost for SUCK" + str(cost))

                print("Location B has been Cleaned. ")

            else:

                print("No action")

                # print(cost)

                # suck and mark clean

                print("Location B is already clean.")

    else:

        print("Vacuum is placed in location B")

        # Location B is Dirty.

        if status\_input == '1':

            print("Location B is Dirty.")

            # suck the dirt  and mark it as clean

            goal\_state['B'] = '0'

            # cost += 1  # cost for suck

            # print("COST for CLEANING " + str(cost))

            print("Location B has been Cleaned.")

            if status\_input\_complement == '1':

                # if A is Dirty

                print("Location A is Dirty.")

                print("Moving LEFT to the Location A. ")

                # cost += 1  # cost for moving right

                # print("COST for moving LEFT" + str(cost))

                # suck the dirt and mark it as clean

                goal\_state['A'] = '0'

                # cost += 1  # cost for suck

                # print("COST for SUCK " + str(cost))

                print("Location A has been Cleaned.")

        else:

            print(cost)

            # suck and mark clean

            print("Location B is already clean.")

            if status\_input\_complement == '1':  # if A is Dirty

                print("Location A is Dirty.")

                print("Moving LEFT to the Location A. ")

                # cost += 1  # cost for moving right

                # print("COST for moving LEFT " + str(cost))

                # suck the dirt and mark it as clean

                goal\_state['A'] = '1B'

                # cost += 1  # cost for suck

                # print("Cost for SUCK " + str(cost))

                print("Location A has been Cleaned. ")

            else:

                # print("No action " + str(cost))

                # suck and mark clean

                print("Location A is already clean.")

    # done cleaning

    print("GOAL STATE: ")

    print(goal\_state)

    # print("Performance Measurement: " + str(cost))

vacuum\_world()

Output:

