Lab-04
To Understanding how agents and environments interact

Objectives:
The objective of this lab is to show the loop of interaction between the agent and the environment.

Apparatus:

Hardware Requirement
Personal computer.
Software Requirement
Anaconda.

Theory

Agents and environments

A simple reflex agent is the most basic of the intelligent agents out there. It performs actions based on a current situation. When something happens in the environment of a simple reflex agent, the agent quickly scans its knowledge base for how to respond to the situation at-hand based on pre-determined rules.

It would be like a home thermostat recognizing that if the temperature increases to 75 degrees in the house, the thermostat is prompted to kick on. It doesn't need to know what happened with the temperature yesterday or what might happen tomorrow. Instead, it operates based on the idea that if _____ happens, _____ is the response.

Simple reflex agents are just that - simple. They cannot compute complex equations or solve complicated problems. They work only in environments that are fully-observable in the current percept, ignoring any percept history. If you have a smart light bulb, for example, set to turn on at 6 p.m. every night, the light bulb will not recognize how the days are longer in summer and the lamp is not needed until much later. It will continue to turn the lamp on at 6 p.m. because that is the rule it follows. Simple reflex agents are built on the condition-action rule.

These agents simply decide actions based on their current percept. By identifying that certain actions are warranted in certain conditions,

Python code for the abstract Environment

```
@author: hira farman
'''
from abc import abstractmethod
 class Environment(object):
    '''    classdocs
           '''


    @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!')
     defdelay(self,n=100):
 self.delay = n
```

For the Two Room Vacuum Cleaner Environment

```
 from com.environment import Environment  from com.environment import Room fromcom.agent import
VaccumAgent
 class TwoRoomVaccumCleanerEnvironment(Environment.Environment):
    '''    classdocs    '''      def
__init__(self, agent):
            '''
            Constructor       '''
            self.r1       =       Room.Room('A','dirty')
            self.r2       =       Room.Room('B','dirty')
            self.agent                =                agent
            self.currentRoom           =           self.r1
            self.delay = 1000          self.step    =    1
            self.action = ""
            defexecuteStep(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':
```

University Of Karachi

```
            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))
            defdelay(self,n=100):
            self.delay = n
```

Room class

```
@author: hira farman
'''          class
Room:
   def __init__(self,location,status="dirty"):
            self.location = location     self.status = status
```

Abstract agent

```
@author: hira farman
'''
from abc import abstractmethod
 classAgent(object):
   '''     classdocs
           ''


   @abstractmethod          def
__init__(self):     pass


            @abstractmethod
            defsense(self,environment):
            pass


   @abstractmethod          def
act(self):
            pass
```

University Of Karachi

Vaccum cleaner Agent

```
from com.agent import Agent
 class VaccumAgent(Agent.Agent):
    '''    classdocs
           '''


    def__init__(self):        '''
           Constructor
'''        pass
    defsense(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'
```

Test program

```
 if    __name__    ==    '__main__':                  vcagent    =
VaccumAgent.VaccumAgent()                              env     =
TwoRoomVaccumCleanerEnvironment(vcagent)
           env.executeStep(50)
```

LAB TASKS

1. Run the two room vacuum cleaner agent program and understand it. Convert the program to a Three room environment.
2. Convert the environment to a ‘n’ room environment where n >= 2
3. Does the agent ever stop? If no, can you make it stop? Is your program rational?
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
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?