Experiment 1: Intelligent Agent

Samkit Shah
2019130060
Batch C
TE Comps
October 5, 2021

Aim: To develop a simple intelligent agent using python and identify a task environment for it.

Theory:

1. Agent: An agent is anything that can perceive its environment through its sensors and can act upon it with the help of its actuators.
2. Task Environment: Task environments are essentially the “problems” to which rational agents are the “solutions.”
3. Performance measurement: Performance measures are the desirable qualities that we want our agent to have.
4. Environment: Environment is everything present around the agent. Agent perceives the environment.
5. Actuators: Actuators are the elements of the agent through which the agent performs certain tasks.
6. Sensors: Sensors are the elements of the agent through which the agent perceives the environment.

Agent	Performance Measures	Environment	Actuators	Sensors
Light Adjusting Assistant	Make adjustments in the lighting according to the weather.	Laptop	Screen, Speaker	Weather Sensors.

Code:

```
import pyttsx3
import json
import speech_recognition as sr
import datetime
import webbrowser
import os
import requests
from datetime import datetime, timedelta
import time


engine = pyttsx3.init('sapi5')
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[0].id)

light = "OFF"

def takeCommand():

    r = sr.Recognizer()

    with sr.Microphone() as source:

        print("Speak now: ")
        r.pause_threshold = 1
        audio = r.listen(source)

    try:
        print("Evaluating the speech: ")
        query = r.recognize_google(audio, language ='en-in')
        print(f"speech: {query}\n")

    except Exception as e:
        print(e)
        print("Unable to Recognize your voice.")
        return "None"
```

```
    return query


def speak(audio):
    engine.say(audio)
    engine.runAndWait()

def turnonlight():
    print("Lights turned on !")
    global light
    light = "ON"

def turnofflight():
    print("Lights turned off !")
    global light
    light = "OFF"

def dimlight():
    print("Lights dimmed !")
    global light
    light = "DIM"

def brightenlight():
    print("Lights brightened !")
    global light
    light = "BRIGHT"



if __name__ == '__main__':
    # clear = lambda: os.system('cls')
    # clear()

    # speak("Hello! How may I help you ?")

    # hour = int(datetime.datetime.now().hour)
    # if hour>= 0 and hour<8:
    #     turnofflight()
```

```
    # elif hour>= 8 and hour<12:
    #     turnonlight()

    # elif hour>= 12 and hour<18:
    #     dimlight()

    # else:
    #     brightenlight()

    while True:

        # command = takeCommand().lower()
        # matches = ["site", "website", "webpage"]
        # exit = ["sleep","exit","stop","bye","goodbye"]
        # lights = ["light","adjust","change","adjustment"]

        # if 'time' in command:
        #     strTime = datetime.datetime.now().strftime("%H %M %S")
        #     speak(f"Sir, the time is {strTime}")

        # elif any(x in command for x in matches):
        #     speak('Speak the name of the website')
        #     website = takeCommand()
        #     webbrowser.open(website)


        # elif "weather" in command:
        url = "https://community-open-weather-map.p.rapidapi.com/weather"
        # speak("Please speak the city name")
        city = 'Pune'
        print(city)
        querystring =
{"q":city+",in","lat":"0","lon":"0","id":"2172797","lang":"null","units":"
metric","mode":"JSON"}
        headers = {
            'x-rapidapi-host':
"community-open-weather-map.p.rapidapi.com",
            'x-rapidapi-key':
"8e5b5c251amshc212add6414d43cp14c4c0jsn5ddd339d73e6"
            }
```

```
        try:

            response = requests.request("GET", url, headers=headers,
params=querystring)
        # print(response.text)
            x = response.text
            x = json.loads(x)
            print(f"The weather in {city} is {x['weather'][0]['main']},
the temperature is {x['main']['temp']} and it feels like
{x['main']['feels_like']} as the temperature ranges from
{x['main']['temp_min']} to {x['main']['temp_max']} with humidity of
{x['main']['humidity']}")
            # speak(f"The weather in {city} is {x['weather'][0]['main']},
the temperature is {x['main']['temp']} and it feels like
{x['main']['feels_like']} as the temperature ranges from
{x['main']['temp_min']} to {x['main']['temp_max']} with humidity of
{x['main']['humidity']}")
            if(x['weather'][0]['main'] == "Clouds" and light != "ON"):
                turnonlight()
            elif(x['weather'][0]['main'] == "Clear" and light != "OFF"):
                turnofflight()
            elif(x['weather'][0]['main'] == "Smoke" or
x['weather'][0]['main'] == "Mist" and light != "BRIGHT"):
                brightenlight()
            elif(x['weather'][0]['main'] == "Haze" and light != "DIM"):
                dimlight()
            else:
                print("Light settings already adjusted")
        except:
            speak("Unable to fetch weather details")
        dt = datetime.now() + timedelta(seconds=300)
        # dt = dt.replace(minute=10)
        count = 1
        while datetime.now() < dt:
            print(count)
            count+=1
            time.sleep(1)

        # elif any(x in command for x in lights):
```

```
        #     speak("What adjustment do you want to make ?")
        #     speak("Turn on the lights, turn off the lights, dim the
light, brighten the light")
        #     print("Turn on the lights, turn off the lights, dim the
light, brighten the light")

        #     query = takeCommand()
        #     if "dim" in query:
        #         dimlight()
        #     elif "brighten" in query:
        #         brightenlight()
        #     elif "on" in query:
        #         turnonlight()
        #     elif "off" or "of" in query:
        #         turnofflight()


        # elif any(x in command for x in exit):
        #     speak("Bye, Have a good day !")
        #     break
```

Conclusion: While performing the experiment for intelligent agents I learned about intelligent agents and how to make best use of them by fetching weather information and making adjustments in the lighting.