**Artificial Intelligence**

**Vacuum Cleaner using AI Agents**

**INTRODUCTION**

An agent can be anything that perceives its environment through sensors and act upon that environment through actuators. An Agent runs in the cycle of **perceiving**, **thinking**, and **acting**. An agent can be:

* **Human-Agent:** A human agent has eyes, ears, and other organs which work for sensors and hand, legs, vocal tract work for actuators.
* **Robotic Agent:** A robotic agent can have cameras, infrared range finder, NLP for sensors and various motors for actuators.
* **Software Agent:** Software agent can have keystrokes, file contents as sensory input and act on those inputs and display output on the screen.

Hence the world around us is full of agents such as thermostat, cellphone, camera, and even we are also agents.

**INTELLIGENT AGENTS**

An intelligent agent is an autonomous entity which act upon an environment using sensors and actuators for achieving goals. An intelligent agent may learn from the environment to achieve their goals. A thermostat is an example of an intelligent agent.

Following are the main four rules for an AI agent:

* **Rule 1:** An AI agent must have the ability to perceive the environment.
* **Rule 2:** The observation must be used to make decisions.
* **Rule 3:** Decision should result in an action.
* **Rule 4:** The action taken by an AI agent must be a rational action.

**PEAS REPRESENTATION**

PEAS is a type of model on which an AI agent works upon. When we define an AI agent or rational agent, then we can group its properties under PEAS representation model. It is made up of four words:

* **P:** Performance measure
* **E:** Environment
* **A:** Actuators
* **S:** Sensors

Here performance measure is the objective for the success of an agent's behavior.

**VACUUM CLEANER**

In this project, I have built a simple Vacuum Cleaner that works with the help of AI agents. In real world implementation, we can use PEAS representation to create a better and efficient model.

Table : A PEAS representation example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Agent** | **Performance Measure** | **Environment** | **Actuators** | **Sensors** |
| Vacuum Cleaner | * Cleanness * Efficiency * Battery life * Security | * Room * Table * Wood floor * Carpet * Various obstacles | * Wheels * Brushes * Vacuum Extractor | * Camera * Dirt detection sensor * Cliff sensor * Bump Sensor * Infrared Wall Sensor |

In this environment, the cleaning agents clean up dirt and communicate with each other. They identify which room (circle) has the dirt (according to the color) and clean up the space.

* **Dirt**

A dirt is a passive object that is characterized by its color - orange. Dirt can be cleaned (removed from the circle) by the cleaning agents and created (placed in the circle) by user agents.

* **Agents**

### **Cleaning Agents**

It is an autonomous actor that are capable of performing actions within the environment. It is the job of agent to clean the circle of any dirt that may be present.

### **User Agents**

It is an autonomous actor that, instead of cleaning, makes the circle dirty. A user drops orange on its current location.

* **Turtle Module in Python**

“Turtle” is a Python feature like a drawing board or a canvas, which lets us command a turtle to draw all over it. To make use of the turtle methods and functionalities, I have imported turtle. “turtle” comes packed with the standard Python package and need not be installed externally.

**CODE**

from turtle import Turtle, Screen

from random import choice

from time import sleep

from queue import SimpleQueue

w: int

w, h = (853, 480)

wn = Screen()

wn.screensize(w, h)

wn.bgcolor("#d3d3d3")

Room\_state = {"Clean": "#FFFFFF",

"Dirty": "#b5651d"}

cleaned = 0

def filler(t, color, delay=0, vacclean = False):

global cleaned

t.fillcolor(color)

t.penup()

if color == Room\_state['Clean']:

sleep(delay) #To avoid cleaning immediately

if vacclean:

cleaned += 1

t.begin\_fill()

t.circle(130)

t.end\_fill()

def setup():

A = Turtle() # Draws Circle in A

B = Turtle() # Draws Circle in B

X = Turtle() # Text Below A

Y = Turtle() # Text Below B

A.ht()

B.ht()

X.ht()

Y.ht()

A.speed(100)

B.speed(100)

X.speed(100)

Y.speed(100)

A.penup()

B.penup()

X.penup()

Y.penup()

A.setpos(-w / 4, -120)

B.setpos(w / 4, -120)

X.setpos(-w / 4, -200)

Y.setpos(w / 4, -200)

A.pendown()

B.pendown()

filler(A, Room\_state['Clean'], False)

filler(B, Room\_state['Clean'], False)

# Creates rooms and boundary

t1 = Turtle()

t1.ht()

t1.speed(20)

t1.penup()

t1.setposition(w / 2, h / 2)

t1.pendown()

t1.pensize(10)

t1.right(90)

t1.forward(h)

t1.right(90)

t1.forward(w)

t1.right(90)

t1.forward(h)

t1.right(90)

t1.forward(w)

t1.backward(w / 2)

t1.right(90)

t1.pensize(5)

t1.forward(h - 90)

t1.penup()

t1.setpos(-w / 4, h / 2 - 70)

t1.write("Room A", align="center", font=("Arial", 20, "normal"))

t1.setpos(w / 4, h / 2 - 70)

t1.write("Room B", align="center", font=("Arial", 20, "normal"))

return A, B, X, Y

A, B, X, Y = setup()

#Vaccum Cleaner

C = Turtle()

C.speed(8)

C.penup()

C.shape("circle")

C.setpos(A.xcor(), A.ycor() + 130)

count = 1

iter = Turtle()

cleanwriter = Turtle()

iter.ht()

cleanwriter.ht()

iter.penup()

cleanwriter.penup()

iter.setpos(0, -h / 2 + 50)

cleanwriter.setpos(0, -h / 2 + 20)

room\_state = list(Room\_state.keys())

state = SimpleQueue()

state.put\_nowait(((choice(room\_state)), choice(room\_state)))

while True:

iter.clear()

cleanwriter.clear()

iter.write("Iteration : " + str(count), align="center", font=("Arial", 16, "normal"))

cleanwriter.write("Times Cleaned : " + str(cleaned), align="center", font=("Arial", 16, "normal"))

condition = state.get\_nowait()

stateA = condition[0]

stateB = condition[1]

X.clear()

Y.clear()

nextA = choice(room\_state)

nextB = choice(room\_state)

state.put\_nowait((nextA, nextB))

filler(A, Room\_state[stateA])

filler(B, Room\_state[stateB])

X.write("Now : " + stateA + "\nNext : " + nextA, align="center", font=("Arial", 16, "normal"))

Y.write("Now : " + stateB + "\nNext : " + nextB, align="center", font=("Arial", 16, "normal"))

print("\nA : " + stateA, "\tB : " + stateB)

if stateA == 'Dirty' and stateB == 'Dirty':

if C.xcor() < 0:

print("Both Dirty, Cleaned A going to B")

#PyTypeChecker for inspection given from here on

filler(A, Room\_state['Clean'], 0.5, True)

stateA = 'Clean'

C.setpos(B.xcor(), B.ycor() + 130)

filler(B, Room\_state['Clean'], 0.5, True)

stateB = 'Clean'

elif C.xcor() > 0:

print("Both Dirty, Cleaned B going to A")

filler(B, Room\_state['Clean'], 0.5, True)

stateB = 'Clean'

C.setpos(A.xcor(), A.ycor() + 130)

filler(A, Room\_state['Clean'], 0.5, True)

stateA = 'Clean'

if stateA == 'Dirty':

print("Cleaned A")

C.goto(A.xcor(), A.ycor() + 130)

filler(A, Room\_state['Clean'], 0.3, True)

elif stateB == 'Dirty':

print("Cleaned B")

C.goto(B.xcor(), B.ycor() + 130)

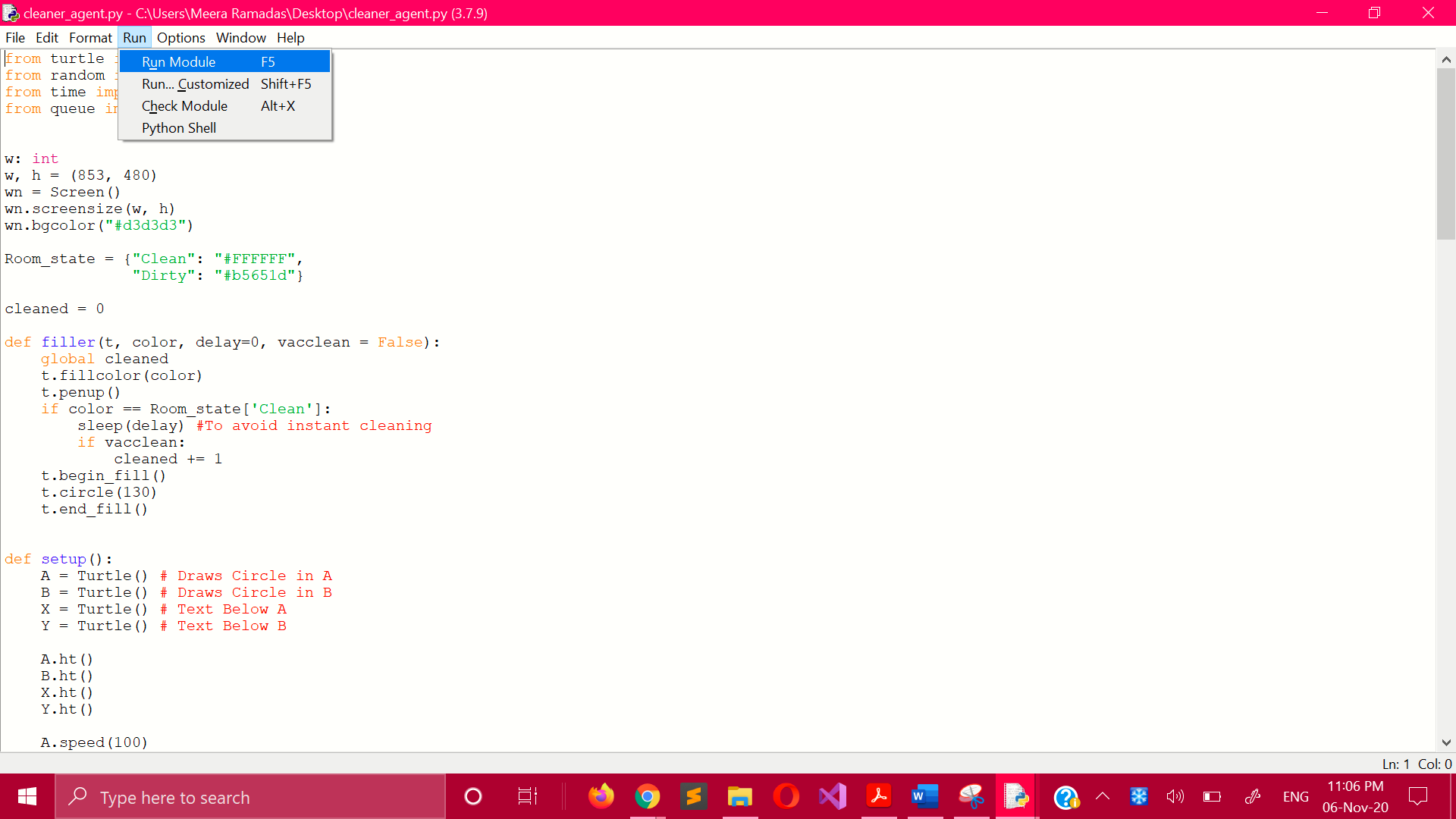
filler(B, Room\_state['Clean'], 0.3, True)

count += 1

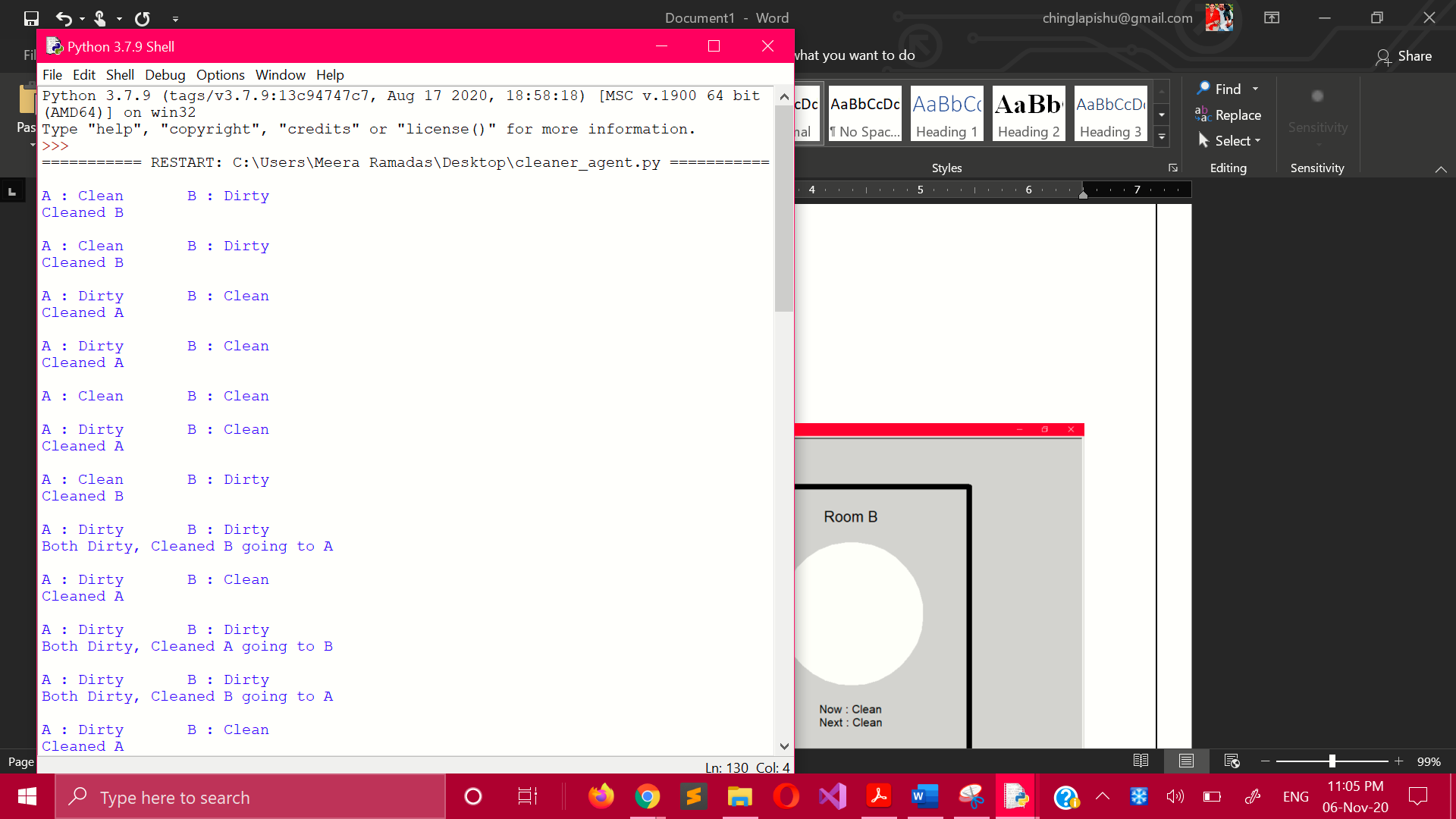
sleep(0.5)

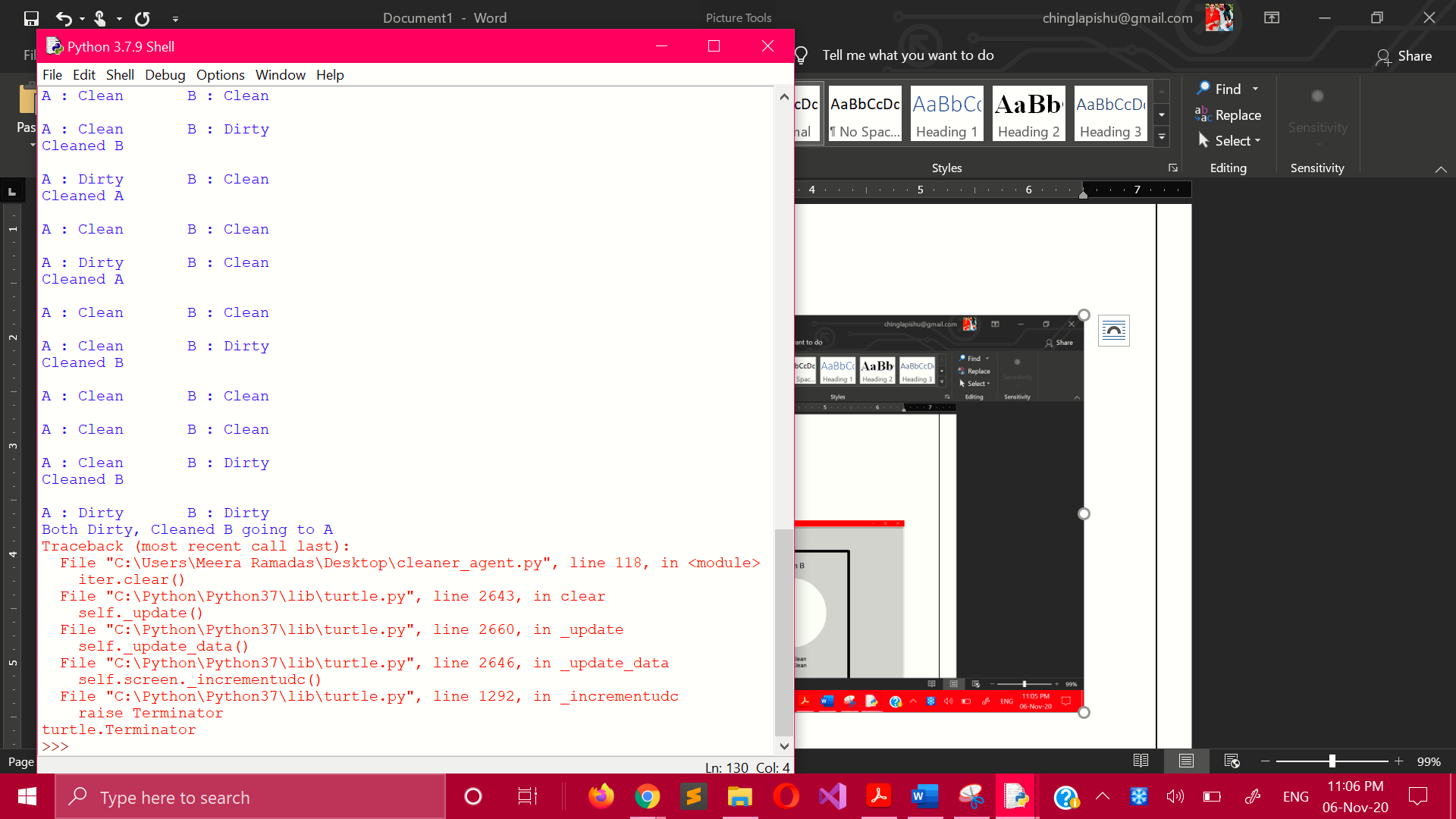
**OUTPUT**

**1. Run the code**



**2. Python Shell**





**3. Turtle Graphics**

