## Indian Institute of Information Technology Surat

****

# Lab Report on

# Artificial Intelligence (CS 701) Practical

**Submitted by**

### [RAHUL KUMAR SINGH] (UI21CS44)

**Course Faculty**

### Dr. Ritesh Kumar

### Mrs. Archana Balmik

## Department of Computer Science and Engineering

## Indian Institute of Information Technology Surat

## Gujarat-394190, India

**Aug-2024**

## Lab No: 12

## Aim:

Write a program to implement the Wumpus World problem that enables an agent to pick up the gold while considering all environmental conditions.

## Description:

* 4x4 grid world designed for the Wumpus World problem simulation.
* The AI agent navigates using sensors, avoiding pits and the Wumpus while searching for gold.
* Uses decision-making logic to balance safety and exploration based on sensed stench, breeze, or glitter.
* Simulation concludes when gold is found, the agent falls into a pit, or the Wumpus kills it.
* Agent strategy involves visiting unexplored safe positions and learning from sensors to avoid hazards.

## Code:

**import random**

**class WumpusWorld:**

**def \_\_init\_\_(self):**

**self.grid\_size = 4**

**self.world = [["" for \_ in range(self.grid\_size)] for \_ in range(self.grid\_size)]**

**self.agent\_position = (0, 0)**

**self.gold\_position = self.\_place\_item("G")**

**self.wumpus\_position = self.\_place\_item("W")**

**self.pit\_positions = [self.\_place\_item("P") for \_ in range(3)]**

**self.visited = set()**

**self.safe\_moves = set()**

**self.sensors\_log = []**

**def \_place\_item(self, item):**

**while True:**

**x, y = random.randint(0, 3), random.randint(0, 3)**

**if (x, y) != (0, 0) and not self.world[x][y]:**

**self.world[x][y] = item**

**return (x, y)**

**def sensors(self, x, y):**

**sensors = []**

**for dx, dy in [(-1, 0), (1, 0), (0, -1), (0, 1)]:**

**nx, ny = x + dx, y + dy**

**if 0 <= nx < self.grid\_size and 0 <= ny < self.grid\_size:**

**if "W" in self.world[nx][ny]:**

**sensors.append("Stench")**

**if "P" in self.world[nx][ny]:**

**sensors.append("Breeze")**

**if "G" in self.world[x][y]:**

**sensors.append("Glitter")**

**return sensors**

**def move\_agent(self, move):**

**x, y = self.agent\_position**

**if move == "up" and x > 0:**

**x -= 1**

**elif move == "down" and x < self.grid\_size - 1:**

**x += 1**

**elif move == "left" and y > 0:**

**y -= 1**

**elif move == "right" and y < self.grid\_size - 1:**

**y += 1**

**self.agent\_position = (x, y)**

**self.visited.add(self.agent\_position)**

**return self.check\_status()**

**def check\_status(self):**

**x, y = self.agent\_position**

**if self.agent\_position == self.gold\_position:**

**return "Gold found! You win!"**

**elif self.agent\_position == self.wumpus\_position:**

**return "Eaten by Wumpus! Game over!"**

**elif self.agent\_position in self.pit\_positions:**

**return "Fell into a pit! Game over!"**

**else:**

**return "Safe, but proceed carefully."**

**def get\_possible\_moves(self, x, y):**

**moves = []**

**if x > 0: moves.append("up")**

**if x < self.grid\_size - 1: moves.append("down")**

**if y > 0: moves.append("left")**

**if y < self.grid\_size - 1: moves.append("right")**

**return moves**

**def ai\_decision(self):**

**x, y = self.agent\_position**

**sensors = self.sensors(x, y)**

**self.sensors\_log.append((self.agent\_position, sensors))**

**if "Glitter" in sensors:**

**return "Gold found! You win!"**

**possible\_moves = self.get\_possible\_moves(x, y)**

**safe\_moves = [move for move in possible\_moves if self.simulate\_move(move) not in self.visited]**

**random.shuffle(safe\_moves)**

**if safe\_moves:**

**chosen\_move = safe\_moves[0]**

**else:**

**chosen\_move = random.choice(possible\_moves)**

**return self.move\_agent(chosen\_move)**

**def simulate\_move(self, move):**

**x, y = self.agent\_position**

**if move == "up" and x > 0:**

**return (x - 1, y)**

**elif move == "down" and x < self.grid\_size - 1:**

**return (x + 1, y)**

**elif move == "left" and y > 0:**

**return (x, y - 1)**

**elif move == "right" and y < self.grid\_size - 1:**

**return (x, y + 1)**

**return (x, y)**

**def play(self):**

**print("Starting AI Agent Simulation in the Wumpus World...")**

**while True:**

**status = self.ai\_decision()**

**print(f"\nAgent is at {self.agent\_position}.")**

**print("Sensors:", self.sensors(\*self.agent\_position))**

**print(status)**

**if "Game over" in status or "win" in status:**

**break**

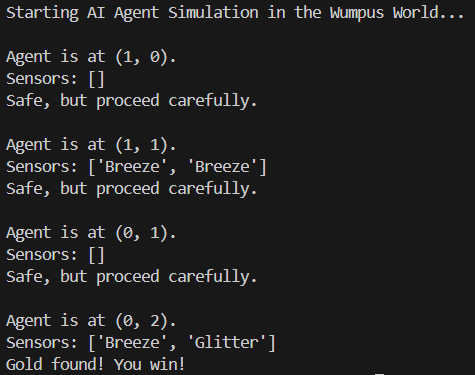
**if \_\_name\_\_ == "\_\_main\_\_":**

**game = WumpusWorld()**

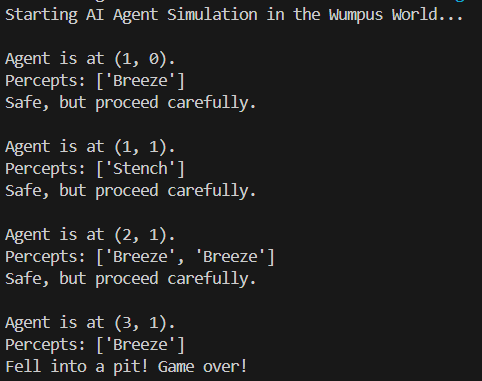
**game.play()**

## Output:

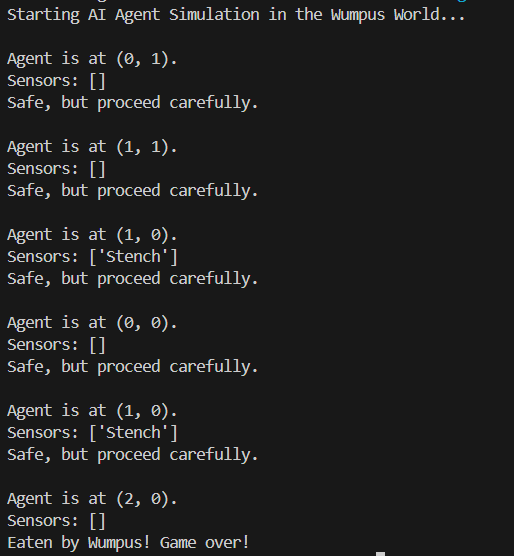
**Outcome - Gold (Received):**

****

**Outcome - Pit (Fallen):**

****

**Outcome - Wumpus (Eaten):**

****

## Conclusion:

* In Wumpus World, the AI uses sensor-based logic to make strategic and safe moves toward the gold.
* AI carefully balances exploration and caution, ensuring minimal risk while seeking optimal outcomes.
* AI employs decision-making rules, leveraging sensory input to avoid the Wumpus and deadly pits.
* Player’s challenge lies in observing AI behavior, understanding environmental cues, and predicting moves.