import random

# Constants

WORLD\_SIZE = 4

NUM\_PITS = 3

WUMPUS\_LOCATION = (3, 3)

GOLD\_LOCATION = (2, 2)

DIRECTIONS = ['UP', 'RIGHT', 'DOWN', 'LEFT']

class WumpusWorld:

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

self.grid = [['' for \_ in range(WORLD\_SIZE)] for \_ in range(WORLD\_SIZE)]

self.agent\_location = (0, 0)

self.agent\_direction = 'RIGHT'

self.has\_gold = False

self.alive = True

self.performance = 0 # Performance measure (score)

self.setup\_world()

def setup\_world(self):

self.place\_pits()

self.place\_wumpus()

self.place\_gold()

def place\_pits(self):

placed = 0

while placed < NUM\_PITS:

x, y = random.randint(0, WORLD\_SIZE - 1), random.randint(0, WORLD\_SIZE - 1)

if (x, y) != (0, 0) and self.grid[x][y] == '':

self.grid[x][y] = 'PIT'

placed += 1

def place\_wumpus(self):

x, y = WUMPUS\_LOCATION

self.grid[x][y] = 'WUMPUS'

def place\_gold(self):

x, y = GOLD\_LOCATION

self.grid[x][y] = 'GOLD'

def perceive(self):

x, y = self.agent\_location

perception = []

# Stench: Wumpus nearby

if self.is\_adjacent(x, y, 'WUMPUS'):

perception.append('STENCH')

# Breeze: Pit nearby

if self.is\_adjacent(x, y, 'PIT'):

perception.append('BREEZE')

# Glitter: Gold in current cell

if self.grid[x][y] == 'GOLD':

perception.append('GLITTER')

return perception

def is\_adjacent(self, x, y, item):

adjacent = [(x - 1, y), (x + 1, y), (x, y - 1), (x, y + 1)]

for i, j in adjacent:

if 0 <= i < WORLD\_SIZE and 0 <= j < WORLD\_SIZE:

if self.grid[i][j] == item:

return True

return False

def move\_forward(self):

x, y = self.agent\_location

if self.agent\_direction == 'UP' and x > 0:

x -= 1

elif self.agent\_direction == 'DOWN' and x < WORLD\_SIZE - 1:

x += 1

elif self.agent\_direction == 'LEFT' and y > 0:

y -= 1

elif self.agent\_direction == 'RIGHT' and y < WORLD\_SIZE - 1:

y += 1

else:

print("BUMP into wall.")

return # Bump into wall

self.agent\_location = (x, y)

self.performance -= 1 # Cost for moving

def turn\_left(self):

idx = DIRECTIONS.index(self.agent\_direction)

self.agent\_direction = DIRECTIONS[(idx - 1) % 4]

self.performance -= 1

def turn\_right(self):

idx = DIRECTIONS.index(self.agent\_direction)

self.agent\_direction = DIRECTIONS[(idx + 1) % 4]

self.performance -= 1

def grab(self):

x, y = self.agent\_location

if self.grid[x][y] == 'GOLD':

print("Gold grabbed!")

self.grid[x][y] = ''

self.has\_gold = True

self.performance += 100

else:

print("No gold here.")

self.performance -= 1

def check\_status(self):

x, y = self.agent\_location

if self.grid[x][y] == 'PIT':

print("Fell into PIT!")

self.alive = False

self.performance -= 1000

elif self.grid[x][y] == 'WUMPUS':

print("Eaten by WUMPUS!")

self.alive = False

self.performance -= 1000

def print\_world(self):

print("\nWorld Grid (Hidden):")

for row in self.grid:

print(row)

print(f"\nAgent Location: {self.agent\_location}, Direction: {self.agent\_direction}")

print(f"Performance: {self.performance}")

print(f"Gold Collected: {'Yes' if self.has\_gold else 'No'}")

print(f"Alive: {'Yes' if self.alive else 'No'}")

class Agent:

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

self.world = world

def act(self):

if not self.world.alive:

print("Agent is dead. Cannot act.")

return False

perceptions = self.world.perceive()

print(f"Perceptions: {perceptions}")

if 'GLITTER' in perceptions:

self.world.grab()

elif 'STENCH' in perceptions or 'BREEZE' in perceptions:

self.world.turn\_right()

else:

self.world.move\_forward()

self.world.check\_status()

return self.world.alive and not self.world.has\_gold

# Simulation

world = WumpusWorld()

agent = Agent(world)

steps = 0

while agent.act():

steps += 1

print(f"\n--- Step {steps} ---")

world.print\_world()

if steps > 15:

print("Too many steps. Ending simulation.")

break

print("\nFinal World State:")

world.print\_world()