

NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO VISVESVARAYA TECHNOLOGICAL
UNIVERSITY,BELGAUM, APPROVED BY AICTE & GOVT.OF KARNATAKA



Artificial Intelligence & NEURAL Network With Python Lab

Report On : WUMPUS WORLD PROBLEM

*Submitted in partial fulfillment of the requirement for the award of Degree
of Bachelor of Engineering
in
Computer Science and Engineering*

Submitted by: Bibek Yadav 1NT19CS055

Under the Guidance of
Ms. CHAITRA H V

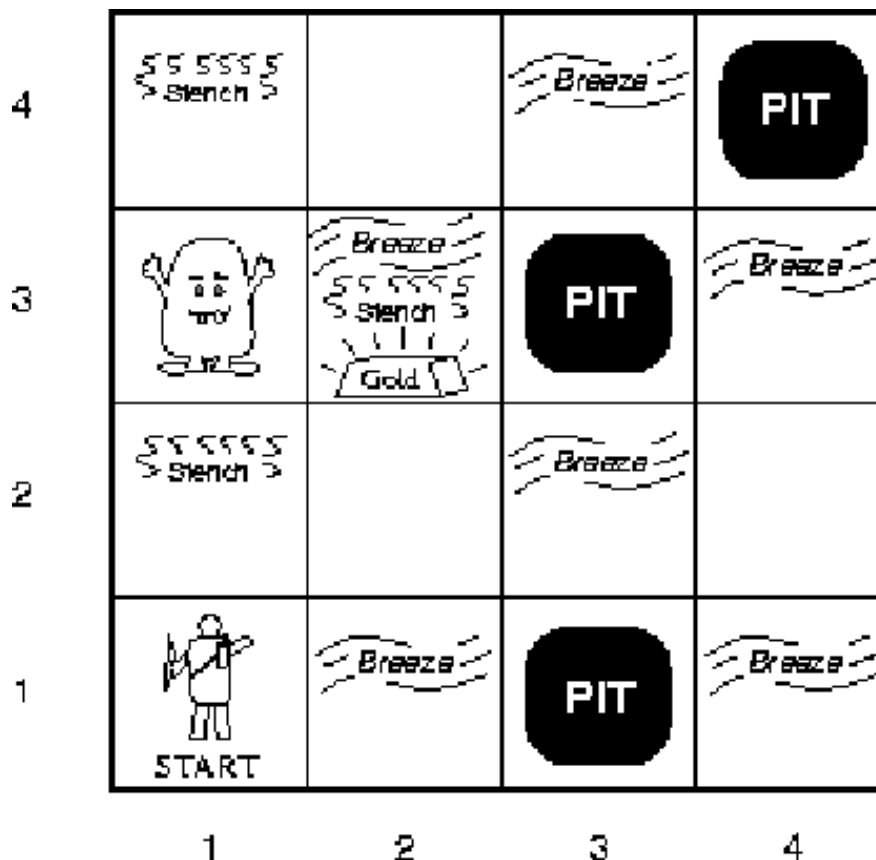
Assistant Professor, Dept. of CS&E, NMIT



WUMPUS WORLD PROGRAM

PROBLEM STATEMENT:

Write a python program to implement The Wumpus world problem. The problem deals with an AI robot navigating its way through a 4x4 puzzle to try and find gold. The robot must safely navigate its way around bottomless pits of death and evil Wumpus creatures to locate the gold hidden on the board. After it has successfully found the gold, it must safely navigate its way back to the starting point. The robot must use its light sensors and the signals sent to it at each square to determine which way to properly navigate to reach its goal. – Give the PEAS description and analyses the game wrt to knowledge base and how AGENT can succeed in grabbing the Gold and exit from the Wumpus World. Simulate the 3 different scenario, of Wumpus World.



PEAS DESCRIPTION:

PEAS represents Performance Measures, Environment, Actuators, and Sensors. The PEAS description helps in grouping the agents.

PEAS Description for the Wumpus World problem:

1. Performance measures:

- Agent gets the gold and return back safe = *+1000 points*
- Agent dies = *-1000 points*
- Each move of the agent = *-1 point*
- Agent uses the arrow = *-10 points*

2. Environment:

- A cave with *16(4×4)* rooms
- Rooms adjacent (not diagonally) to the Wumpus are stinking
- Rooms adjacent (not diagonally) to the pit are breezy
- The room with the gold glitters
- Agent's initial position – *Room[1, 1]* and facing right side
- Location of Wumpus, gold and *3 pits* can be anywhere, except in *Room[1, 1]*.

3. Actuators:

Devices that allow the agent to perform the following actions in the environment.

- Move forward
- Turn right
- Turn left
- Shoot
- Grab
- Release

4. Sensors:

Devices which helps the agent in sensing the following from the environment.

- Breeze
- Stench
- Glitter
- Scream (When the Wumpus is killed)
- Bump (when the agent hits a wall)

Wumpus World Characterization:

- **Partially Observable:** knows only the local perceptions
- **Deterministic:** outcome is precisely specified
- **Sequential:** subsequent level of actions performed
- **Static:** Wumpus, pits are immobile
- **Discrete:** discrete environment
- **Single-agent:** The knowledge-based agent is the only agent whereas the wumpus is considered as the environment's feature.

PYTHON CODE:

```
def learnagent(world,i,j):  
    '''Function for an agent to know what position contains which environment objects''' if  
        (world[i][j]==9):
```

agi,agj=i,j

```
        print("\nNow the agent is at "+str(agi)+" "+str(agj))
        print("You came across a stench")
        return agi,agj elif
(world[i][j]==8):
    agi,agj=i,j
    print("\nNow the agent is at "+str(agi)+" "+str(agj))
    print("You came across a glitter") return agi,agj elif
(world[i][j]==7):
    agi,agj=i,j
    print("\nNow the agent is at "+str(agi)+" "+str(agj))
    print("You came across a pit")
    return -5,-5
elif (world[i][j]==6):
    agi,agj=i,j
    print("\nNow the agent is at "+str(agi)+" "+str(agj))
    print("You found gold")
    return -4,-4
elif (world[i][j]==5):
    agi,agj=i,j
    print("\nNow the agent is at "+str(agi)+" "+str(agj))
    print("You feel breeze")
    return agi,agj
elif (world[i][j]==-1):
    agi,agj=i,j
    print("\nNow the agent is at "+str(agi)+" "+str(agj))
    print("You met wumpus")
    return -5,-5
else:
    #if world environment was empty agi,agj=i,j
    print("\nNow the agent is at "+str(agi)+" "+str(agj))
    return agi,agj
```

def checkinp(agi,agj):

"Function for checking input going in forward direction to get gold"

if(agi==0 and agj==0):

```
        print("\nyou can go at      "+str(agi+1)+" "+str(agj))#can move upward
        print("you can go at      "+str(agi)+" "+str(agj+1)) #can move right
        agvi=int(input("\nEnter input for row => ")) agvj=int(input("Enter input for
        column => ")) if(agvi==agi+1 and agvj==agj or agvi==agi and
        agvj==agj+1):
            return agvi,agvj
```

else:

return -5

elif(agi==3 and agj==0):

```
        print("\nyou can go at      "+str(agi-1)+" "+str(agj))      #can go left
        print("you can go at      "+str(agi)+" "+str(agj+1)) #can go right
```

```

    agvi=int(input("\nEnter input for row => ")) agvj=int(input("Enter
input for column => ")) if(agvi==agi-1 and agvj==agj or agvi==agi and
agvj==agj+1):
        return agvi,agvj else:
            return -5
elif(agi==3 and agj==3):
    print("\nyou can go at      "+str(agi-1)+" "+str(agj))
#can go down print("you can go at "+str(agi)+" "+str(agj-1)) #can go
left
    agvi=int(input("\nEnter input for row => "))
    agvj=int(input("Enter input for column => ")) if(agvi==agi-1
and agvj==agj or agvi==agi and agvj==agj-1):
        return agvi,agvj else:
            return -5
elif(agi==0 and agj==3):
    print("\nyou can go at      "+str(agi+1)+" "+str(agj))#can go upward
    print("you can go at      "+str(agi)+" "+str(agj-1)) #can go left
    agvi=int(input("\nEnter input for row => "))
    agvj=int(input("Enter input for column => ")) if(agvi==agi+1
and agvj==agj or agvi==agi and agvj==agj-1):
        return agvi,agvj else:
            return -5,-5
elif(agi==1 and agj==0 or agi==2 and agj==0 or agi==3 and agj==0):
    print("\nyou can go at      "+str(agi+1)+" "+str(agj))#can go upward
    print("you can go at      "+str(agi)+" "+str(agj+1)) #can move right
    agvi=int(input("\nEnter input for row => "))
    agvj=int(input("Enter input for column => ")) if(agvi==agi+1
and agvj==agj or agvi==agi and agvj==agj+1):
        return agvi,agvj else:
            return -5,-5 elif(agi==0 and agj==3 or agi==1 and agj==3 or agi==2
and agj==3 or agi==3 and agj==3):
    print("you can go at      "+str(agi+1)+" "+str(agj)) #can go upward
    print("you can go at      "+str(agi)+" "+str(agj-1)) #can go left
    agvi=int(input("Enter input for row => "))
    agvj=int(input("Enter input for column => ")) if(agvi==agi+1
and agvj==agj or agvi==agi and agvj==agj-1):
        return agvi,agvj else:
            return -5,-5
elif(agi==3 and agj==1 or agi==3 and agj==2 or agi==3 and agj==3):
print("\nyou can go at      "+str(agi)+" "+str(agj+1)) #can go right
    print("you can go at      "+str(agi)+" "+str(agj-1)) #can go left
    print("you can go at      "+str(agi-1)+" "+str(agj)) #can move
downward
    agvi=int(input("\nEnter input for row => ")) agvj=int(input("Enter
input for column => "))
    if(agvi==agi and agvj==agj+1 or agvi==agi and agvj==agj-1 or agvi==agi-1
and agvj==agj):

```

```

        return agvi,agvj
    else:
        return -5,-5
else:
    print("\nyou can go at      "+str(agi)+"      "+str(agj+1)) #can go right
    print("you can go at      "+str(agi)+"      "+str(agj-1))  #can go left
    print("you can go at      "+str(agi+1)+"      "+str(agj))    #can move upward
    agvi=int(input("\nEnter input for row => ")) agvj=int(input("Enter
input for column => "))
    if(agvi==agi and agvj==agj+1 or agvi==agi and agvj==agj-1 or agvi==agi+1
        and agvj==agj):
        return agvi,agvj
    else:
        return -5,-5

```

def checkinpreverse(agi,agj):

"Function for checking input going in reverse direction to get back to original position" if(agi==0 and agj==3):

```

    print("you can go at  "+str(agi)+"      "+str(agj-1))          #can go left
    agvi=int(input("\nEnter input for row => "))
    agvj=int(input("Enter input for column => "))
    if(agvi==agi and agvj==agj-1): return agvi,agvj
    else:
        return -5,-5

```

elif(agi==0 and agj==2 or agi==0 and agj==1):

```

    print("you can go at  "+str(agi)+"      "+str(agj+1))          #can go right
    print("you can go at  "+str(agi)+"      "+str(agj-1))          #can go left
    agvi=int(input("\nEnter input for row => "))
    agvj=int(input("Enter input for column => ")) if(agvi==agi
and agvj==agj-1 or agvi==agi and agvj==agj+1 ):
        return agvi,agvj else:
            return -5,-5

```

elif(agi==1 and agj==0 or agi==2 and agj==0):

```

    print("\nyou can go at      "+str(agi-1)+" "+str(agj))
#can go downward print("you can go at      "+str(agi)+"      "+str(agj+1)) #can
move right agvi=int(input("\nEnter input for row => "))
agvj=int(input("Enter input for column => ")) if(agvi==agi-1
and agvj==agj or agvi==agi and agvj==agj+1):
    return agvi,agvj else:
        return -5,-5

```

elif(agi==1 and agj==3 or agi==2 and agj==3):

```

    print("you can go at      "+str(agi-1)+" "+str(agj))          #can go downward
    print("you can go at      "+str(agi)+"      "+str(agj-1)) #can go left
    agvi=int(input("Enter input for row => "))
    agvj=int(input("Enter input for column => ")) if(agvi==agi-1
and agvj==agj or agvi==agi and agvj==agj-1):

```

```

        return agvi,agvj
    else:
        return -5,-5
    else:
        print("\nyou can go at      "+str(agi-1)+" "+str(agj))#can go downward
        print("you can go at      "+str(agi)+"    "+str(agj-1))  #can go left
        print("you can go at    "+str(agi)+"    "+str(agj+1))      #can go right
        agvi=int(input("\nEnter input for row => ")) agvj=int(input("Enter input for
        column => ")) if(agvi==agi-1 and agvj==agj or agvi==agi and agvj==agj-1 or
        agvi==agi and agvj==agj+1):
            return agvi,agvj
        else:
            return -5,-5

world=[[0,5,7,5],
        [9,0,8,0],
        [-1,6,7,8],
        [9,0,8,7]]                                #declaration of a world

agi,agj=0,0 #initial agent position print("\n\ninitially agent is at
"+str(agi)+",""+str(agj)) print("\nyou can go at
        "+str(agi+1)+"""+str(agj)) print("you can go at      "+str(agi)+
        "+str(agj+1))

agvi=int(input("Enter input for row => ")) agvj=int(input("Enter input for column => "))
#taking row and column values if(agvi==1 and agvj==0 or agvi==0 and agvj==1):
    agi,agj=learnagent(world,agvi,agvj)            #if input valid calling learn agent
function
else:
    print("Not valid")

while(agi>=0):
    agvi,agvj=checkinp(agi,agj) if(agvi!=-
    5 and agvj!=-5):
        agi,agj=learnagent(world,agvi,agvj) else:
            print("\nNot valid")
if(agi===-5): print("\nGame over Sorry try next
time!!!") else:
    print("\nYou have unlocked next level move back to your initial position") #acquired
    gold

    agi,agj=2,1
    #implementation of reverse logic

    while(agi>=0):

```

```
agvi,agvj=checkinpreverse(agi,agj)
if(agvi==0 and agvj==0): agi,agj=-4,-
4
elif(agvi!=-5 and agvj!=-5):
    agi,agj=learnagent(world,agvi,agvj) else:
    print("\nNot valid")
```

```
if(agi==5):
    print("\nYou were really close but unfortunately you failed!!! Try next time")
else:
    print("\nHurray You won!!!! Three cheers.")
```


PROTOTYPE ENVIRONMENT:

START
↓

0	5	7	5
9	0	8	0
-1	6	7	8
9	0	8	7

#9=stench
#8=glitter
#7=pit
#6=gold
#5=breeze
#-1=wumpus

OUTPUT:

Case 1: Found the gold

```
input
initially agent is at 0,0
you can go at 1 0
you can go at 0 1
Enter input for row => 0
Enter input for column => 1

Now the agent is at 0,1
You feel breeze

you can go at 0 2
you can go at 0 0
you can go at 1 1

Enter input for row => 1
Enter input for column => 1

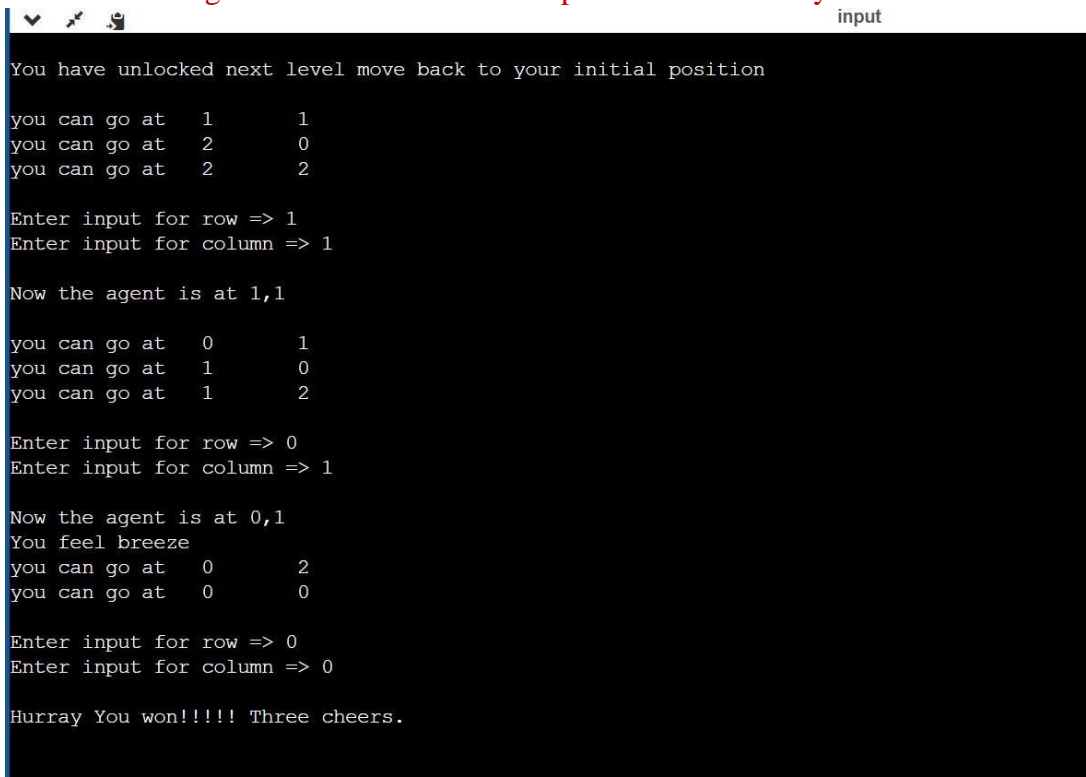
Now the agent is at 1,1

you can go at 1 2
you can go at 1 0
you can go at 2 1

Enter input for row => 2
Enter input for column => 1

Now the agent is at 2,1
You found gold
```

Case 2: Found gold and reached the initial position successfully



```
input
You have unlocked next level move back to your initial position

you can go at 1 1
you can go at 2 0
you can go at 2 2

Enter input for row => 1
Enter input for column => 1

Now the agent is at 1,1

you can go at 0 1
you can go at 1 0
you can go at 1 2

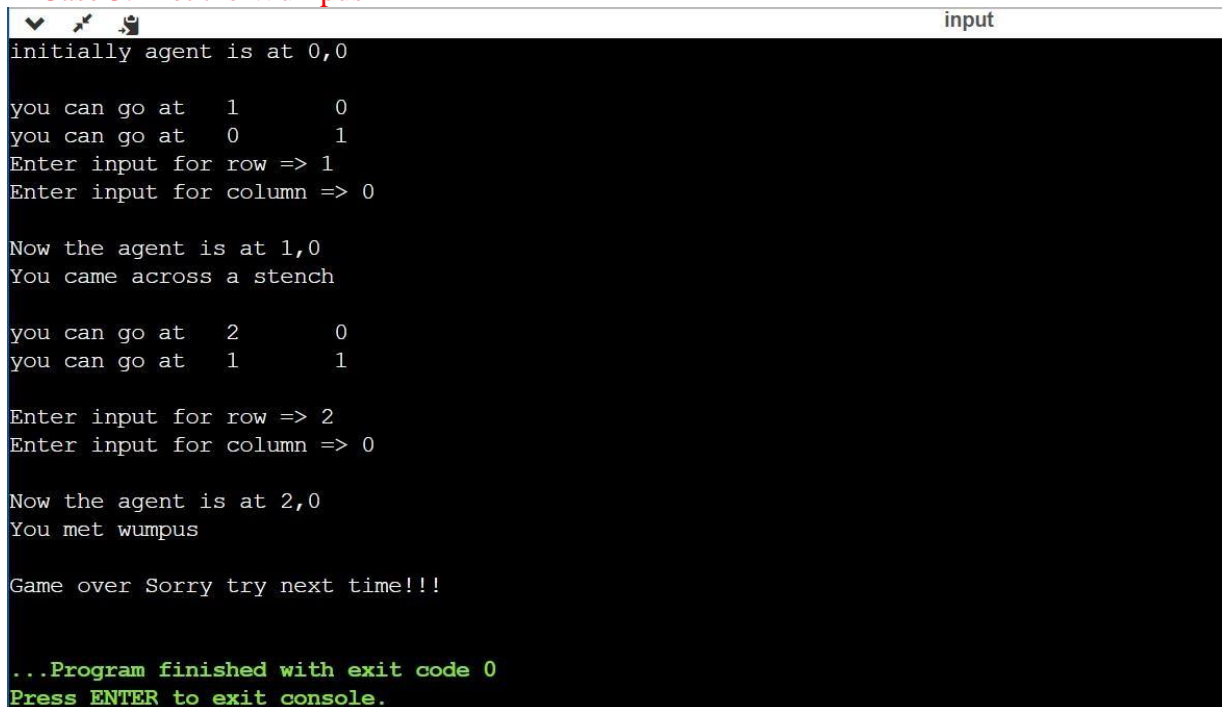
Enter input for row => 0
Enter input for column => 1

Now the agent is at 0,1
You feel breeze
you can go at 0 2
you can go at 0 0

Enter input for row => 0
Enter input for column => 0

Hurray You won!!!! Three cheers.
```

Case 3: Met the Wumpus



```
input
initially agent is at 0,0

you can go at 1 0
you can go at 0 1
Enter input for row => 1
Enter input for column => 0

Now the agent is at 1,0
You came across a stench

you can go at 2 0
you can go at 1 1

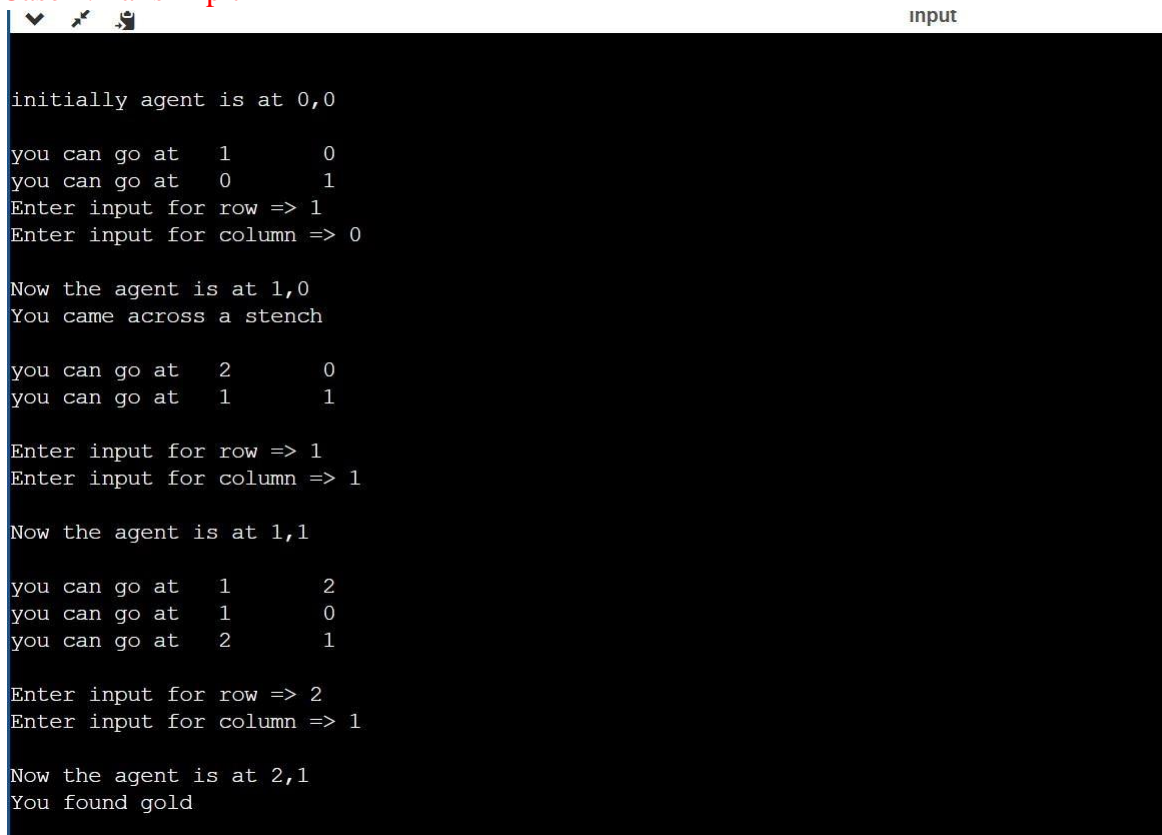
Enter input for row => 2
Enter input for column => 0

Now the agent is at 2,0
You met wumpus

Game over Sorry try next time!!!

...Program finished with exit code 0
Press ENTER to exit console.
```

Case 4: Falls in pit



```
initially agent is at 0,0

you can go at   1       0
you can go at   0       1
Enter input for row => 1
Enter input for column => 0

Now the agent is at 1,0
You came across a stench

you can go at   2       0
you can go at   1       1

Enter input for row => 1
Enter input for column => 1

Now the agent is at 1,1

you can go at   1       2
you can go at   1       0
you can go at   2       1

Enter input for row => 2
Enter input for column => 1

Now the agent is at 2,1
You found gold
```

```
You have unlocked next level move back to your initial position

you can go at   1       1
you can go at   2       0
you can go at   2       2

Enter input for row => 2
Enter input for column => 2

Now the agent is at 2,2
You came across a pit

You were really close but unfortunately you failed!!! Try next time

...Program finished with exit code 0
Press ENTER to exit console.
```