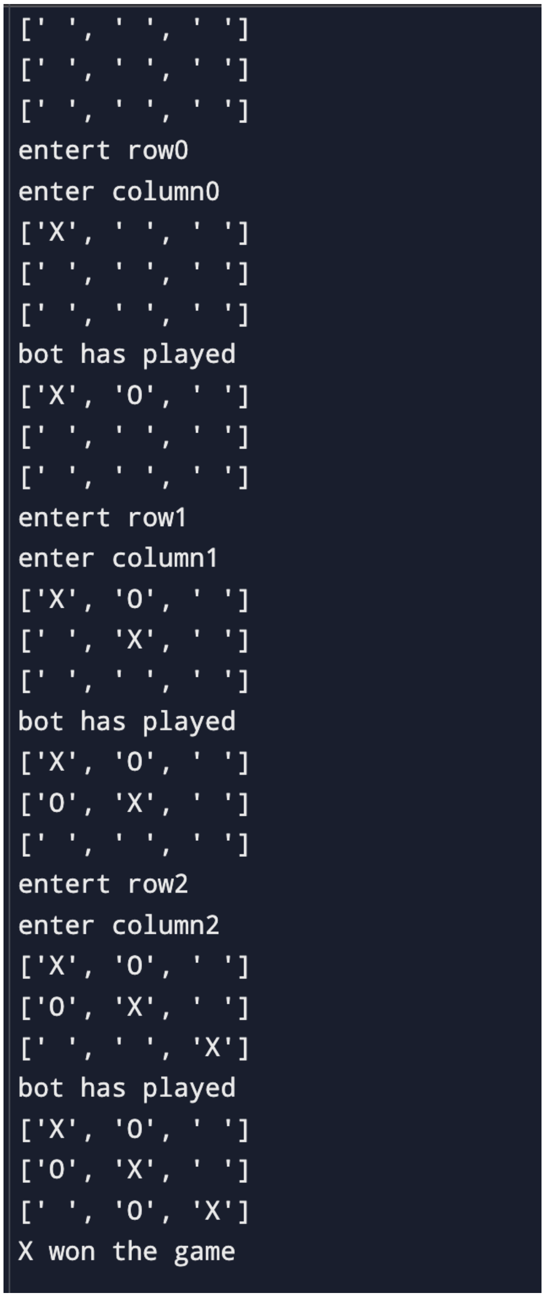
1-10-24 Tic Tac Toe game

Code:

```
import random
board = [[" " for j in range(3)] for i in range(3)]
c=0
def game_over():
    for i in range(3):
        if board[0][i]==board[1][i] and board[2][i]==board[1][i] and board[0][i]!=' ': #columns
            return True,board[0][i]
    for i in range(3):
        if board[i][0]==board[i][1] and board[i][1]==board[i][2] and board[i][0]!=' ':   #rows
            return True,board[i][0]
    if board[0][0]==board[1][1] and board[1][1]==board[2][2] and board[0][0]!=' ':
            return True,board[0][0]
    if board[0][2]==board[1][1] and board[1][1]==board[2][0] and board[1][1]!=' ':
            return True,board[1][1]
    return False,'X'
def num_gen():
    a=random.randint(0,2)
    b=random.randint(0,2)
    return a,b
def take_cell():
    r=int(input("entert row"))
    k=int(input("enter column"))
    return r,k

def display_board():
    for i in range(3):
        print(board[i])
display_board()
while 1:
    i,j=take_cell()
    while board[i][j]!=' ':
        if i>2 or j>2 or i<0 or j<0:
            print("invalid cell")
        print("the cell is already occupied")
        i,j = take_cell()
    board[i][j]='X'
    c+=1
    display_board()
    l,m=num_gen()
    while board[l][m]!=' ':
        l,m = num_gen()
```

```
board[l][m]='O'
c+=1
print("bot has played")
display_board()
e,g= game_over()
if e:
    print(f"{g} won the game")
    break
if c==9:
    print("game is draw")
    break
```



```
[' ', ' ', ' ']
[' ', ' ', ' ']
[' ', ' ', ' ']
entert row0
enter column0
['X', ' ', ' ']
[' ', ' ', ' ']
[' ', ' ', ' ']
bot has played
['X', 'O', ' ']
[' ', ' ', ' ']
[' ', ' ', ' ']
entert row1
enter column1
['X', 'O', ' ']
[' ', 'X', ' ']
[' ', ' ', ' ']
bot has played
['X', 'O', ' ']
['O', 'X', ' ']
[' ', ' ', ' ']
entert row2
enter column2
['X', 'O', ' ']
['O', 'X', ' ']
[' ', ' ', 'X']
bot has played
['X', 'O', ' ']
['O', 'X', ' ']
[' ', 'O', 'X']
X won the game
```

Vacuum cleaner world

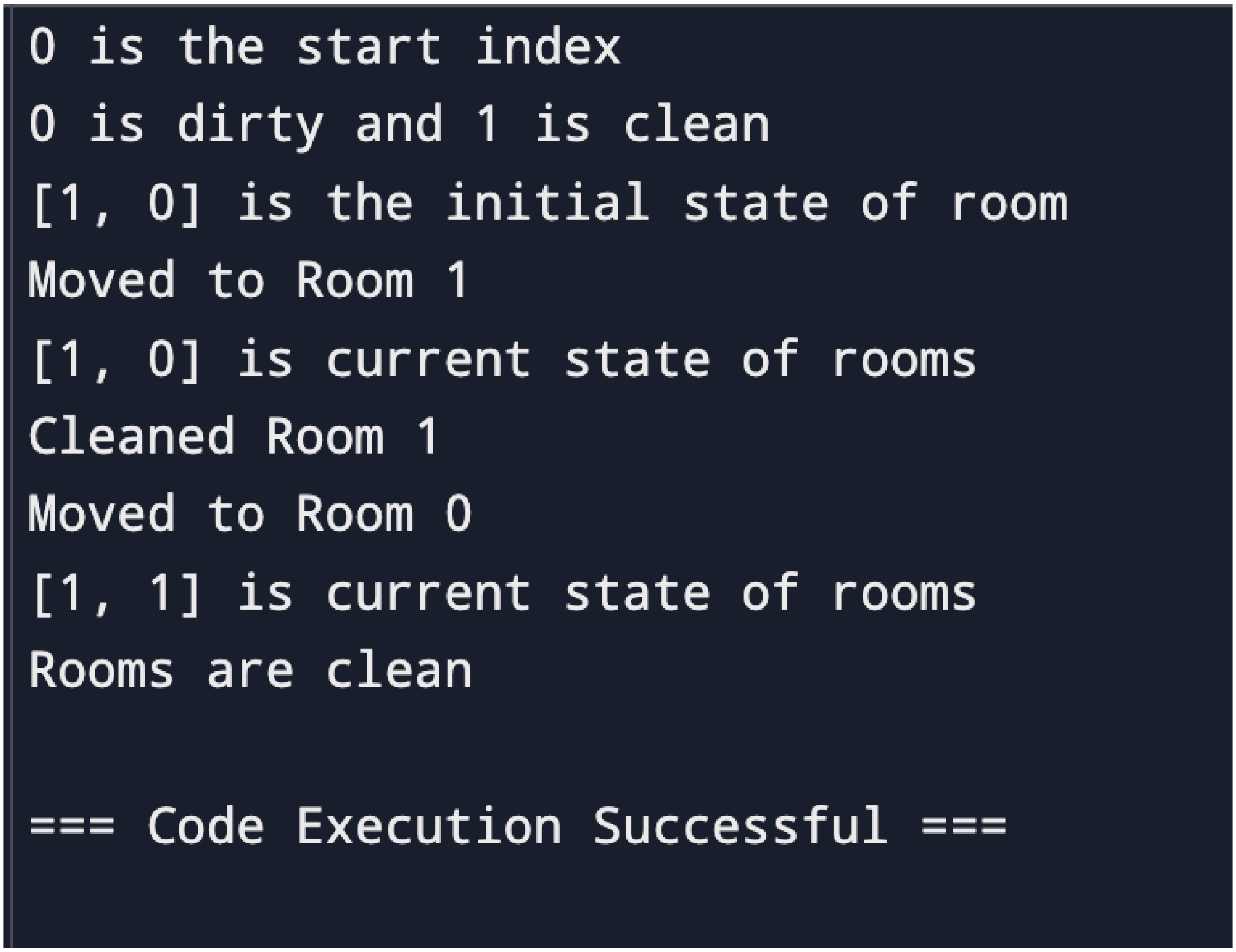
Algorithm:

```
function VacuumCleanerAgent(environment):
    position = (0, 0)
    cleaned_cells_count = 0

    while True:
        if environment[position] is dirty:
            clean(environment[position])
            cleaned_cells_count += 1
            print("Cleaned position:", position)

        next_position = findNextDirty(environment)
        if next_position exists:
            position = next_position
        else:
            print("No more dirty cells found. Cleaning complete.")
            break

function findNextDirty(environment):
    for each cell in environment:
        if cell is dirty:
            return cell's position
    return None
```

Code:

```
import random
l=[random.choice([0,1]),random.choice([0,1])]
def check(i):
    if l[i]==0:
        l[i]=1
        print(f"Cleaned Room {i}")
    print(f"Moved to Room {(i+1)%2}")
    return (i+1)%2
i=random.choice([0,1])
print(f"{i} is the start index")
print("0 is dirty and 1 is clean")
print(f"{l} is the initial state of room")
while sum(l)!=2:
    i=check(i)
    if l[(i+1)%2]==1:
        l[(i+1)%2]=random.choice([0,1])
        if l[(i+1)%2]==0:
            print(f"Room {(i+1)%2} got dirty")
    print(f"{l} is current state of rooms")
print("Rooms are clean")
```

Output:



```
0 is the start index
0 is dirty and 1 is clean
[1, 0] is the initial state of room
Moved to Room 1
[1, 0] is current state of rooms
Cleaned Room 1
Moved to Room 0
[1, 1] is current state of rooms
Rooms are clean

=== Code Execution Successful ===
```