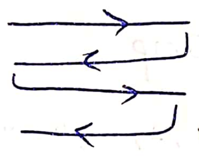
22-12-23 Vacuum cleaner Agent:

Code: (One room).

```
def clean(floor):
    i, j, row, col = 0, 0, len(floor), len(floar[0])
    for i in range(row):
        if(i%2==0):
            for j in range(col):
                if(floor[i][j]==1):
                    print_F(floor, i, j)
                    floor[i][j] = 0
                print_F(floor, i, j)
        else:
            for j in range(col-1, -1, -1):
                if(floor[i][j]==1):
                    print_F(floor, i, j)
                    floor[i][j] = 0
            }
            print_F(floor, i, j)
```

// The vacuum cleaner cannot move diagonally or jump to positions directly. So for all even rows, the vacuum cleaner ~~the~~ moves from left to right and for all odd rows, it moves from right to left, as shown below



If ~~the roo~~ a ~~do~~ grid in a room is dirty (i.e floor[i][j] Then print the ~~room~~ grid and clean it by setting it to 0.

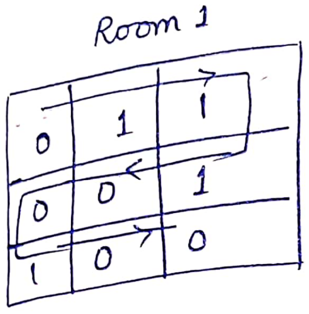
```
def print_F(floor, ~~col,~~ row, col):
    print("The floor matrix is as below:")
    for r in range(len(floor)):
        for c in range(len(floor[r])):
            if r==row and c==col:
                print(f" >{floor[r][c]}< ", end="")
            else:
                print(f" {floor[r][c]} ", end="")
        print(end='\n')
    print(end='\n')


def main():
    floor = []
    m = int(input("Enter the no. of rows:"))
    n = int(input("Enter the no. of columns:"))
    print("Enter the clean status of each cell (1-dirty, 0-clean)")
    for i in range(m):
        f = list(map(int, input().split(" ")))
        floor.append(f)
    print()
    clean(floor).
```

Two rooms:

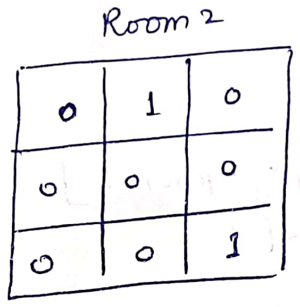
Logic:

In the main function, take input for 2 rooms.

~~If room1~~
Initially, start from room 1 and inspect every grid.
If room 1 is clean i.e [room1] = 0 , move to room 2 and clean it.



0	1	1
0	0	1
1	0	0



0	1	0
0	0	0
0	0	1

Room 1 → dirty → so clean it.
Check if room 1 is completely clean (i.e all grids are 0)
if clean, check if room 2 is already clean, else
move to room 2.

If room 2 → dirty
Clean it by calling 'clean' function and ~~move to room 1~~
If completely clean, return True and check if room 1 is clean, else move to room 1, else
~~and move to~~

if room 1 also is completely clean, return true and exit.

Room 1
Status: clean $\xrightarrow{\text{move to}}$ Room 2.

(Instead of grid, ~~take~~ implement this as just one grid in a room)

2 Code for 2 rooms:

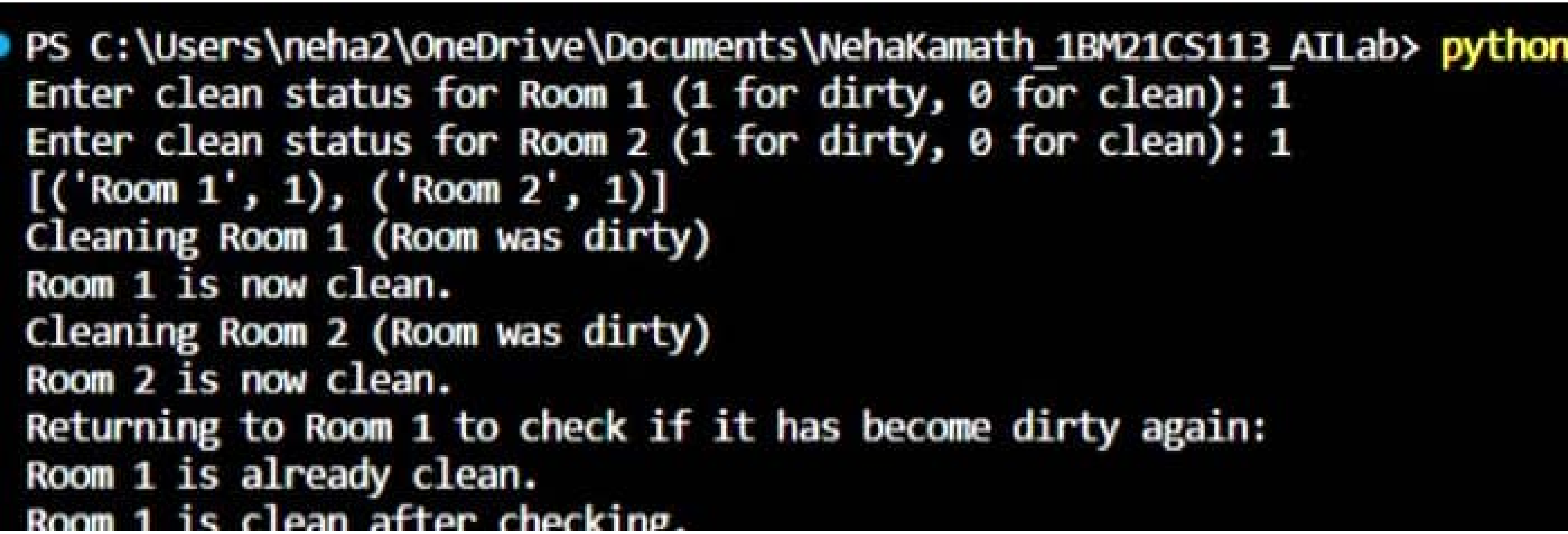
```
def clean_room(room_name, is_dirty):
if is_dirty:
  print(f"Cleaning {room_name} (Room was dirty)")
  print(f"{room_name} is now clean.")
  return 0

else:
  print(f"{room-clean} is already clean.")
  return 0

def main():
rooms = ["Room 1", "Room2"]
room_statuses = []

 for room in rooms:
   status = int(input(f"Enter the clean status for {room} (1 for dirty,
                                                         0 for clean):

 room_statuses.append((room, status))
   i,                     enumerate
 for (room, status) in ^(room_statuses):
                (room,
room_statuses[i] = clean_room(room, status))

print(f"Returning to {Room[0]} to check if it ~~is~~ has become
                                   dirty again.")
                (rooms[0],
room_statuses[0]=clean_room(rooms[0], room_statuses[0][1]))
                                          room_statuses[0][1]
print(f"{rooms[0]} is {'dirty' if ~~status~~ else 'clean'} after
if __name__ == "__main__":                              checking.")
   main()
```

P
Proceed
8

Output:

```
Enter clean status for room 1 : 1
Enter clean status for room 2 : 0
[('Room 1', 1), ('Room 2', 0)]
Cleaning Room 1 (Room was dirty)
Room 1 is now clean.
Room 2 is already clean.
Returning to room 1 to check if it has become dirty again:
Room 1 is already clean.
Room 1 is clean after checking.
```

[illegible]
22/12/23



```
PS C:\Users\neha2\OneDrive\Documents\NehaKamath_1BM21CS113_AILab> python
Enter clean status for Room 1 (1 for dirty, 0 for clean): 1
Enter clean status for Room 2 (1 for dirty, 0 for clean): 1
[('Room 1', 1), ('Room 2', 1)]
Cleaning Room 1 (Room was dirty)
Room 1 is now clean.
Cleaning Room 2 (Room was dirty)
Room 2 is now clean.
Returning to Room 1 to check if it has become dirty again:
Room 1 is already clean.
Room 1 is clean after checking.
```

Scanned with OKEN Scanner