

## TWO ROOMS SETUP

```
def vacuum_cleaner():
    # Input the state of rooms A and B
    state_A = int(input("Enter state of A (0 for clean, 1 for dirty): "))
    state_B = int(input("Enter state of B (0 for clean, 1 for dirty): "))
    location = input("Enter location (A or B): ").upper()


    cost = 0
    rooms = {'A': state_A, 'B': state_B}

    # Function to clean a room if dirty
    def clean_room(room):
        nonlocal cost
        if rooms[room] == 1:
            print(f"Cleaned {room}.")
            rooms[room] = 0
            cost += 1
        else:
            print(f"{room} is clean.")

    # Start cleaning based on location
    if location == 'A':
        clean_room('A')
        print("Moving vacuum right")
        clean_room('B')
    elif location == 'B':
        clean_room('B')
        print("Moving vacuum left")
        clean_room('A')
    else:
        print("Invalid starting location!")

    print(f"Cost: {cost}")
    print(rooms)






if __name__ == "__main__":
    vacuum_cleaner()
```



VACCUM SHREYAS SINHA 1BM23CS321.ipynb

File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all ▼




12s

```
nonlocal cost
if rooms[room] == 1:
    print(f"Cleaned {room}.")
    rooms[room] = 0
    cost += 1
else:
    print(f"{room} is clean.")

# Start cleaning based on location
if location == 'A':
    clean_room('A')
    print("Moving vacuum right")
    clean_room('B')
elif location == 'B':
    clean_room('B')
    print("Moving vacuum left")
    clean_room('A')
else:
    print("Invalid starting location!")

print(f"Cost: {cost}")
print(rooms)

if __name__ == "__main__":
    vacuum_cleaner()
```



Enter state of A (0 for clean, 1 for dirty): 0  
Enter state of B (0 for clean, 1 for dirty): 1  
Enter location (A or B): A  
A is clean.  
Moving vacuum right  
Cleaned B.  
Cost: 1  
{'A': 0, 'B': 0}

## FOUR ROOMS SETUP:

```
def vacuum_cleaner():
    # Taking user input for the state of each room
    state_A = int(input("Enter state of A (0 for clean, 1 for dirty): "))
    state_B = int(input("Enter state of B (0 for clean, 1 for dirty): "))
    state_C = int(input("Enter state of C (0 for clean, 1 for dirty): "))
```

```
state_D = int(input("Enter state of D (0 for clean, 1 for dirty): "))
location = input("Enter location (A, B, C, or D): ").upper()
```

```
cost = 0
rooms = {'A': state_A, 'B': state_B, 'C': state_C, 'D': state_D}
```

```
# Function to clean a room and update the cost
```

```
def clean_room(room):
```

```
    nonlocal cost
```

```
    if rooms[room] == 1:
```

```
        print(f"Cleaned {room}.")
```

```
        rooms[room] = 0
```

```
        cost += 1
```

```
    else:
```

```
        print(f"{room} is clean.")
```

```
if location == 'A':
```

```
    clean_room('A')
```

```
    print("Moving vacuum right")
```

```
    clean_room('B')
```

```
    print("Moving vacuum down")
```

```
    clean_room('D')
```

```
    print("Moving vacuum left")
```

```
    clean_room('C')
```

```
elif location == 'B':
```

```
    clean_room('B')
```

```
    print("Moving vacuum left")
```

```
    clean_room('A')
```

```
    print("Moving vacuum down")
```

```
    clean_room('D')
```

```
    print("Moving vacuum right")
```

```
    clean_room('C')
```

```
elif location == 'C':
```

```
    clean_room('C')
```

```
    print("Moving vacuum right")
```

```
    clean_room('D')
```

```
    print("Moving vacuum up")
```

```
    clean_room('B')
```

```
    print("Moving vacuum left")
```

```
    clean_room('A')
```

```
elif location == 'D':
```

```
    clean_room('D')
```

```
    print("Moving vacuum up")
```

```
    clean_room('B')
```

```
    print("Moving vacuum right")
```

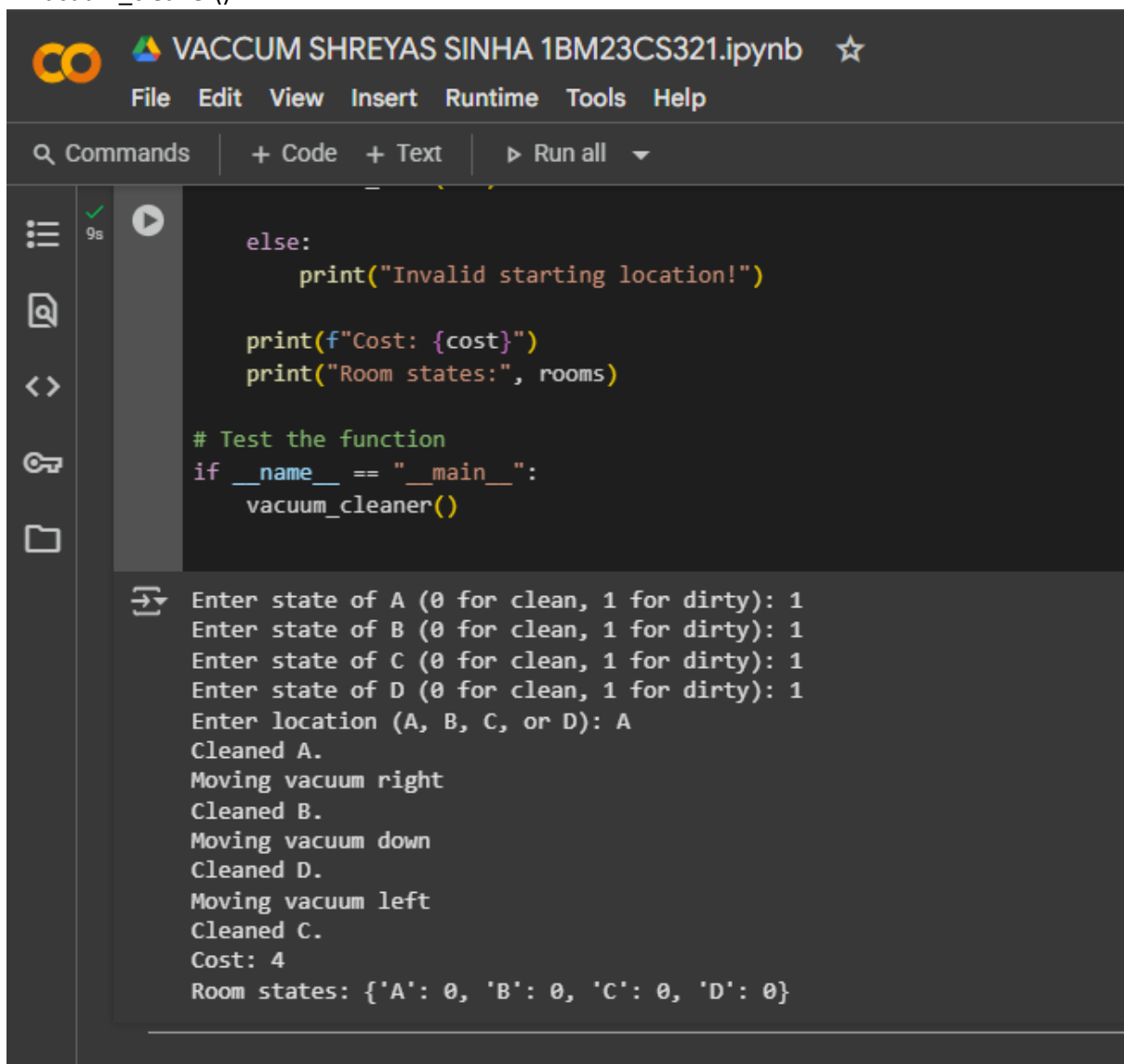
```
clean_room('C')
print("Moving vacuum left")
clean_room('A')
```

```
else:
    print("Invalid starting location!")
```

```
print(f"Cost: {cost}")
print("Room states:", rooms)
```

```
# Test the function
```

```
if __name__ == "__main__":
    vacuum_cleaner()
```



The screenshot shows a Jupyter Notebook titled "VACCUM SHREYAS SINHA 1BM23CS321.ipynb". The interface includes a top bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help" menus. Below the menu bar is a toolbar with "Commands", "+ Code", "+ Text", and "Run all" buttons. The left sidebar contains icons for a table of contents, search, expand/collapse, key, and file explorer. The main area displays the Python code from the previous blocks. The code cell is marked with a green checkmark and a "9s" execution time. Below the code, the output of the program is shown, including prompts for room states, location input, and the final cost and room states.

```
else:
    print("Invalid starting location!")

    print(f"Cost: {cost}")
    print("Room states:", rooms)

# Test the function
if __name__ == "__main__":
    vacuum_cleaner()
```

Enter state of A (0 for clean, 1 for dirty): 1  
Enter state of B (0 for clean, 1 for dirty): 1  
Enter state of C (0 for clean, 1 for dirty): 1  
Enter state of D (0 for clean, 1 for dirty): 1  
Enter location (A, B, C, or D): A  
Cleaned A.  
Moving vacuum right  
Cleaned B.  
Moving vacuum down  
Cleaned D.  
Moving vacuum left  
Cleaned C.  
Cost: 4  
Room states: {'A': 0, 'B': 0, 'C': 0, 'D': 0}

## LAB-II, VACUUM CLEANER AGENT

# Algorithm  $\Rightarrow$

① Two Room Setup:

(i) Start

(ii) Implement initial state with dust and vacuum cleaner with 2 room setup.

(iii) If vacuum cleaner is in room 'A', and dust is present suck it.

(iv) After cleaning A, ask user to move to room 'B' and clean the dust in B.

(v) Then move the cleaner back to A.

(vi) End

~~② 4-room Setup:~~

② 4 - room Setup:

(i) Initialize

(ii) Start at R1 and move through rooms in a specific path.

(iii) If R1 is not clean, clean the dust and then ask user for the next room.

(iv) Move to either R2 or R3 and then clean the dust in that room.

(v) Then repeat the process so that all the rooms are clean and the objective is achieved.

(vi) End the process.

25/8/25