# ROBOT\_VACUUM

## Task 1

1. **Khởi tạo môi trường:**

**.** Biểu diễn phòng bằng một ma trận 2 chiều (mảng NumPy), trong đó:

**+** 0 → ô sạch

**+** 1 → ô bẩn

**.** Một phần các ô được chọn ngẫu nhiên là bẩn, theo xác suất dirt\_prob.

1. **Vị trí của agent:**

**.** Robot được đặt tại vị trí ngẫu nhiên trong lưới.

**.** Vị trí được theo dõi xuyên suốt quá trình mô phỏng.

1. **Cảm biến và hành động:**

**.** Ở mỗi bước, môi trường cung cấp cho agent dữ liệu cảm biến gồm:

**+** Trạng thái ô hiện tại (clean hoặc dirty),

**+** Vị trí của robot,

**+** Kích thước môi trường.

**.** Agent chọn một hành động trong:

**['UP', 'DOWN', 'LEFT', 'RIGHT', 'SUCK']**

**.** Môi trường phản ứng:

**+** Nếu "SUCK" → làm sạch ô hiện tại (đặt = 0).

**+** Nếu di chuyển → cập nhật vị trí robot (kiểm tra tường để tránh ra ngoài lưới).

1. **Điều kiện dừng:**

**.** Mô phỏng tiếp tục cho đến khi:

**+** Tất cả ô đều sạch (env == 0), hoặc

**+** Số bước đạt max\_steps (hết năng lượng).

1. **Thước đo hiệu suất (Performance Measure):**

**.** Sau khi mô phỏng kết thúc:

**performance = (số ô sạch / tổng số ô) × 100%**

**.** Đồng thời theo dõi:

**+** Số hành động đã thực hiện (actions\_taken),

**+** Số bước mô phỏng (steps).

1. **Hiển thị (Debugging và Visualization):**

**.** Khi verbose=True, mỗi bước hiển thị trạng thái môi trường:

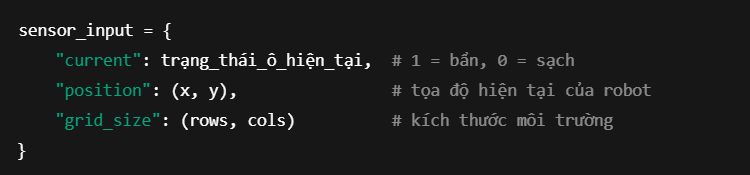
**+** 1: ô bẩn

**+** 0: ô sạch

## Task 2

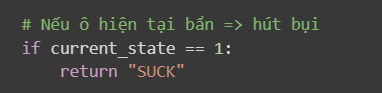
1. **Nhận cảm biến từ môi trường**

Môi trường gửi cho agent một bộ thông tin cảm biến dưới dạng dictionary:

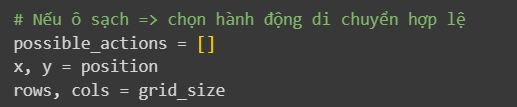


1. **Phân tích trạng thái ô hiện tại**

**.** Nếu **ô hiện tại đang bẩn (**current == 1**)**  
→ Agent **chọn hành động “SUCK”** để làm sạch ô đó.



**.** Nếu **ô hiện tại đã sạch (**current == 0**)**  
→ Agent **chọn một hướng di chuyển ngẫu nhiên**



1. **Xác định hành động di chuyển hợp lệ:**

Dựa vào vị trí (x, y) và kích thước (rows, cols):

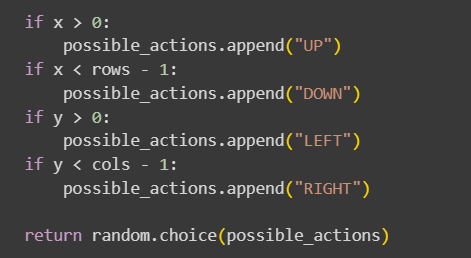
**.** Nếu x > 0 → có thể đi UP,

**.** Nếu x < rows - 1 → có thể đi DOWN,

**.** Nếu y > 0 → có thể đi LEFT,

**.** Nếu y < cols - 1 → có thể đi RIGHT.

Agent chỉ chọn ngẫu nhiên trong các hành động hợp lệ này.



1. **Trả về hành động**

Sau khi ra quyết định, agent trả về hành động cho môi trường:

Môi trường sẽ cập nhật trạng thái dựa trên hành động:

**+ "SUCK"** → đặt ô hiện tại thành sạch (0),

**+ "UP"/"DOWN"/"LEFT"/"RIGHT"** → di chuyển robot sang ô mới.

1. **Tiếp tục lặp lại**

Quy trình (nhận cảm biến → chọn hành động → cập nhật môi trường)  
được lặp lại cho đến khi:

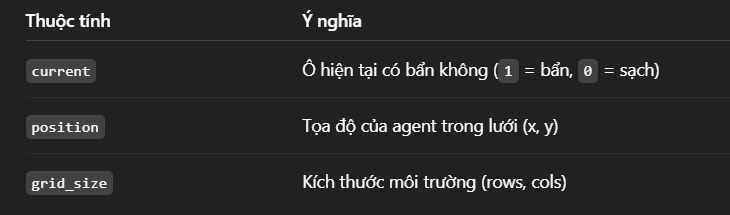
**+** Tất cả ô đều sạch, hoặc

**+** Robot hết năng lượng (đạt max\_steps).

## Task 3

1. **Nhận thông tin cảm biến (sensor\_input)**

Mỗi bước mô phỏng, môi trường sẽ cung cấp cho agent một “gói dữ liệu” gồm:



1. **Kiểm tra trạng thái ô hiện tại**

**.** Nếu current == 1 (tức ô đang bẩn) → agent ngay lập tức hút bụi ("SUCK").

**.** Đây là hành động phản xạ tức thì, luôn ưu tiên làm sạch trước khi di chuyển.

**3. Ghi nhớ trạng thái (State Memory)**

Nếu ô hiện tại sạch, agent sẽ:

**.** Ghi lại tọa độ hiện tại vào state["visited"] — tập hợp các ô đã đi qua,

**.** Giúp agent không bị lặp lại đường đi,

**.** Tránh bỏ sót vùng chưa làm sạch.

**4. Di chuyển theo mô hình Zig-Zag**

Agent sử dụng state["direction"] để xác định hướng di chuyển.

Khi hướng là "RIGHT":

**.** Nếu chưa tới biên phải, tiếp tục "RIGHT".

**.** Nếu chạm tường phải, đổi hướng thành "LEFT" và xuống một hàng ("DOWN").

Khi hướng là "LEFT":

**.** Nếu chưa tới biên trái, tiếp tục "LEFT".

**.** Nếu chạm tường trái, đổi hướng thành "RIGHT" và xuống một hàng ("DOWN").

**5. Khi đến góc cuối cùng**

**.** Khi agent đến ô dưới cùng bên phải (hoặc hết đường đi),

**.** Nó không còn ô mới để di chuyển,

**.** Agent sẽ hút bụi lần cuối ("SUCK") để đảm bảo sạch toàn bộ khu vực.

## Task 4

**1. Hàm evaluate\_agent(agent\_func, grid\_size, dirt\_prob, runs, max\_steps)**

Mục đích: đánh giá một agent bằng cách:

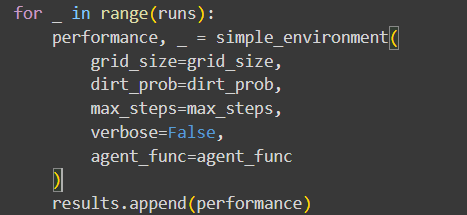
**.** Chạy mô phỏng runs lần (100 lần),

**.** Mỗi lần khởi tạo môi trường mới (với xác suất ô bẩn là dirt\_prob = 0.5),

**.** Agent hoạt động tối đa max\_steps bước.

Kết quả:

**.** Lấy hiệu suất trung bình (% ô sạch) của 100 lần chạy.



→ Gọi lại **simple\_environment()** để chạy mô phỏng.  
→ Mỗi kết quả (performance) là tỉ lệ ô sạch cuối cùng.



→ Trả về **hiệu suất trung bình**.

**2. Thiết lập các kích thước môi trường**



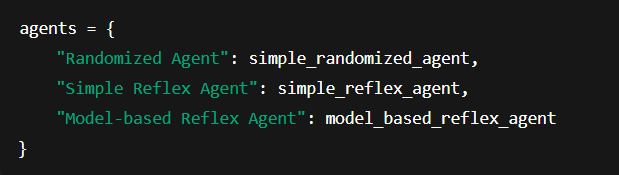
→ Ta sẽ kiểm thử 3 quy mô phòng:

**.** Nhỏ (5×5),

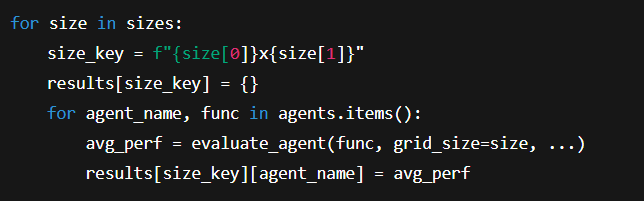
**.** Trung bình (10×10),

**.** Lớn (100×100).

1. **Định nghĩa các agent**

****

1. **Vòng lặp chính để đánh giá tất cả agent trên mọi kích thước**



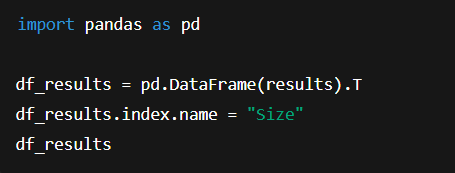
**.** Với mỗi kích thước (ví dụ 5×5),

**.** Chạy 3 agent, mỗi agent 100 lần,

**.** Tính hiệu suất trung bình,

**.** Lưu kết quả vào results.

1. **Tạo bảng kết quả (DataFrame)**



**Kết quả quan sát**

**Randomized Agent**

**+** Hoạt động dựa trên xác suất

**+** Không có trí nhớ hoặc chiến lược di chuyển

**+** Hiệu suất **dao động quanh 50%**, không tăng theo kích thước.

**Simple Reflex Agent**

**+** Biết **hút khi bẩn**, tránh tường

**+ Không nhớ vị trí**, có thể đi vòng lại hoặc bỏ sót

**+** Khi kích thước lớn (100×100), hiệu suất **giảm nhẹ** vì khó bao phủ toàn bộ.

**Model-Based Reflex Agent**

**+** Có **trí nhớ**, **chiến lược zig-zag**, **đảm bảo bao phủ toàn bộ**

**+** Hiệu suất rất cao (**≈ 95–100%**)

**+** Không bị ảnh hưởng bởi kích thước (chỉ tốn nhiều bước hơn).