

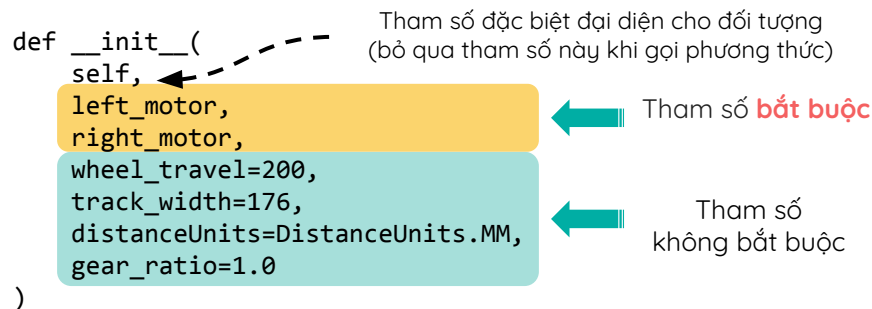
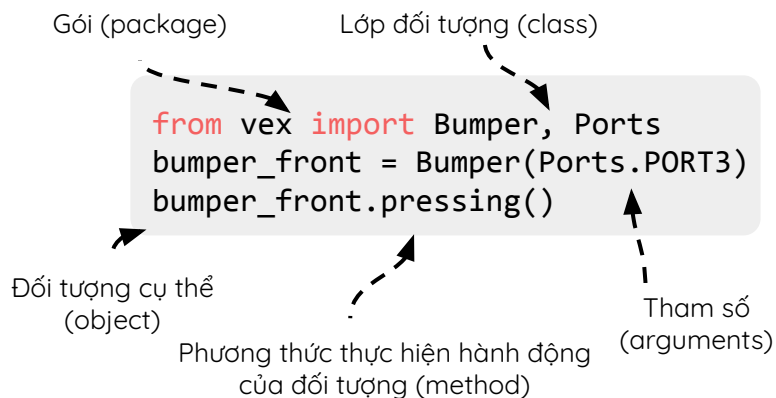


CS 201 - Huấn luyện kỹ năng **Python Cơ Bản 4**



Đây là một sản phẩm trí tuệ có bản quyền thuộc về STEAM for Vietnam. Các bên chỉ được sử dụng với mục đích học tập, nghiên cứu, và không được quyền sử dụng sản phẩm này nhằm mục đích thu lợi nhuận dù trực tiếp hay gián tiếp.

ÔN BÀI CŨ



Lưu ý quan trọng: Code Python trong Robot Mesh Studio chỉ nhận tham số theo đúng thứ tự





ÔN BÀI CŨ

Viết chương trình cho **robot di chuyển cách tường 20cm thì lùi về vị trí cũ.**

Gợi ý: Xác định khoảng cách với vật phía trước bằng cảm biến khoảng cách, tên lớp là Sonar thuộc gói “vex”



Distance Sensor (Cảm biến khoảng cách)

Đặc điểm:

Đo khoảng cách trong dải từ
50 mm đến 1m

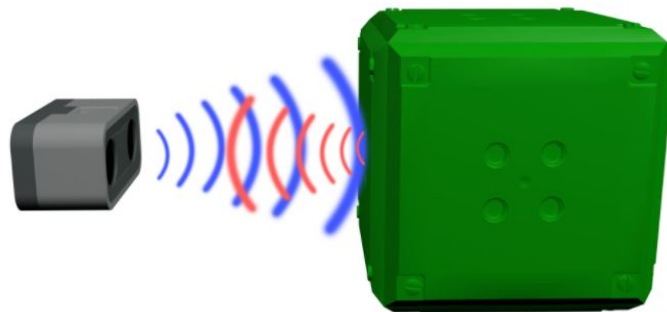
Ứng dụng:

Phát hiện vật cản không tiếp
xúc trong dải đo xác định



Nguyên tắc hoạt động:

Thu phát sóng siêu âm để phát hiện vật
cản



ÔN BÀI CŨ



Chúng ta hãy cùng làm bài tập nhé!

https://bit.ly/S4V_CS201_Practice_Sonar



BÀI 4: Các kiến thức sẽ học

- Scope(phạm vi) trong Python
- FSM(Finite state machine)(Trạng thái máy hữu hạn)
 - Tìm hiểu về FSM
 - Ứng dụng của FSM trong lập trình robotics



SCOPE TRONG PYTHON

Chúng ta hãy cùng xem một ví dụ nhé!

https://bit.ly/S4V_CS201_Scope_Python



PHẠM VI (SCOPE)

```
def set_name():  
    name = "CS201"  
    print(name)
```

set_name()



Chương trình in ra "CS201"

```
def get_name():  
    print(name)
```

get_name()

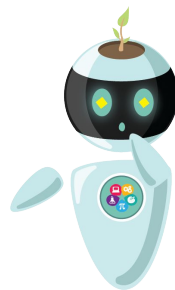


Exception: 0xea Name err

Chương trình báo lỗi ở hàm **get_name()**

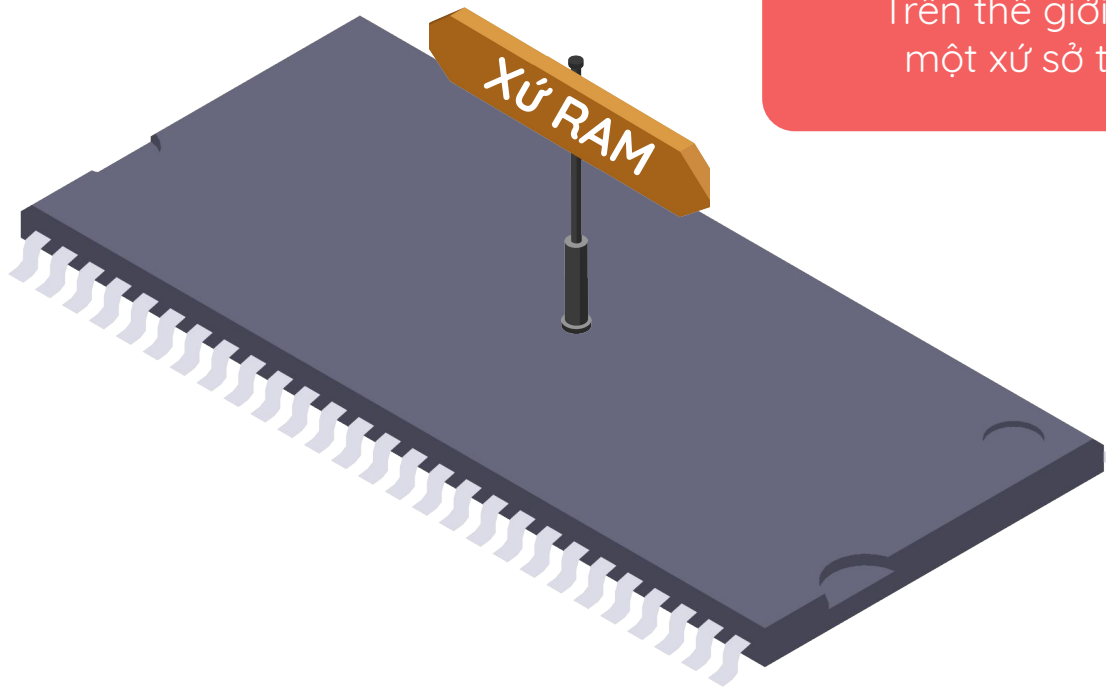


Tại sao vậy nhỉ?

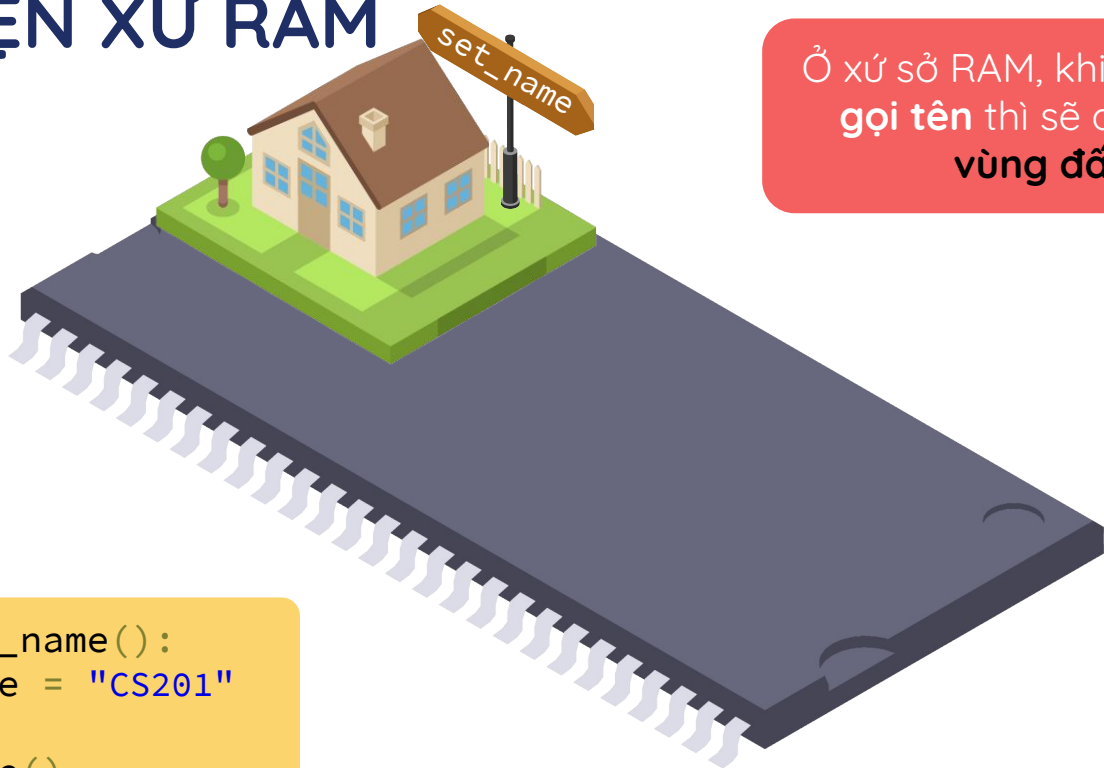


CHUYỆN XỬ RAM

Trên thế giới này tồn tại
một xử sở tên là **RAM**



CHUYỆN XỨ RAM



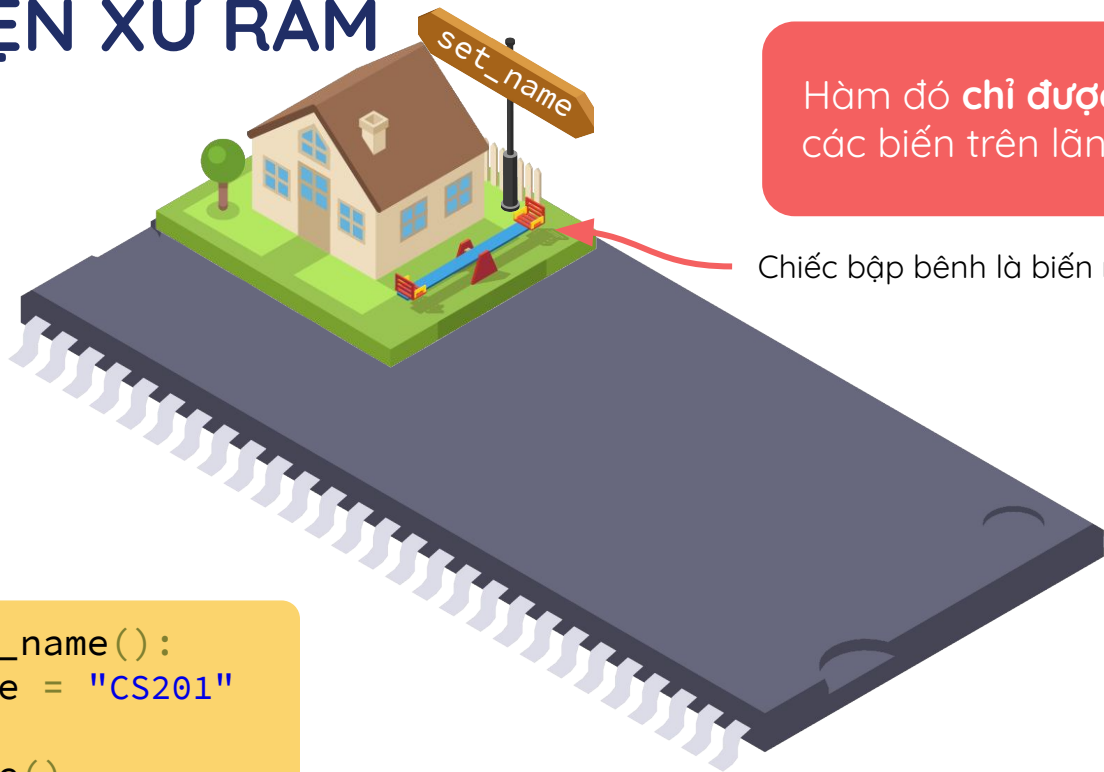
Ở xứ sở RAM, khi một hàm được **gọi tên** thì sẽ được ban một **vùng đất riêng**

```
def set_name():  
    name = "CS201"
```

```
set_name()
```



CHUYỆN XỨ RAM



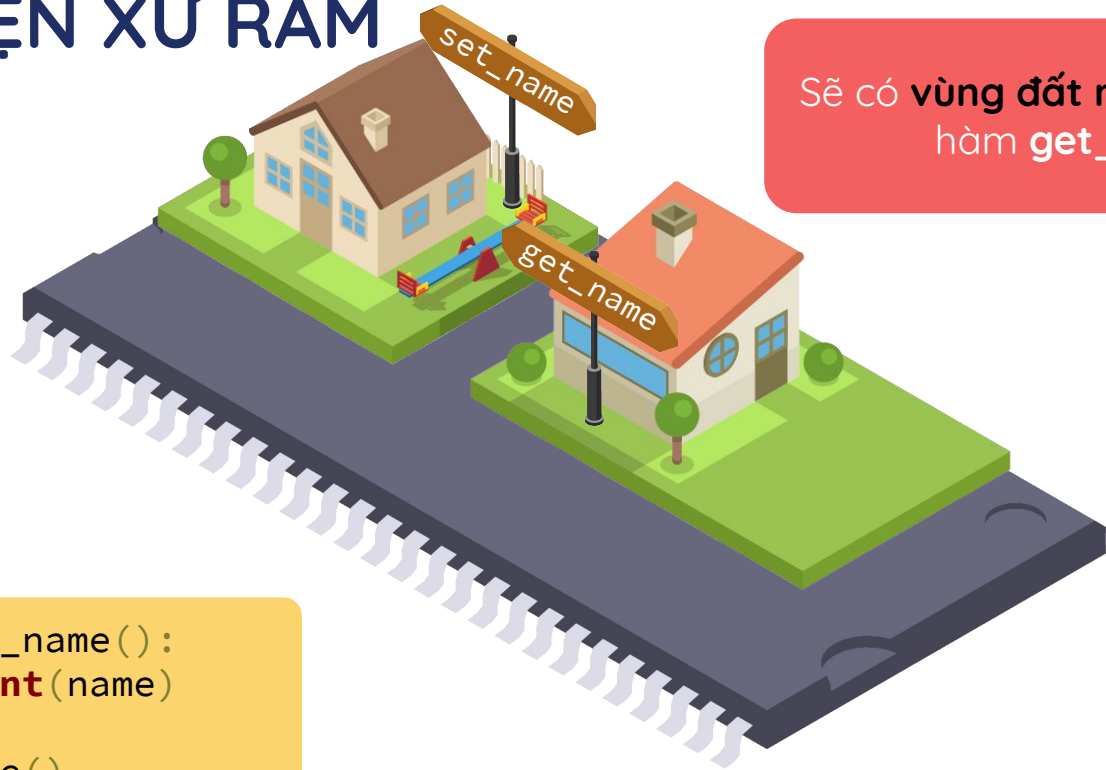
Hàm đó **chỉ được phép** khởi tạo các biến trên lãnh thổ của mình!

Chiếc bập bênh là biến **name**

```
def set_name():  
    name = "CS201"  
  
set_name()
```



CHUYỆN XỨ RAM



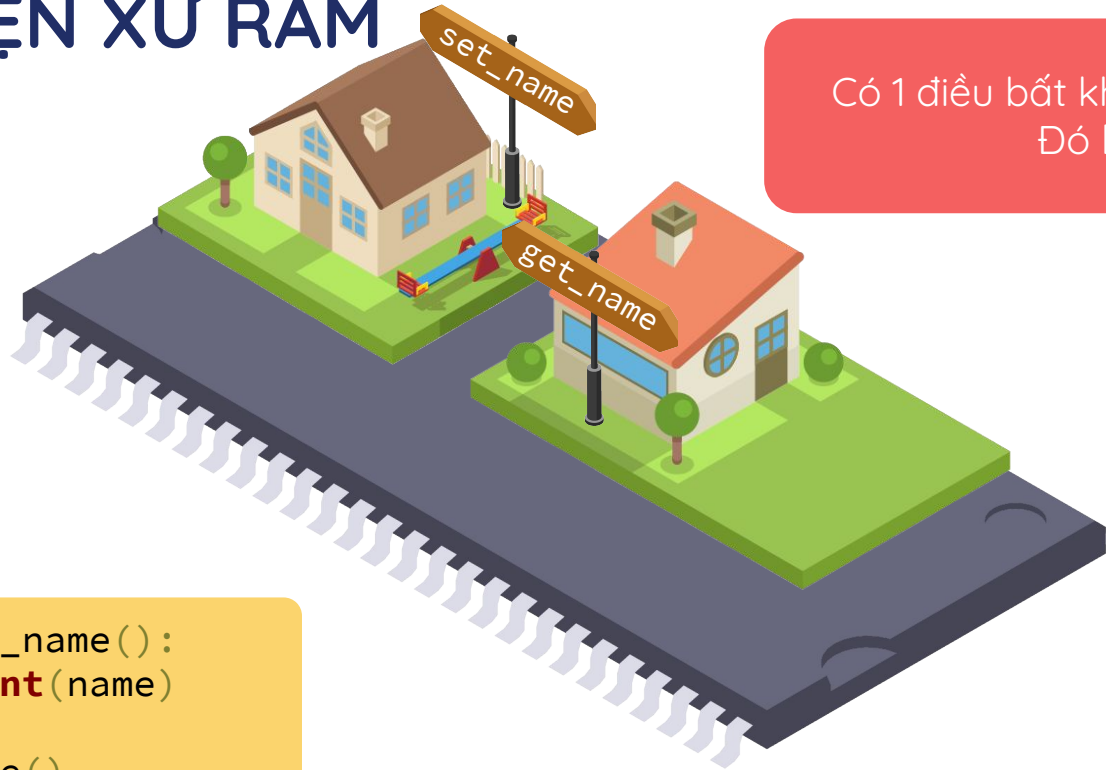
Sẽ có **vùng đất riêng** khi gọi tên
hàm **get_name()**

```
def get_name():  
    print(name)
```

get_name()



CHUYỆN XỨ RAM



Có 1 điều bất khả xâm phạm.
Đó là...

```
def get_name():  
    print(name)
```


```
get_name()
```



CHUYỆN XỨ RAM



Các lãnh thổ **không** được xâm phạm lẫn nhau



```
def get_name():  
    print(name)  
  
get_name()
```


CHUYỆN XỬ RAM



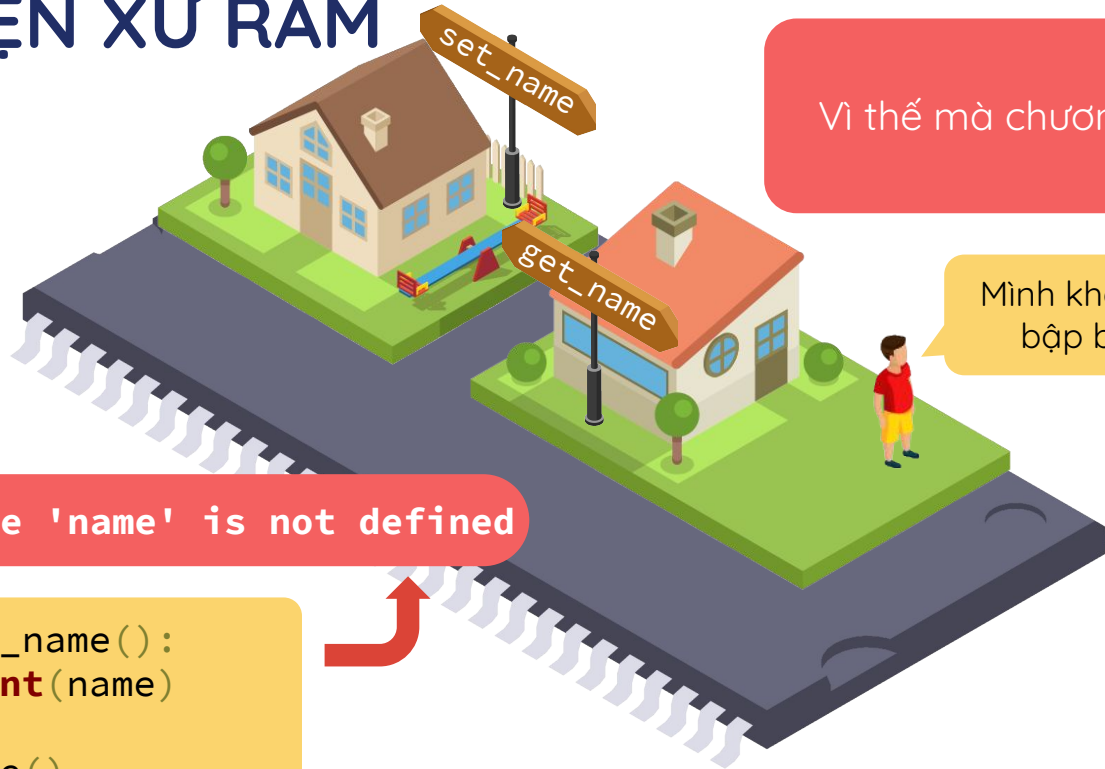
Do đó, khi hàm **get_name()** tìm biến **name** thì không thấy

Chiếc bập bênh
đâu nhỉ?

```
def get_name():  
    print(name)  
  
get_name()
```



CHUYỆN XỨ RAM



Vì thế mà chương trình báo **lỗi**

Mình không có
bập bênh

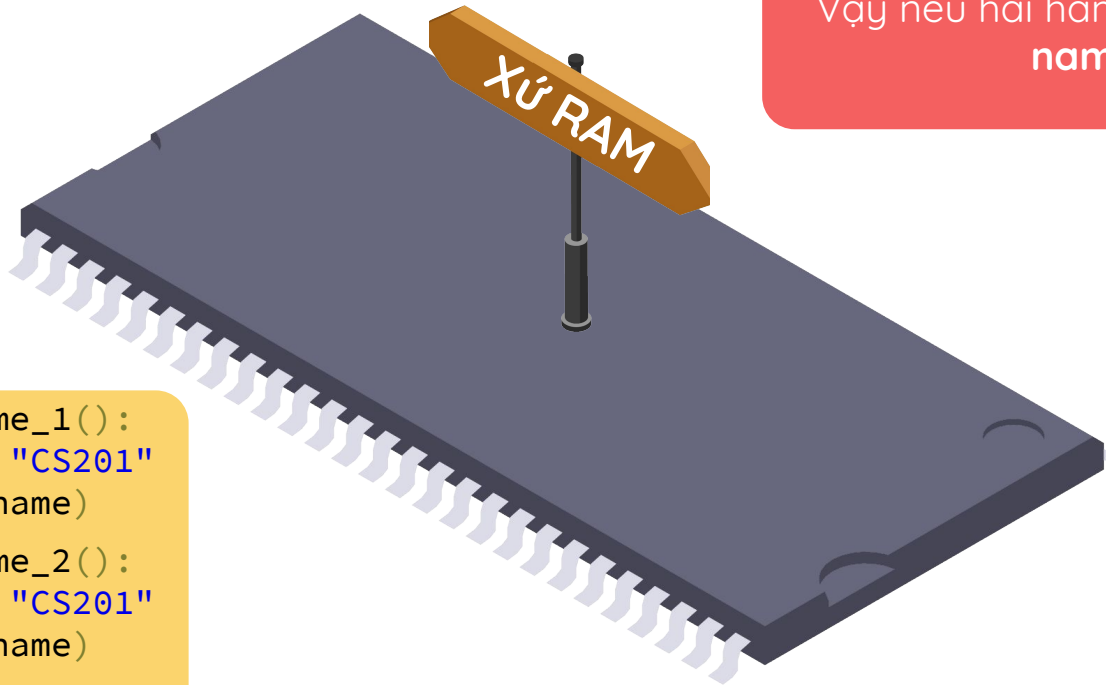
NameError: name 'name' is not defined

```
def get_name():  
    print(name)  
  
get_name()
```



CHUYỆN XỬ RAM

Vậy nếu hai hàm có cùng biến
name?



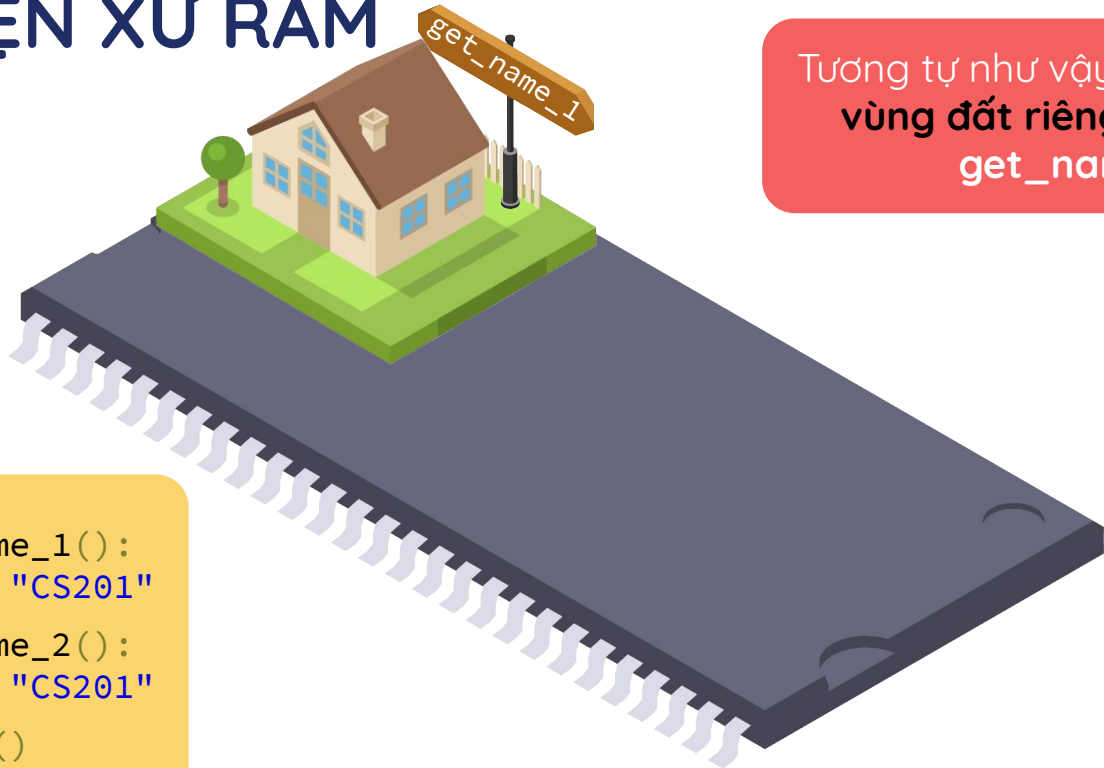
```
def get_name_1():  
    name = "CS201"  
    print(name)
```

```
def get_name_2():  
    name = "CS201"  
    print(name)
```

```
get_name_1()  
get_name_2()
```



CHUYỆN XỨ RAM

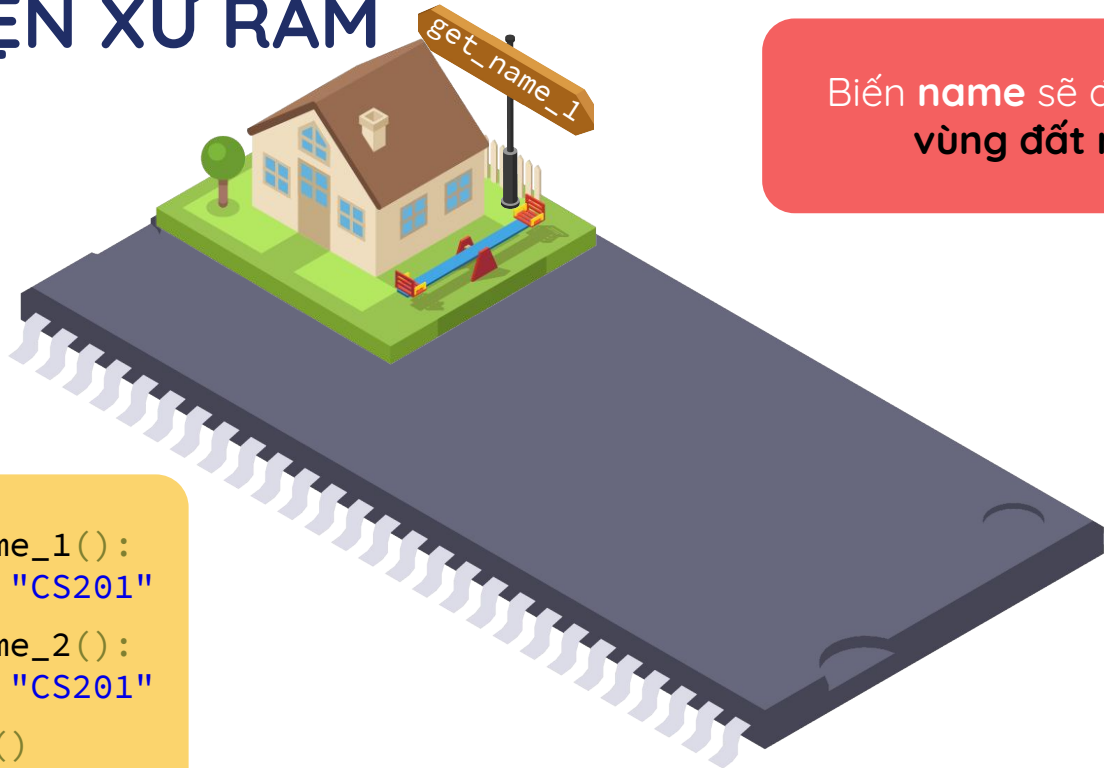


Tương tự như vậy, chúng ta sẽ có **vùng đất riêng** khi gọi hàm `get_name_1()`

```
def get_name_1():  
    name = "CS201"  
  
def get_name_2():  
    name = "CS201"  
  
get_name_1()  
get_name_2()
```



CHUYỆN XỨ RAM

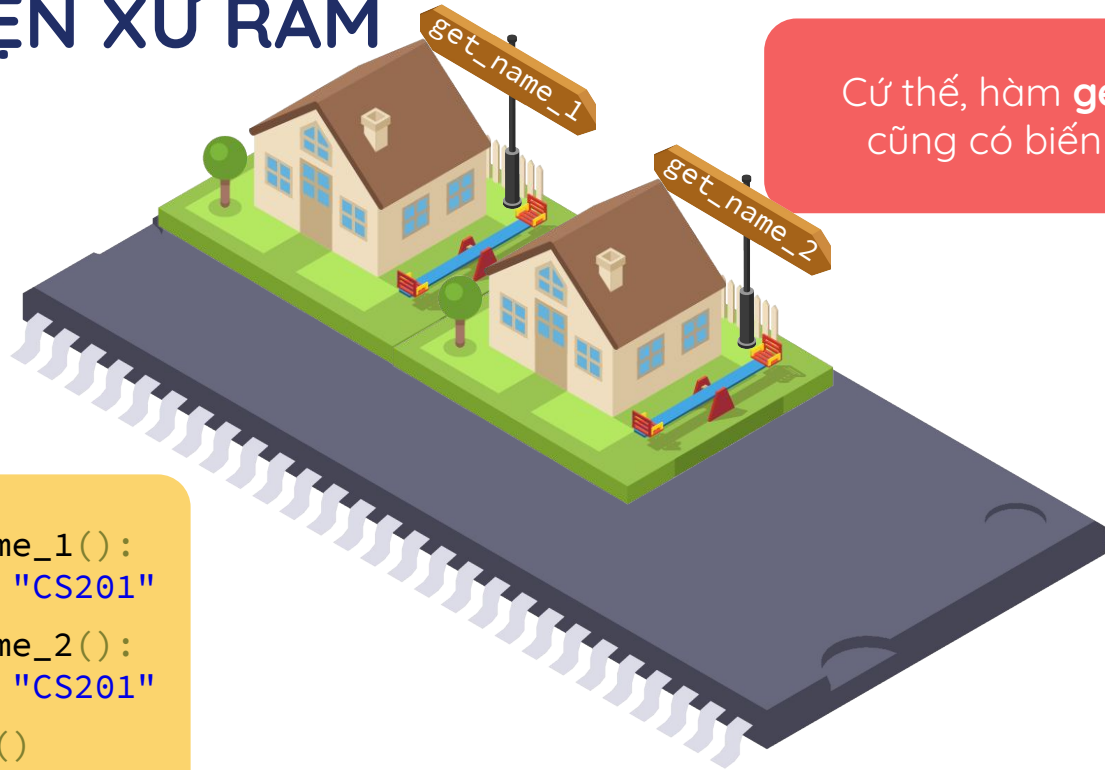


Biến **name** sẽ được tạo trong vùng đất riêng này

```
def get_name_1():  
    name = "CS201"  
  
def get_name_2():  
    name = "CS201"  
  
get_name_1()  
get_name_2()
```



CHUYỆN XỨ RAM

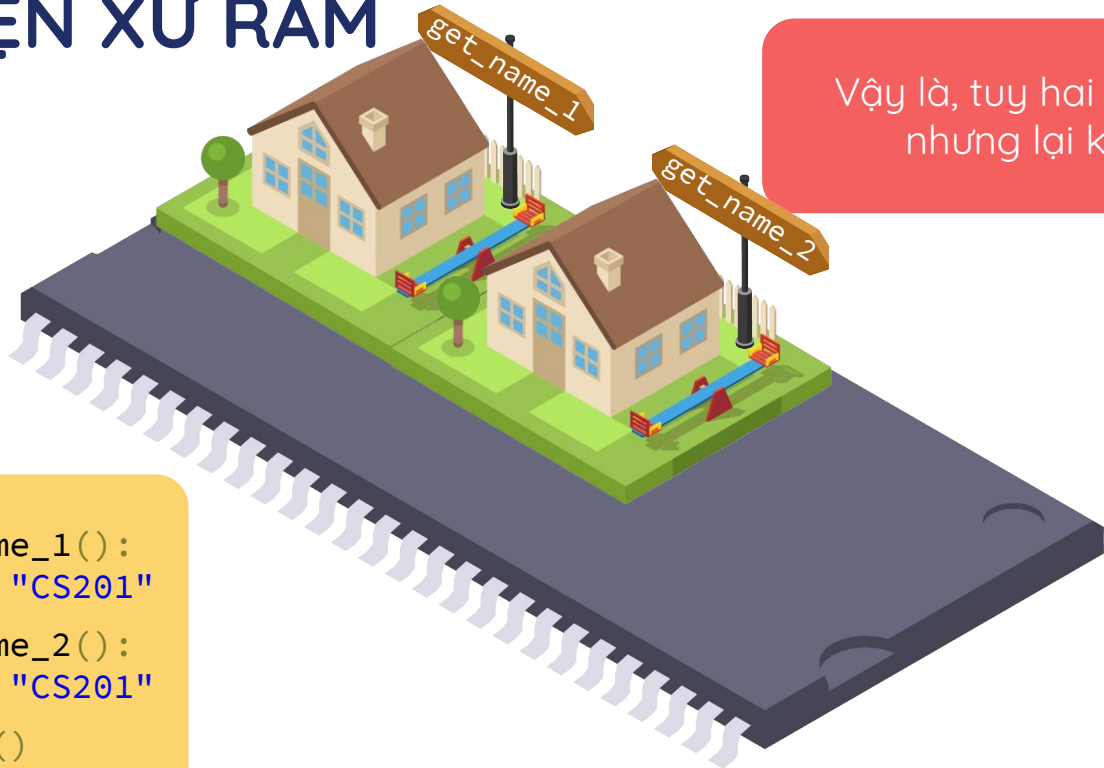


Cứ thế, hàm **get_name_2()**
cũng có biến **name** riêng

```
def get_name_1():  
    name = "CS201"  
  
def get_name_2():  
    name = "CS201"  
  
get_name_1()  
get_name_2()
```



CHUYỆN XỨ RAM



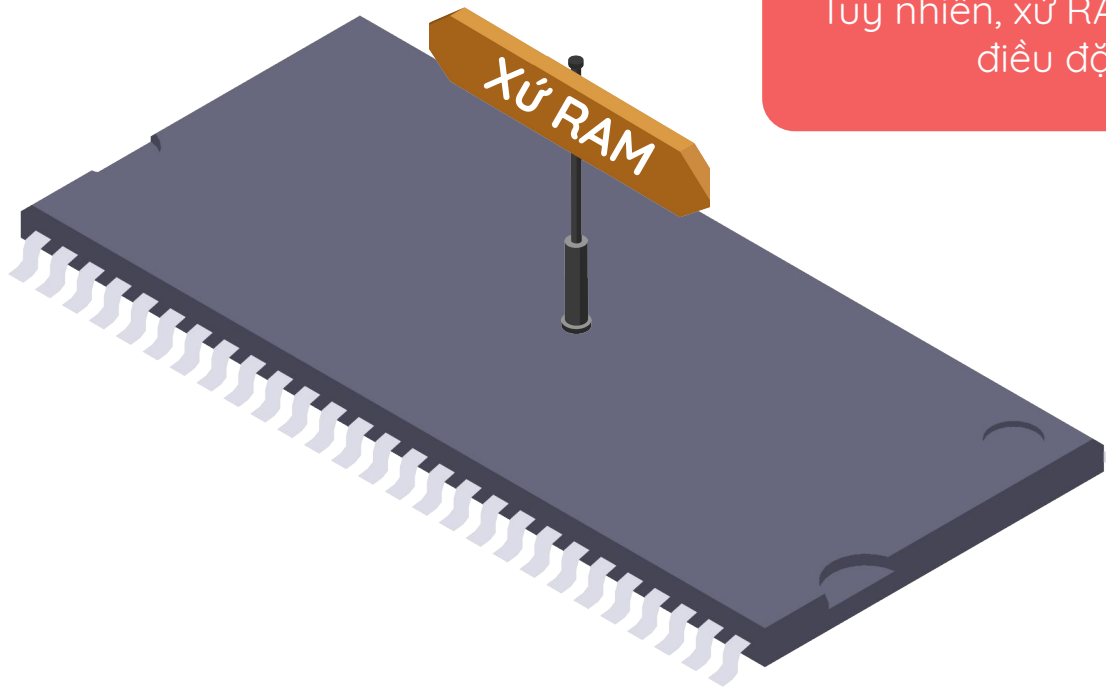
Vậy là, tuy hai biến cùng tên
nhưng lại khác nhau!

```
def get_name_1():  
    name = "CS201"  
  
def get_name_2():  
    name = "CS201"  
  
get_name_1()  
get_name_2()
```



CHUYỆN XỬ RAM

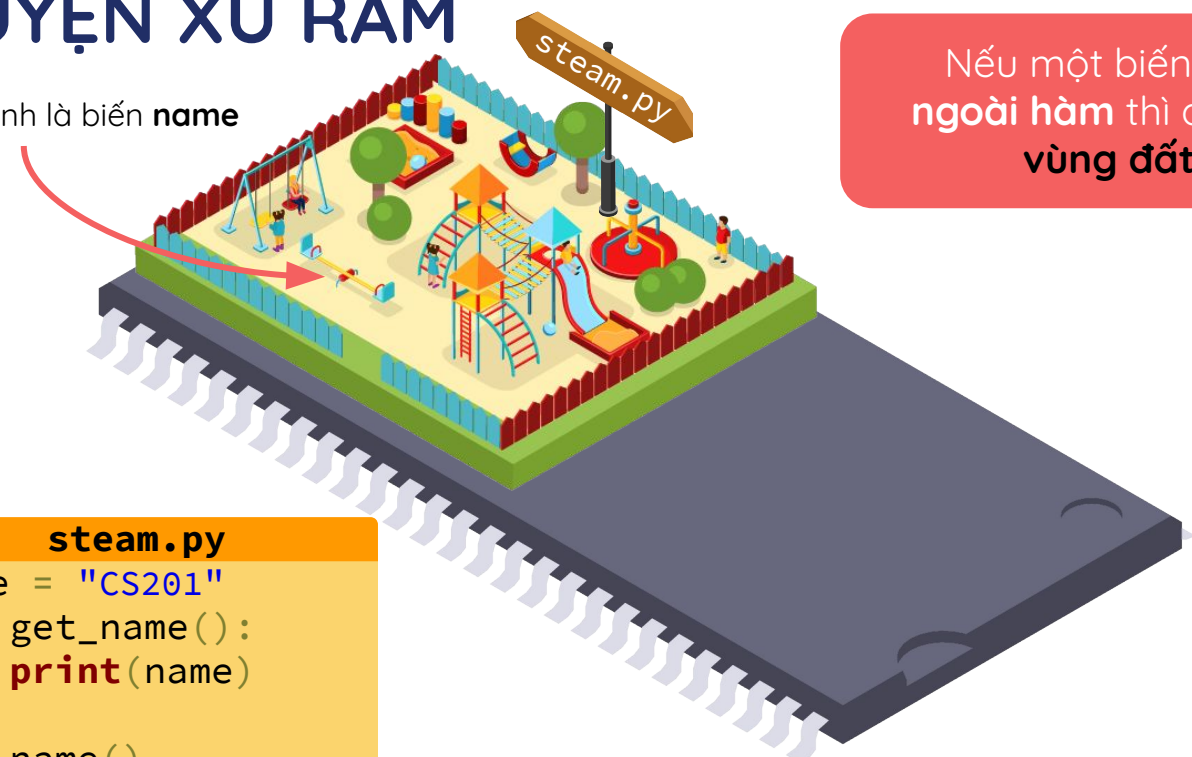
Tuy nhiên, xử RAM vẫn có một điều đặc biệt



CHUYỆN XỨ RAM

Chiếc bập bênh là biến **name**

Nếu một biến được tạo ở
ngoài hàm thì được đưa vào
vùng đất chung



```
steam.py  
name = "CS201"  
def get_name():  
    print(name)  
  
get_name()
```



CHUYỆN XỨ RAM



Như cũ, khi gọi tên **get_name()** sẽ có **vùng đất riêng** được cấp

```
steam.py  
name = "CS201"  
def get_name():  
    print(name)
```

get_name()



CHUYỆN XỨ RAM



Khi hàm `get_name()` tìm biến **name** trong vùng của mình **trước** nhưng không thấy thì...

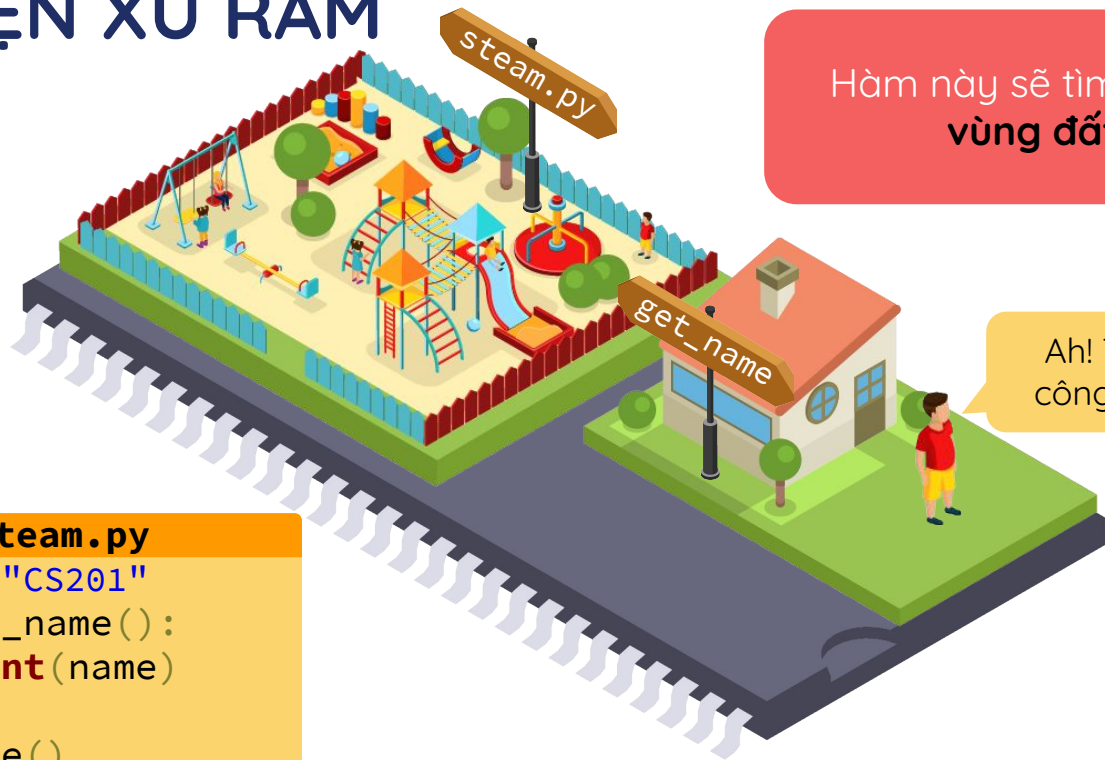
Mình không có chiếc bật bênh

```
steam.py
name = "CS201"
def get_name():
    print(name)

get_name()
```



CHUYỆN XỨ RAM



Hàm này sẽ tìm biến **name** ở vùng đất chung

Ah! Thử tìm ở công viên nhé!

```
steam.py
name = "CS201"
def get_name():
    print(name)

get_name()
```



CHUYỆN XỨ RAM

Ồn giời!
Cậu đây rồi

Khi tìm thấy biến **name**, chương trình sẽ in ra mà không bị lỗi

```
steam.py
name = "CS201"
def get_name():
    print(name)

get_name()
```



PHẠM VI (SCOPE)

Vùng đất chung được gọi là
phạm vi toàn cục (global scope)

Biến trong vùng này gọi là
biến toàn cục (global variable)



Vùng đất riêng được gọi là
phạm vi cục bộ (local scope)

Biến trong vùng này được gọi là
biến cục bộ (local variable)



PHẠM VI (SCOPE)

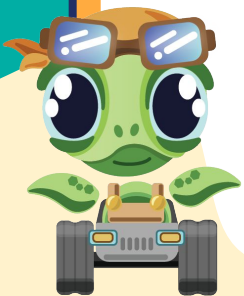
Vậy tại sao không tạo tất cả biến ở **global scope**?



PHẠM VI (SCOPE)

Vì chúng ta không thể có các biến cùng tên.

Do đó, khi cần lưu tên của nhiều người, chúng ta phải có
name_1, name_2, name_3, name_4...



Thực hành:

Biến name dùng trong hàm **print()** là biến nào?

```
1 name = "A"  
2 def get_name():  
3     name = "B"  
4     print(name)
```

- a. Biến của dòng 1
- b. Biến của dòng 3
- c. Biến của cả dòng 1 và 3
- d. Bị lỗi



Thực hành:

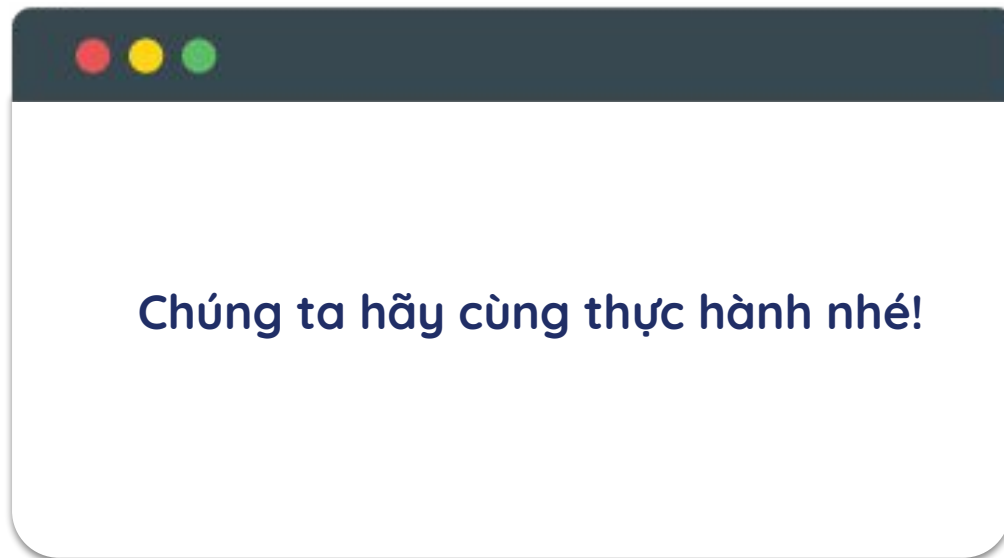
Biến name dùng trong hàm **print()** là biến nào?

```
1 name = "A"  
2 def get_name():  
3     name = "B"  
4     print(name)
```

- a. Biến của dòng 1
- b. Biến của dòng 3**
- c. Biến của cả dòng 1 và 3
- d. Bị lỗi

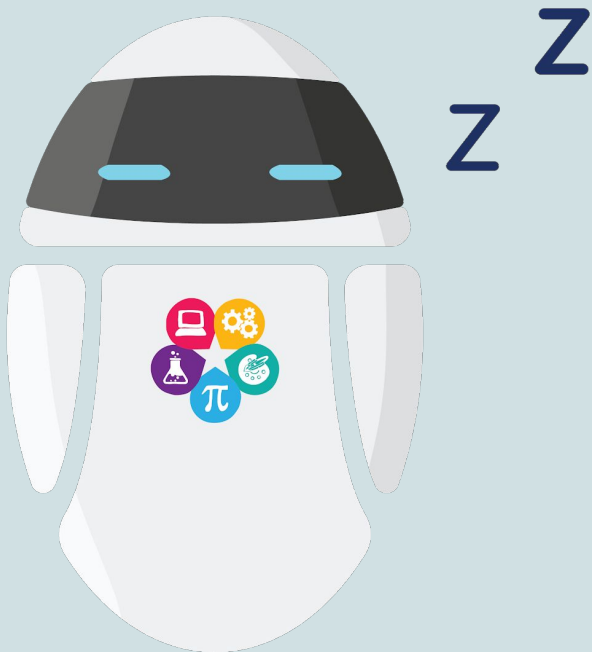


Thực hành



https://bit.ly/S4V_CS201_FSM_Scope_Practice





NGHỈ GIẢ LAO **10 PHÚT**

FINITE STATE MACHINE (FSM)

Finite State Machine(FSM) là gì?

Trạng thái máy hữu hạn là một mô hình tính toán dựa trên các trạng thái, mà ở mỗi thời điểm chỉ có một trạng thái nhất định.

Điều này có nghĩa là để thực hiện các hành động khác nhau thì máy phải chuyển từ trạng thái này sang trạng thái.

Finite State Machine

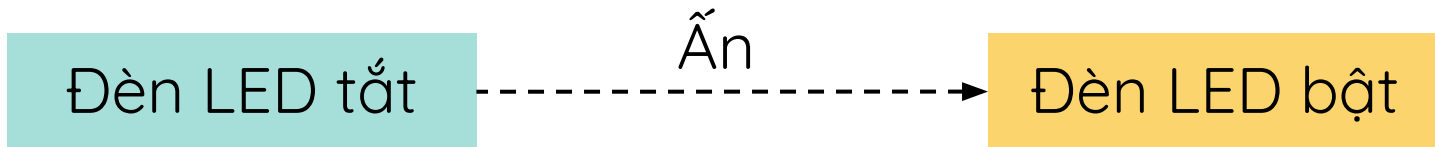
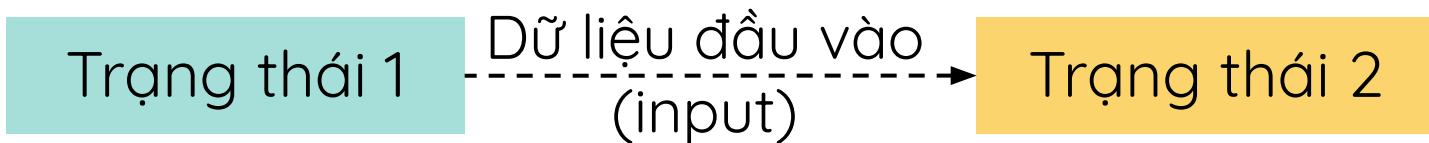
Một số
hữu hạn

Trạng thái tại
một thời điểm

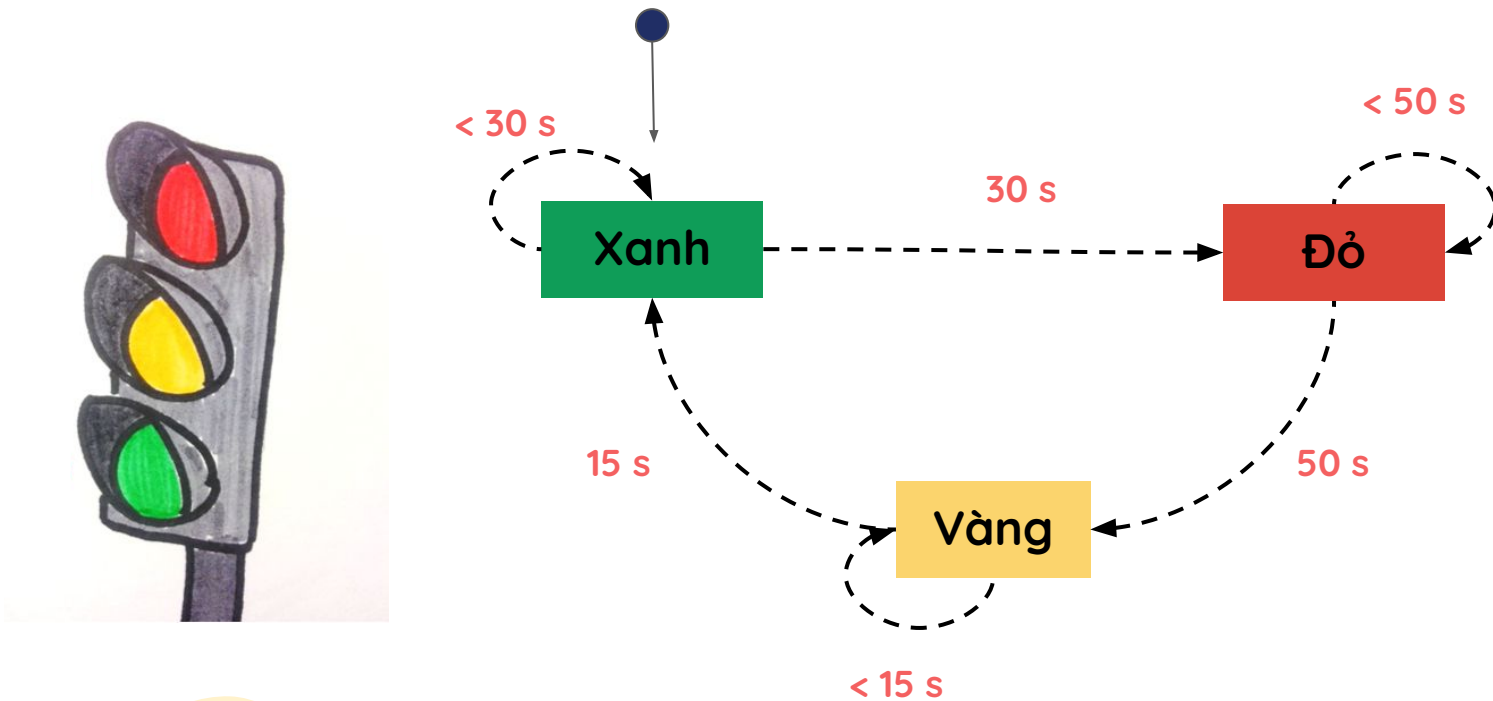
Máy móc, ...



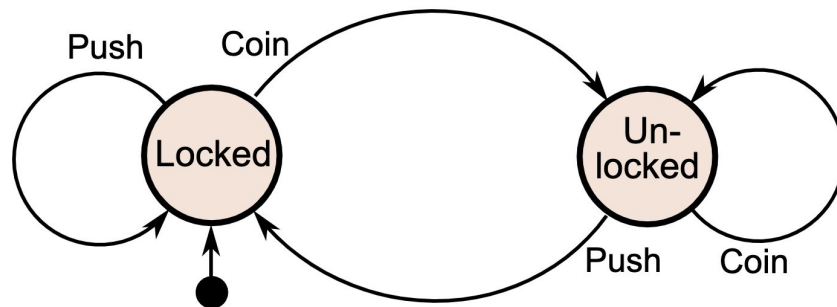
Sự chuyển đổi trạng thái trong FSM



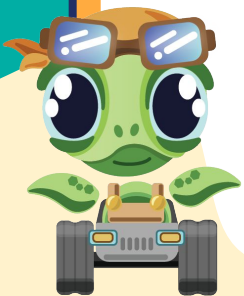
FSM trong thực tế: Đèn giao thông



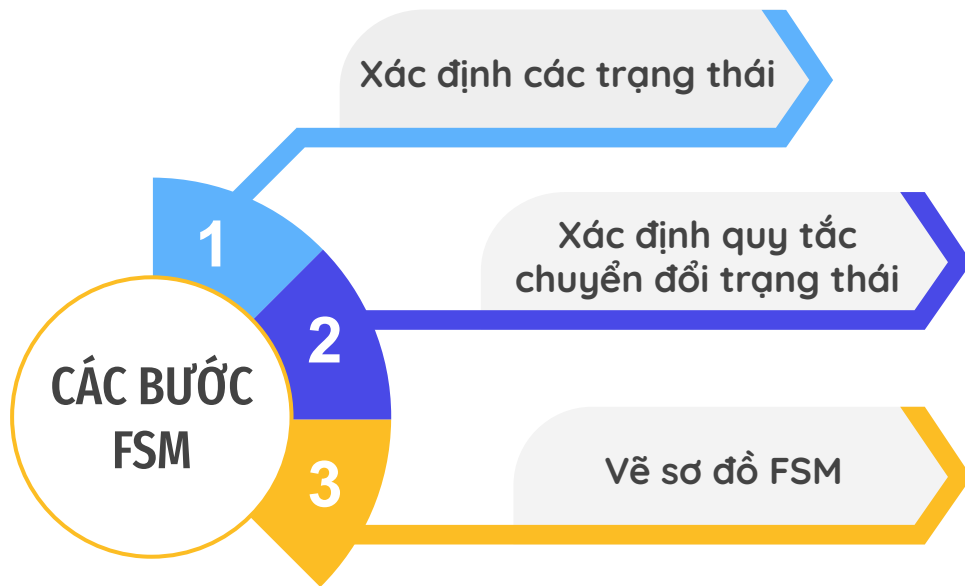
FSM trong thực tế: Cổng xoay tripod turnstile



(Cổng xoay ba càng, Cửa tự động)

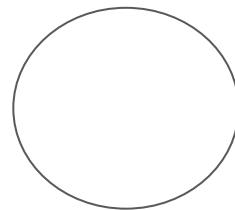


Cách dùng Finite State Machine



Quy ước chung

Trạng thái:



Chuyển đổi



Điểm bắt đầu:



Quy tắc

<điều kiện>



Lưu ý:

Các mũi tên chỉ có một hướng



Ứng dụng của FSM trong lập trình Robotics

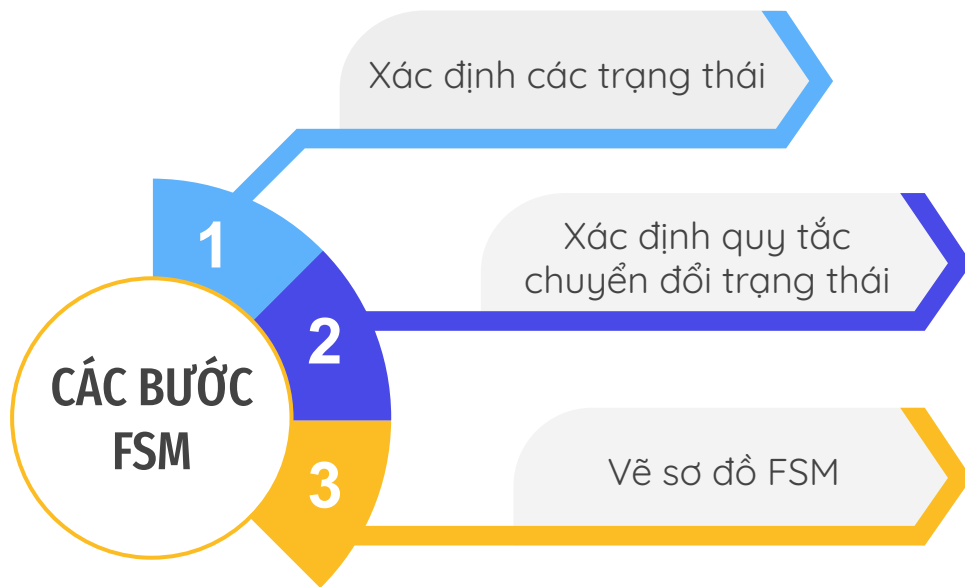
Ví dụ: Đèn LED chuyển đổi màu sắc dựa trên số lần ấn:

- Ấn lần 1: đèn bật màu xanh lá
- Ấn lần 2: đèn bật màu đỏ
- Ấn lần 3: đèn tắt về trạng thái ban đầu



Thực hành vẽ sơ đồ FSM

Nhắc lại:



Bước 1: Xác định các trạng thái

- Đèn LED tắt
- Đèn LED bật màu xanh lá
- Đèn LED bật màu đỏ

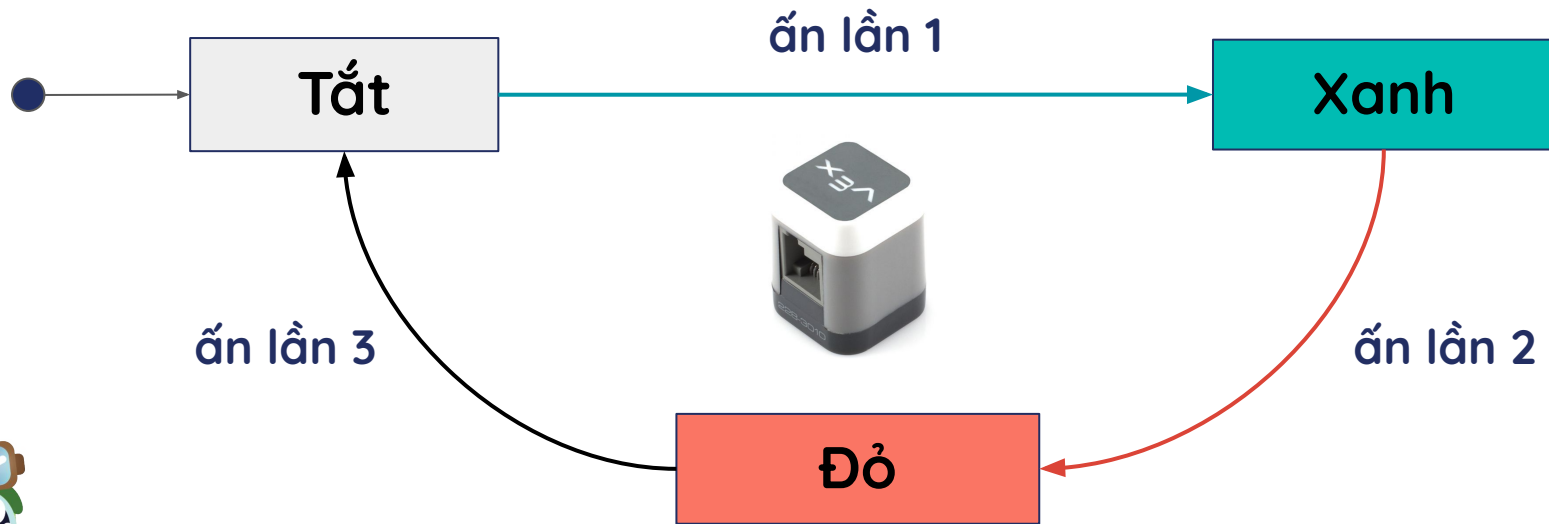


Bước 2: Xác định các quy tắc

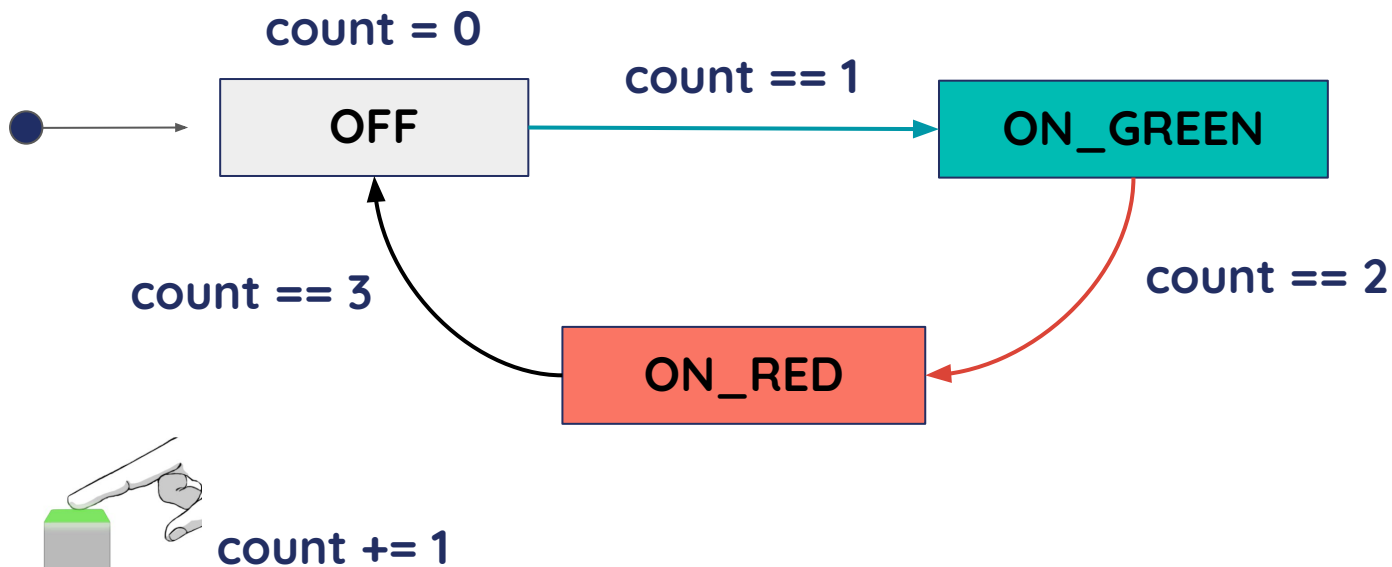
- Bật đầu đèn LED tắt
- Ấn lần 1 đèn LED chuyển sang màu xanh lá
- Ấn lần 2 đèn LED chuyển sang màu đỏ



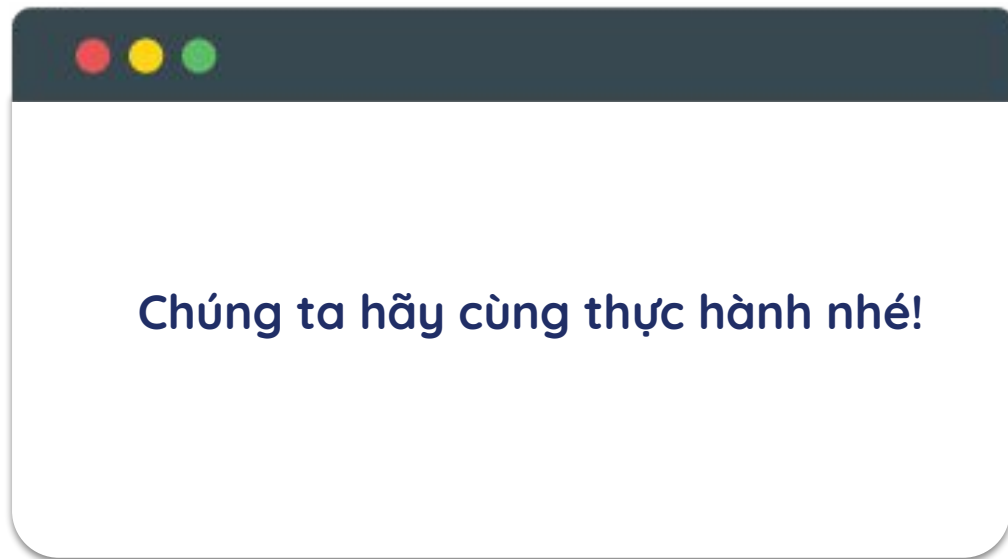
Bước 3: Vẽ sơ đồ chuyển đổi trạng thái(FSM)



Bước 3: Vẽ sơ đồ chuyển đổi trạng thái(FSM)



Thực hành



https://bit.ly/S4V_CS201_FSM_Touch_LED

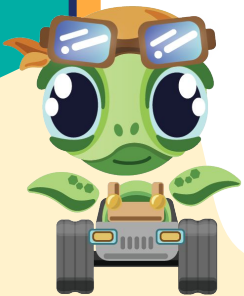


Vì sao nên dùng FSM?



Lợi ích khi dùng FSM trong lập trình robotics

- Dễ phân tích bài toán phức tạp hơn
- Mô phỏng tốt chương trình với sơ đồ trước khi lập trình
- Từng trạng thái chỉ nhận một số input nhất định
- Dễ sửa lỗi(debug)



Bài toán:

Viết chương trình sao cho khi khởi động đèn LED bật màu đỏ.

Khi ấn vào đèn LED, robot tiến về phía trước 50cm.

Nếu trong lúc di chuyển robot gặp phải vật cản thì robot sẽ rẽ trái 90 độ và dừng lại



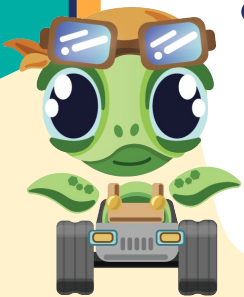
FSM - Bước 1 - Xác định các trạng thái

- Trạng thái nghỉ
- Trạng thái tiến
- Trạng thái rẽ

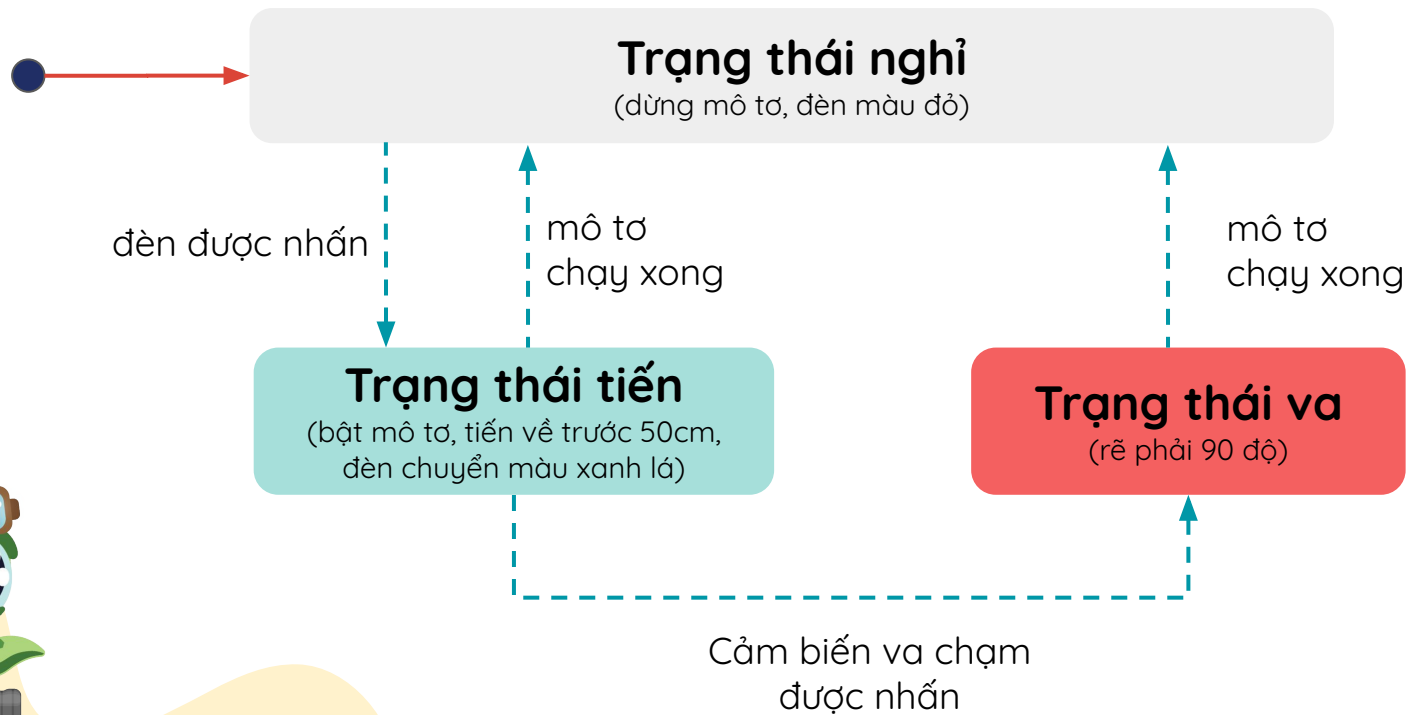


FSM - Bước 2 - Các quy tắc chuyển đổi trạng thái

- **Đèn LED được bấm** thì:
 - Đèn chuyển màu xanh lá
 - Robot di chuyển về trước 50cm
- **Cảm biến va chạm được bấm** thì:
 - Robot dừng lại và rẽ trái 90 độ
- **Motor chạy xong** thì:
 - Robot dừng lại
 - Đèn chuyển màu đỏ



FSM - Bước 3 - Sơ đồ chuyển đổi các trạng thái



Tóm tắt chương trình chính

```
if state == "IDLE_STATE":
```

```
    idle_state()
```

```
elif state == "FORWARD_STATE":
```

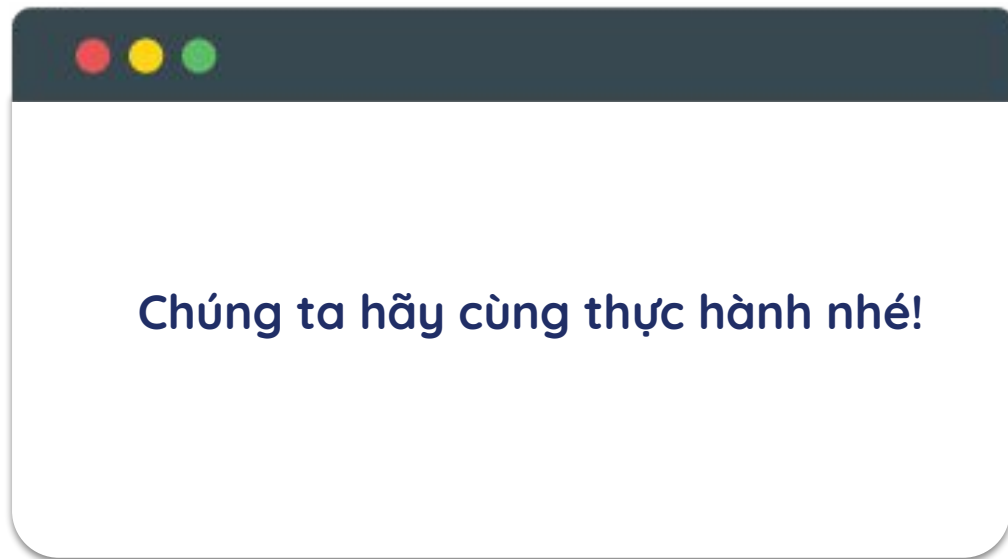
```
    forward_state()
```

```
elif state == "TURN_STATE":
```

```
    turn_state()
```



Thực hành



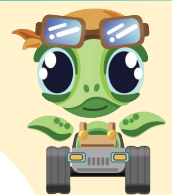
https://bit.ly/S4V_CS201_FSM_Robot_Moving





Hôm nay chúng ta đã học

- Scope(phạm vi) trong Python
- FSM(Finite state machine) (Trạng thái máy hữu hạn)
 - Tìm hiểu về FSM
 - Ứng dụng của FSM trong lập trình robotics



Bài tập tự luyện

Viết chương trình sao cho khi khởi động đèn LED bật màu đỏ.

Khi ấn vào đèn LED, robot tiến về phía trước.

Nếu trong lúc di chuyển robot gặp phải vật cản thì robot sẽ rẽ trái 90 độ và tiếp tục tiến về phía trước.

LỊCH BUỔI TIẾP THEO

Chủ Nhật tuần sau (17/07/2021)

Giờ học: 7:30 sáng giờ VN





Hẹn gặp lại!