Introduction To Methods

What Are Methods?

- Methods (also called Functions) execute blocks of code that makes our game do things.
- To achieve this we must:
  1. DECLARE and define our method
  2. CALL our method when we want it to execute

Real World Example

DECLARE Method

CleanYourRoom

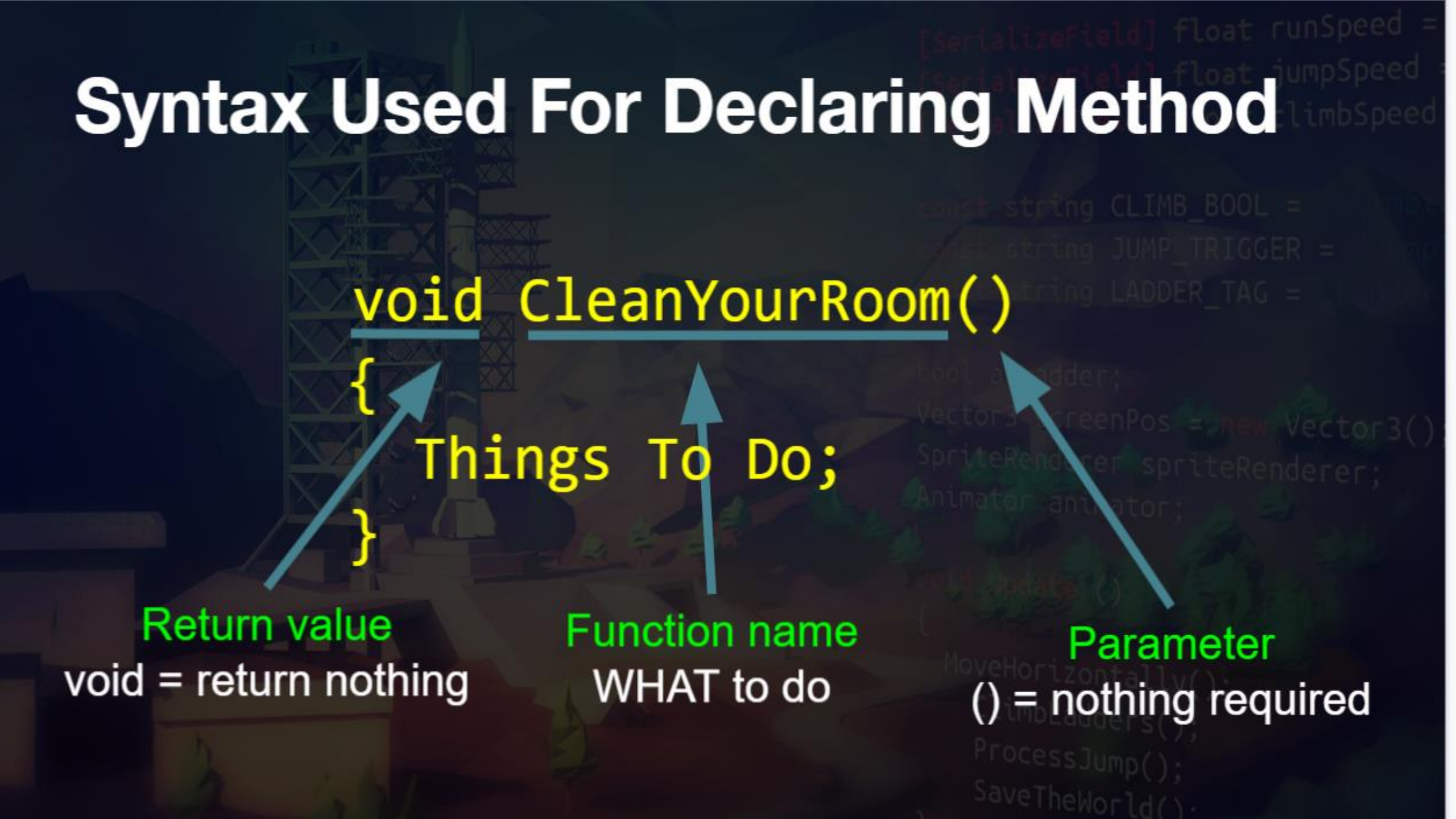
- Pick up clothes
- Throw out garbage
- Kill the mutated science project that threatens to destroy life as we know it

CALL Method

Hey, go and *CleanYourRoom*

"Go and do the steps we've defined and agreed to"

Syntax Used For Declaring Method



More Flavour Can Be Added

Two more common things when calling methods:

1. We can ask for some information to be RETURNED
2. We can specify some PARAMETERS are needed when calling the method

Using our real world example again...

More Flavour Can Be Added

DECLARE Method

CleanYourRoom(deadline)

- Do it before (deadline)
- Pick up clothes
- Throw out garbage
- Kill the mutated science project that threatens to destroy life as we know it
- Is the room dirty?

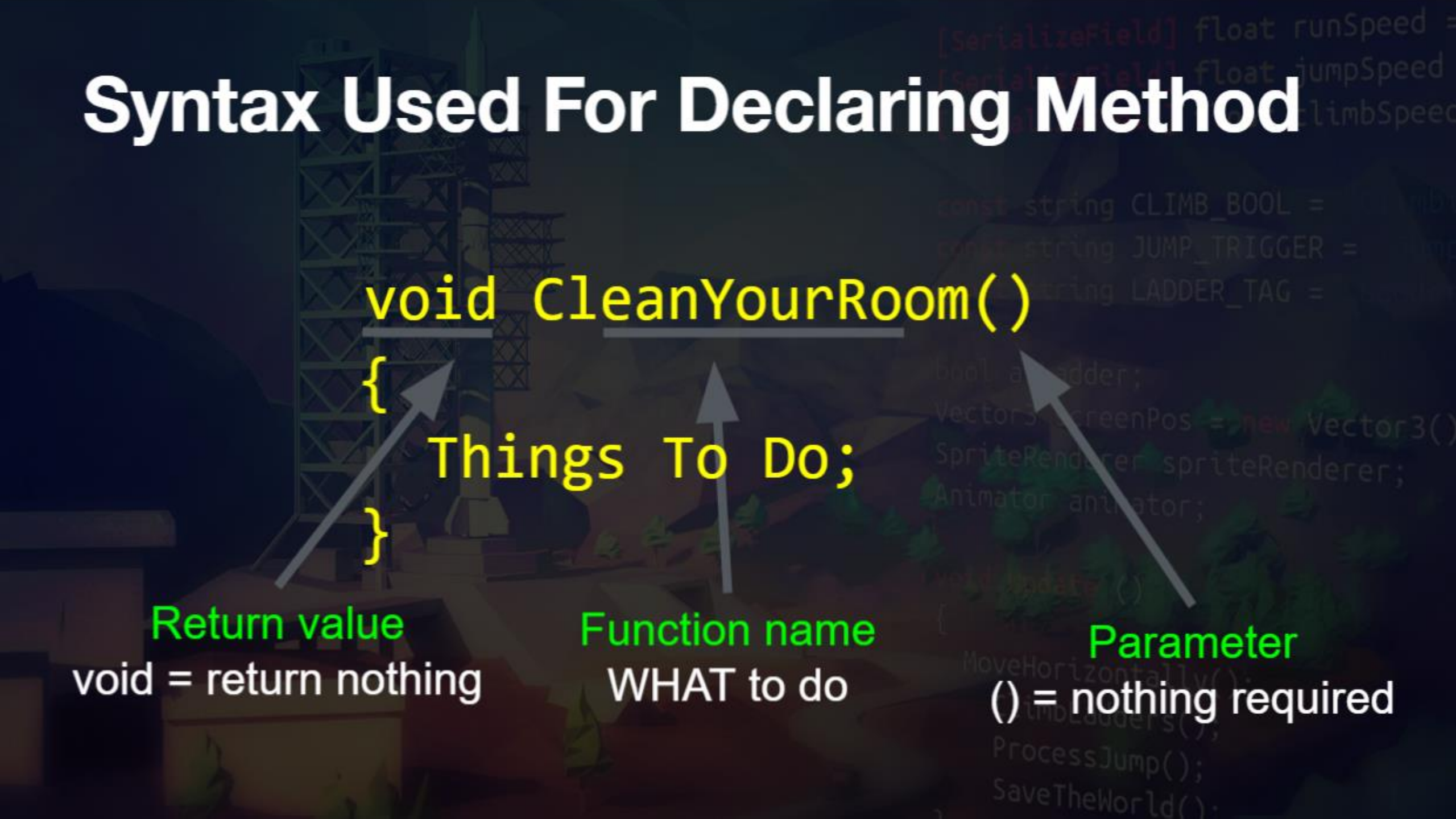
CALL Method

Hey, go and *CleanYourRoom* (*you've got 1 hour*). *isRoomDirty?*

*Parameter / Argument

*Return data

Syntax Used For Declaring Method



Syntax Used For Declaring Method

```
bool CleanYourRoom(int time)
{
    Things To Do;
    Deadline = time;
    return isRoomDirty;
}
```

Return value
(Type = bool)

Return keyword
(Type = bool)

Parameter
Needs type
and name

Syntax Used For Calling Method

```
CleanYourRoom();
```

Remember semi colon

Executes statements defined in the code block...

```
{
  …
}
```

Defining & Calling

DECLARE Method

```
void CleanYourRoom()
{
    Pick up clothes;
    Clean garbage;
    Kill mutated experiment;
}
```

CALL Method

```
CleanYourRoom();
```

But What About Start() & Update()?!

- We've been defining `Start()` & `Update()` but not calling them
- Where are they called?
- Unity's internal logic is taking care of calling them for us at the right time
- These are referred to as "callbacks"