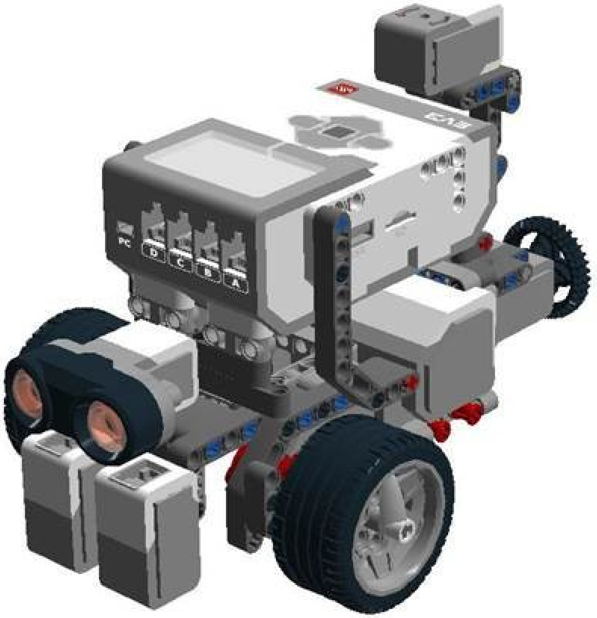
LEGO

MINDSTORMS® EV3

Pybricks on EV3



python™

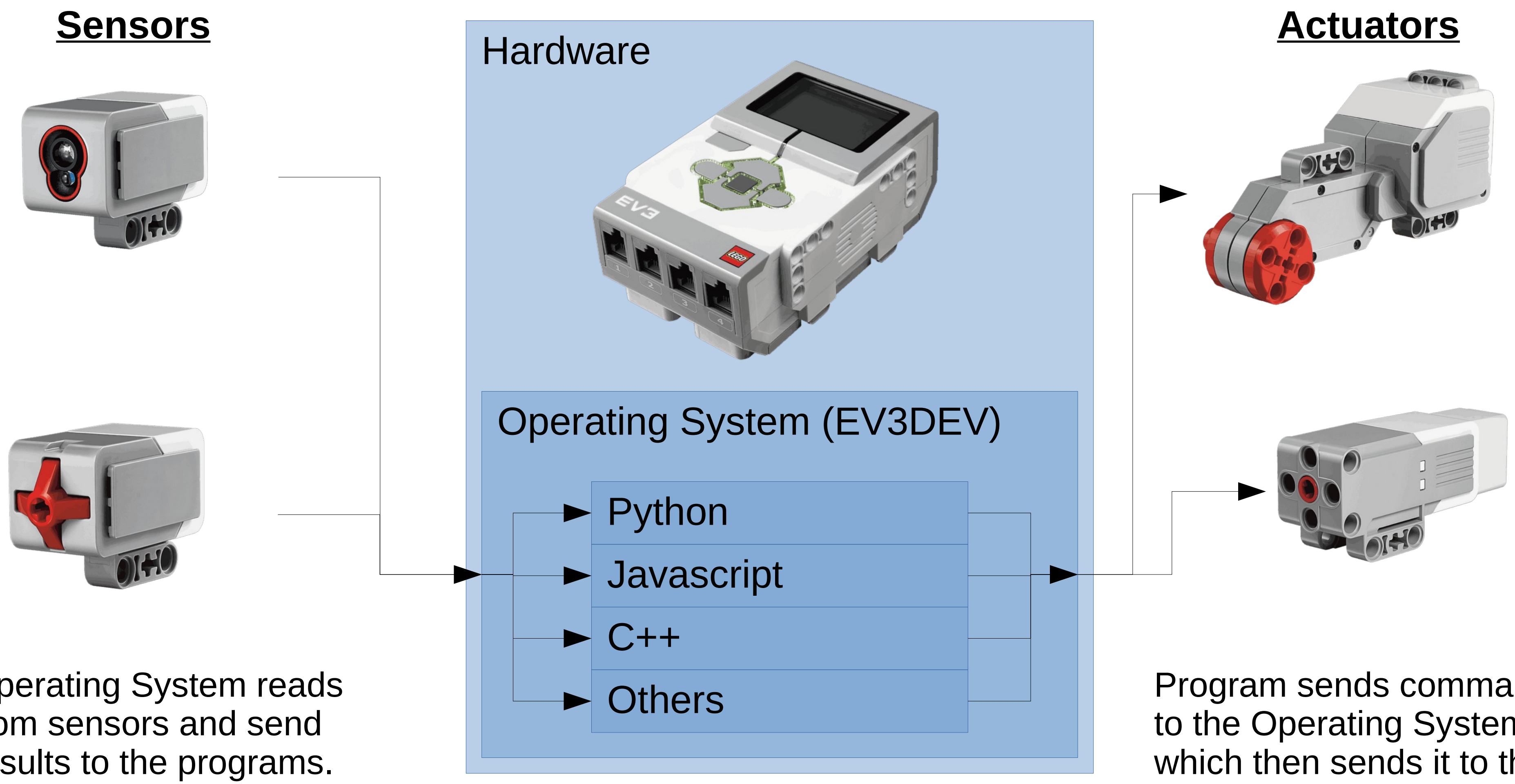
A POSTERIORI

Play · Experience · Learn

What is EV3DEV?

- A Linux-based operating system that runs on the Lego EV3
  - Info: Your EV3 is already running Linux, just a different type
- Runs from a microSD card
- Can run programs written in Python, Javascript, Java, Go, C++, C, and many others
- Can connect keyboard, mouse, webcam, internet, to your EV3

How EV3DEV works?



Sensors

Hardware

Operating System (EV3DEV)

Python

Javascript

C++

Others

Actuators

Operating System reads from sensors and send results to the programs.

EV3DEV replaces the original EV3 operating system (...which is also Linux based), and runs inside the EV3, similar to how windows runs inside a laptop or PC.

Program sends commands to the Operating System, which then sends it to the actuators

What is Pybricks?

- Modified version of EV3Dev
- Only supports micro-python
    - Works just like Python, but slightly simplified to support low end devices
- Uses a different Python API (Application Programming Interface)
- Runs faster than the Python API in EV3Dev
- Also available for Lego Boost, Spike Prime, Technic, etc

EV3Dev VS Pybricks

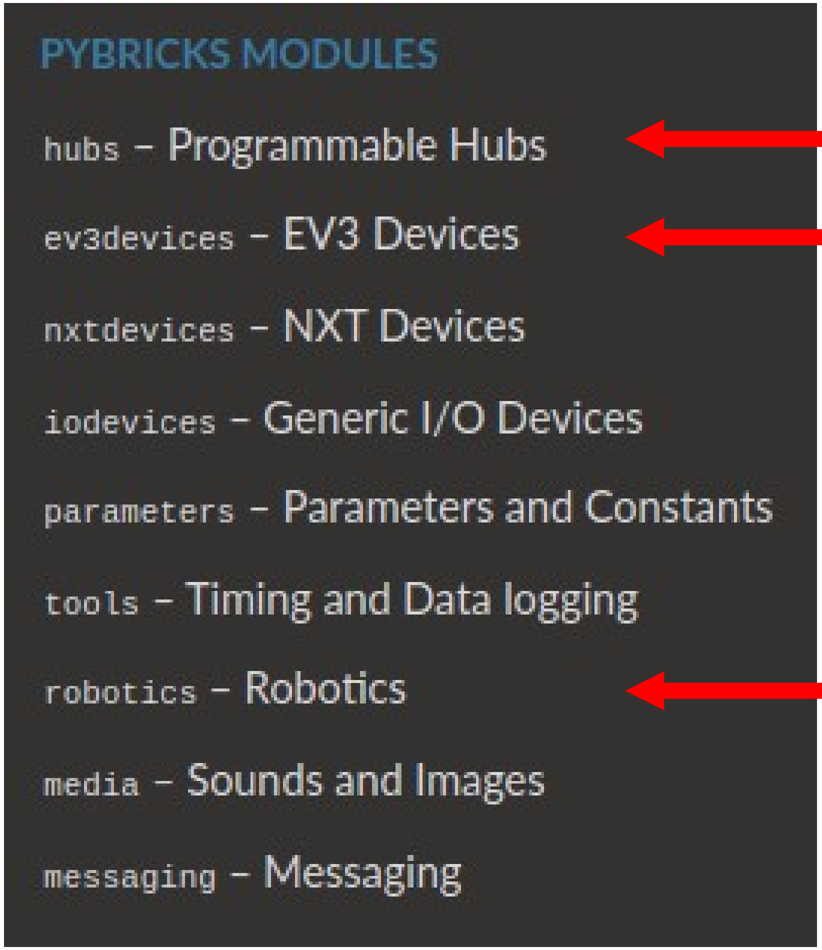
EV3Dev	Pybricks
Support many different programming languages	Only support Python (micro-python)
Only works on EV3	Works on EV3, Spike, Boost, Etc
API provides more advanced capabilities	API is simpler
Performance is poor	Higher performance

Using Pybricks

- EV3 Robot
  - https://pybricks.com/ev3-micropython
  - Follow the installation instructions
  - Write and upload program using VS Code https://code.visualstudio.com/
- Simulator
  - https://a9i.sg/gears
  - In menu, click on “Python” then switch to “Pybricks Mode”
  - Switch to Python tab and write code
  - Switch to Simulator and run

Coding in Pybricks

- Read the documentation



PYBRICKS MODULES

`hubs` – Programmable Hubs

`ev3devices` – EV3 Devices

`nxtdevices` – NXT Devices

`iodevices` – Generic I/O Devices

`parameters` – Parameters and Constants

`tools` – Timing and Data logging

`robotics` – Robotics

`media` – Sounds and Images

`messaging` – Messaging

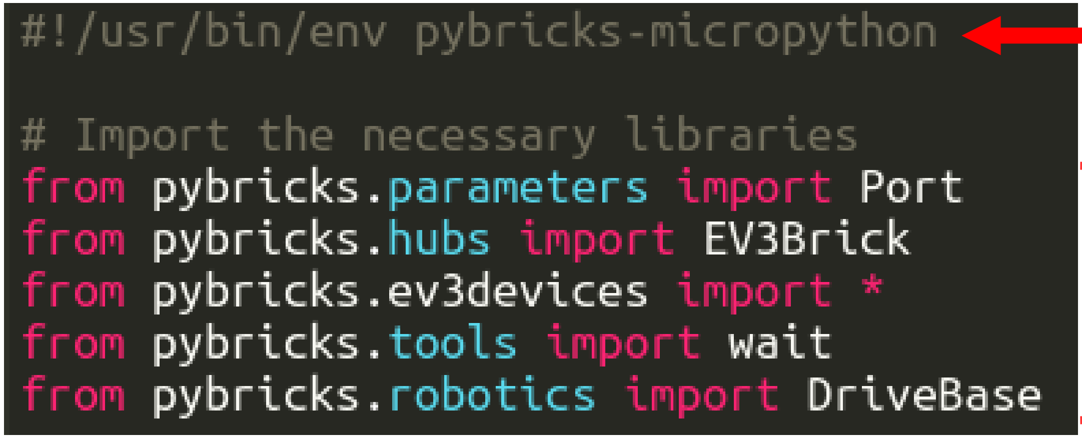
Speaker, LED light, screen

Sensors and motors

Drive base (control two motors together)

You can read the rest, but they are not as important

Coding Quick Start
Imports (Auto-generated)



```
#!/usr/bin/env pybricks-micropython

# Import the necessary libraries
from pybricks.parameters import Port
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import *
from pybricks.tools import wait
from pybricks.robotics import DriveBase
```

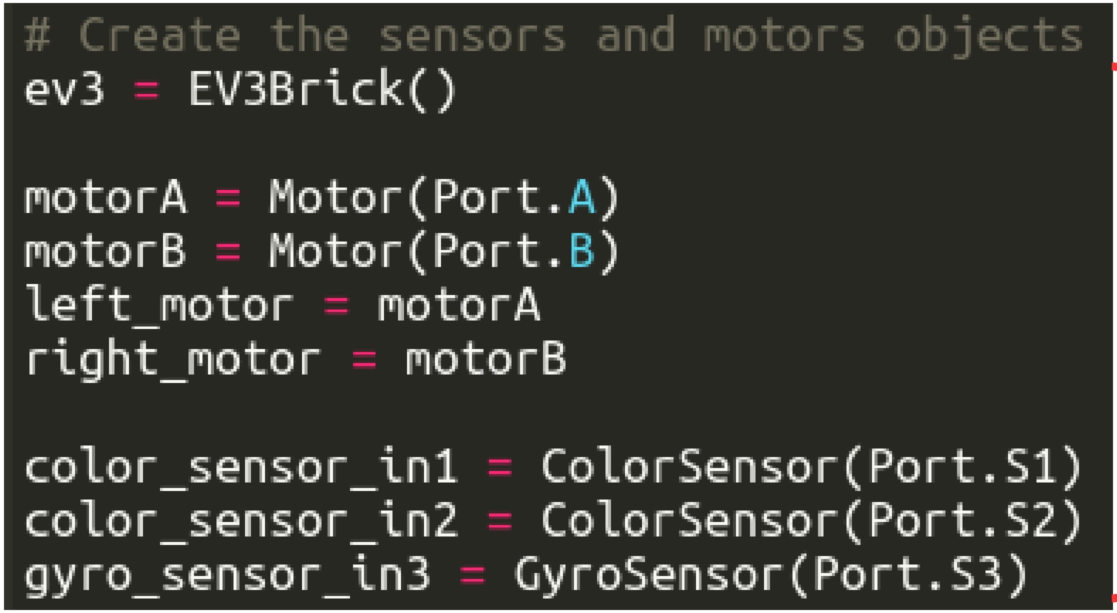
Tells OS this is a pybricks program

Import libraries

For the most part, this is the same for every program.

You may choose to modify it to import more modules
(eg. if you want to send bluetooth messages to another robot)

Coding Quick Start
Create Objects (Auto-generated)



```
# Create the sensors and motors objects
ev3 = EV3Brick()

motorA = Motor(Port.A)
motorB = Motor(Port.B)
left_motor = motorA
right_motor = motorB

color_sensor_in1 = ColorSensor(Port.S1)
color_sensor_in2 = ColorSensor(Port.S2)
gyro_sensor_in3 = GyroSensor(Port.S3)
```

Importing the modules provides **Classes**

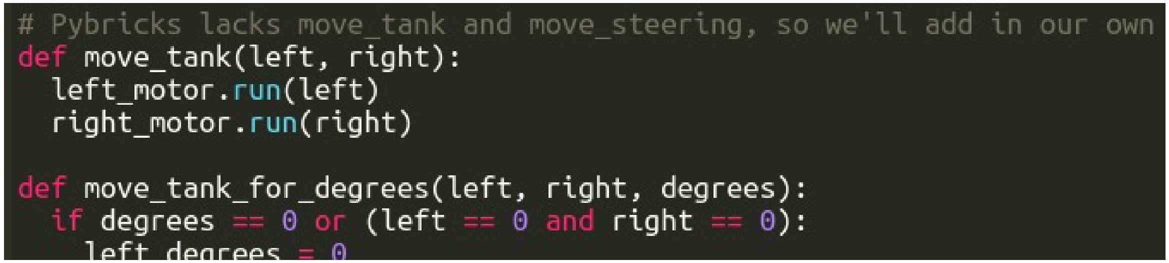
To use the modules, you need to create **Objects** for each sensor and actuator

This will need to be modified to suit each robot

The name of each object (eg. "ev3", "motorA", "color_sensor_in1") is up to you. You can name them whatever you want.

The ports (eg. "Port.A", "Port.S1") should obviously match what you have on your robot. In the simulator, this is done for you automatically.

Coding Quick Start
Move Functions (Auto-generated)

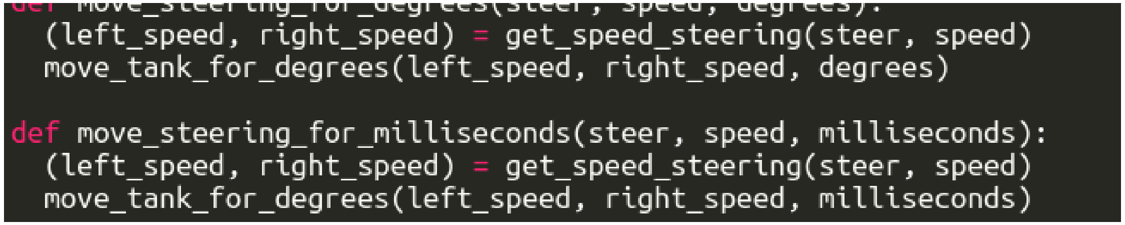


```
# Pybricks lacks move_tank and move_steering, so we'll add in our own
def move_tank(left, right):
  left_motor.run(left)
  right_motor.run(right)

def move_tank_for_degrees(left, right, degrees):
  if degrees == 0 or (left == 0 and right == 0):
    left_degrees = 0
```

.

.

.



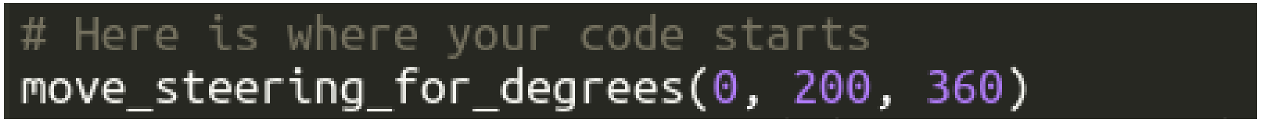
```
def move_steering_for_degrees(steer, speed, degrees):
  (left_speed, right_speed) = get_speed_steering(steer, speed)
  move_tank_for_degrees(left_speed, right_speed, degrees)

def move_steering_for_milliseconds(steer, speed, milliseconds):
  (left_speed, right_speed) = get_speed_steering(steer, speed)
  move_tank_for_degrees(left_speed, right_speed, milliseconds)
```

Pybricks lacks **move_tank** and **move_steering**

These codes here provides replacement functions for them.

You can delete these if you don’t intend to use move_tank and move_steering.

Coding Quick Start
Your Code (Write it yourself)



```
# Here is where your code starts
move_steering_for_degrees(0, 200, 360)
```

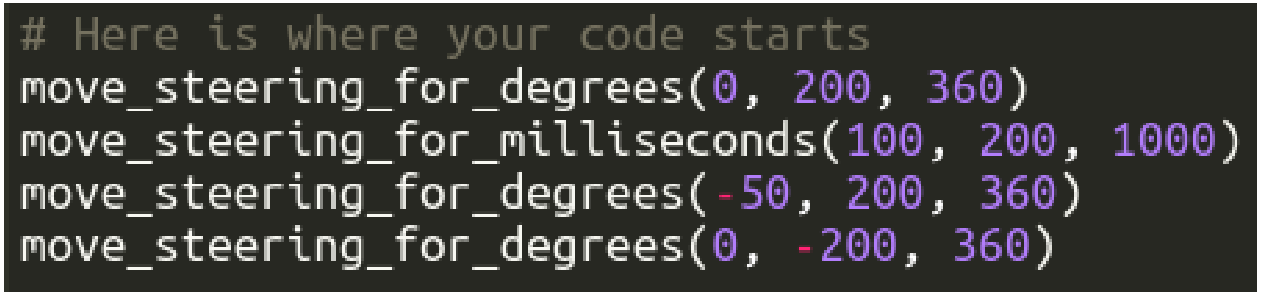
move_steering_for_degrees(steering, speed, degrees)

- **steering: -100 to 100**
  - **-100 : Spin turn left**
  - **0 : Straight**
  - **50 : Pivot turn right**
  - **100 : Spin turn right**
  - **Same as the Lego EV3 software (Labview or Classroom)**

- **speed: -1000 to 1000 (approximate)**
  - **In degrees per second.**
  - **Max speed depends on battery, motors, load, etc**

- **degrees: Any**
  - **Degrees to turn. 360 means one rotation.**

Coding Quick Start
Your Code (Write it yourself)



```
# Here is where your code starts
move_steering_for_degrees(0, 200, 360)
move_steering_for_milliseconds(100, 200, 1000)
move_steering_for_degrees(-50, 200, 360)
move_steering_for_degrees(0, -200, 360)
```

move_steering_for_degrees(0, 200, 360)

- **0: Move straight ahead**
- **200: At speed 200 degrees / second**
- **360: For 360 degrees (1 rotation)**

move_steering_for_milliseconds(100, 200, 1000)

- **100: Spin turn right**
- **200: At speed 200 degrees / second**
- **1000: For 1000 milliseconds (1 second)**

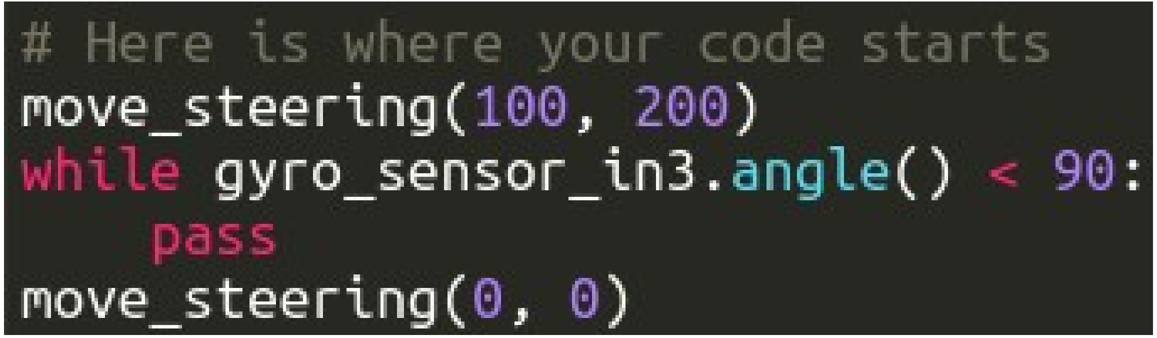
move_steering_for_degrees(-50, 200, 360)

- **Pivot turn left (left wheel stationary, right wheel forward)**

move_steering_for_degrees(0, -200, 360)

- **Move backwards (straight)**

Using Sensors



```
# Here is where your code starts
move_steering(100, 200)
while gyro_sensor_in3.angle() < 90:
    pass
move_steering(0, 0)
```

Read the documentation to see what you can get from each sensor!

move_steering(100, 200)

- **100: Spin turn right**
- **200: At speed 200 degrees / second**
- **This one doesn’t have a degree or time.**
- **Function completes immediately, but robot will continue moving forever until given a different command**

gyro_sensor_in3.angle()

- **Provides the current angle**
- **While angle is less than 90, “pass” (do nothing)**

move_steering(0, 0)

- **0: Speed 0 degrees / second (Stop)**

Copyright

- Created by A Posteriori LLP
- Visit http://aposteriori.com.sg/ for more tips and tutorials

A POSTERIORI

Play · Experience · Learn