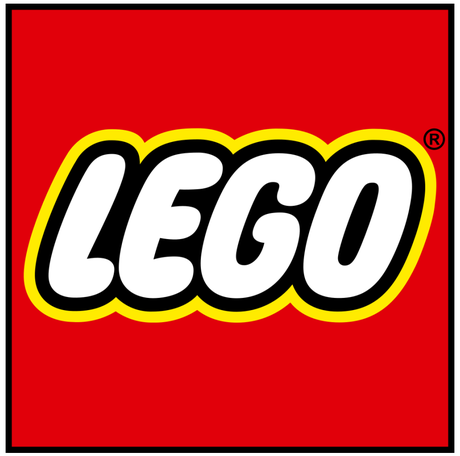
# Parkside Montessori




May 2, 2024

# Programming in Python

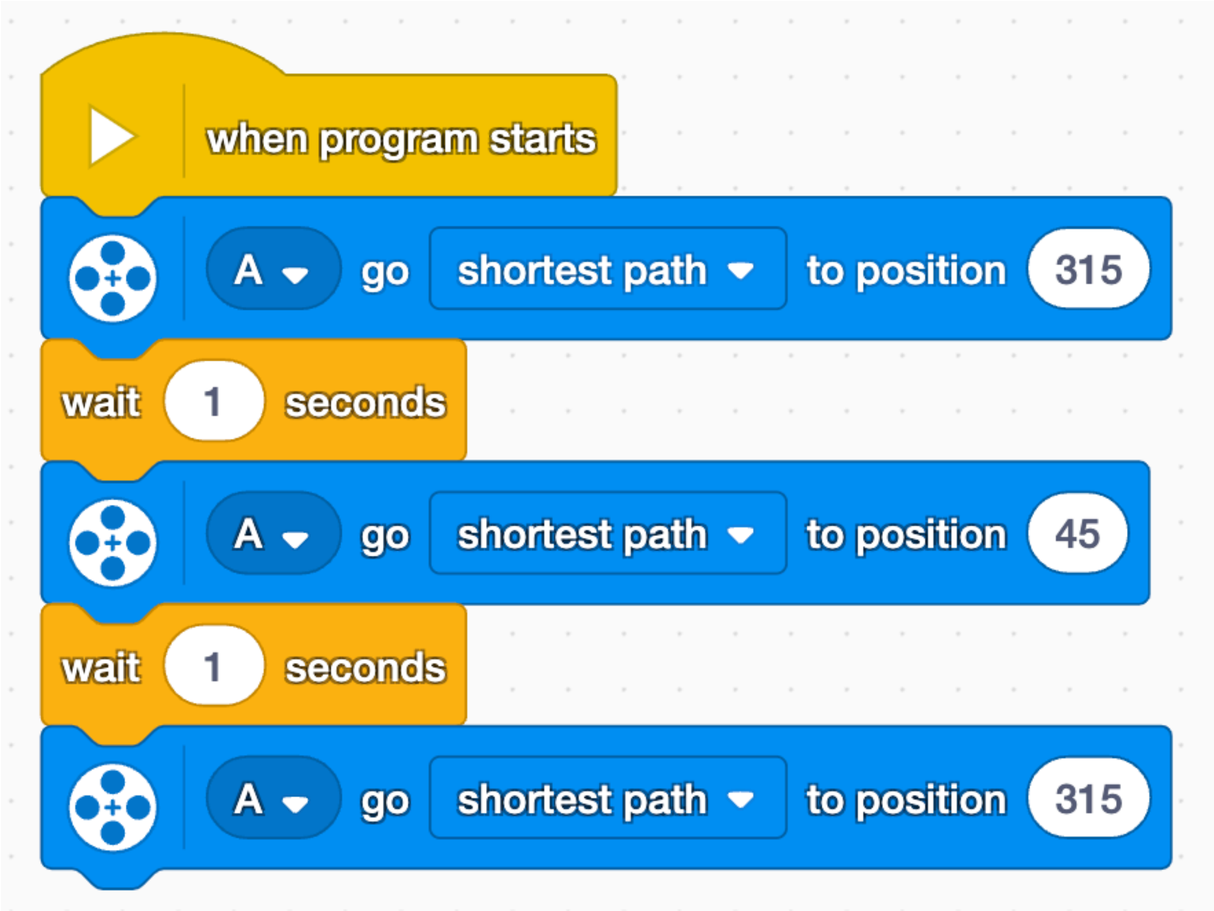
# Programming in Python

- Lots of information today
- Overwhelming at first, but will slowly make sense
- Different levels of experience
- Ask lots of questions!

# Why? I like Scratch...

- More powerful and expressive language
- You can do more than program LEGO
  - Run websites
  - Create games
  - Whatever you can imagine!
- Easier to share and work in larger teams
- Millions of people use it

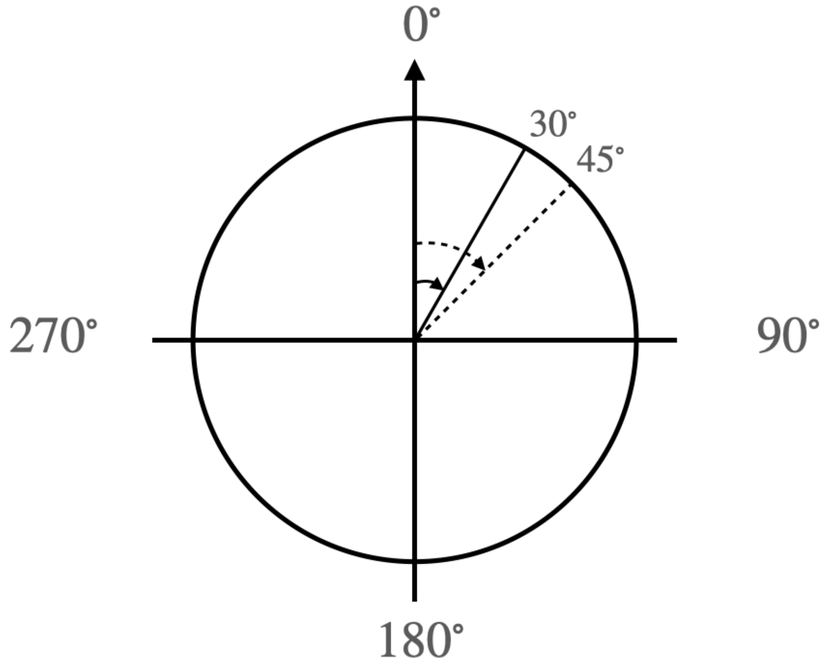
# Objective




← Let’s write this in Python

# Measuring Rotation

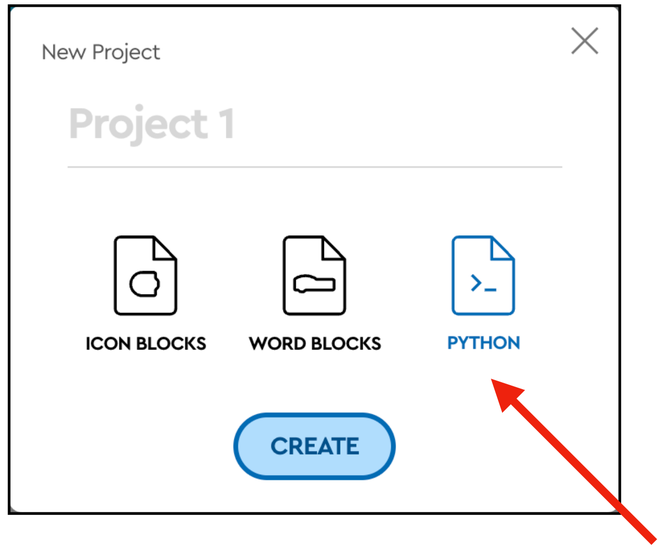
- How do you tell a motor exactly how much to turn?
- Turn amounts are measured in units called *degrees*
- A full circle is 360 degrees

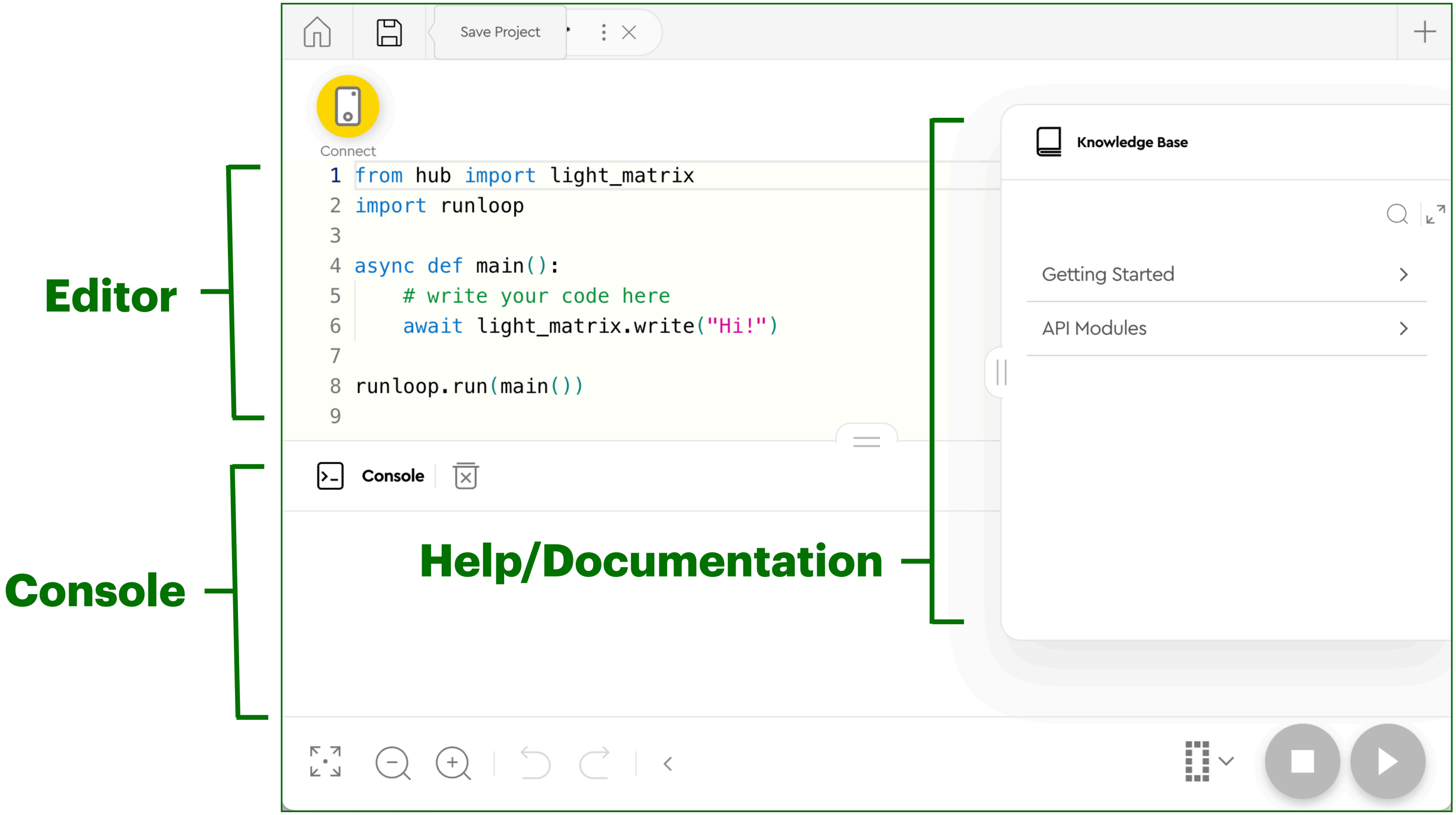


# Getting Started

https://spike.legoeducation.com

Spike Prime > New Project > Python




Save Project
Connect
1 from hub import light_matrix
2 import runloop
3
4 async def main():
5 # write your code here
6 await light_matrix.write("Hi!")
7
8 runloop.run(main())
9
Console
Knowledge Base
Getting Started
API Modules
Editor
Console
Help/Documentation

# A Python Program

```
from hub import light_matrix
import runloop

async def main():
    # write your code here
    await light_matrix.write("Hi!")

runloop.run(main())
```

# A Python Program

```
from hub import light_matrix
import runloop

async def main():
    # write your code here
    await light_matrix.write("Hi!")

runloop.run(main())
```

* We'll ignore the crossed-out words for now to keep things simple

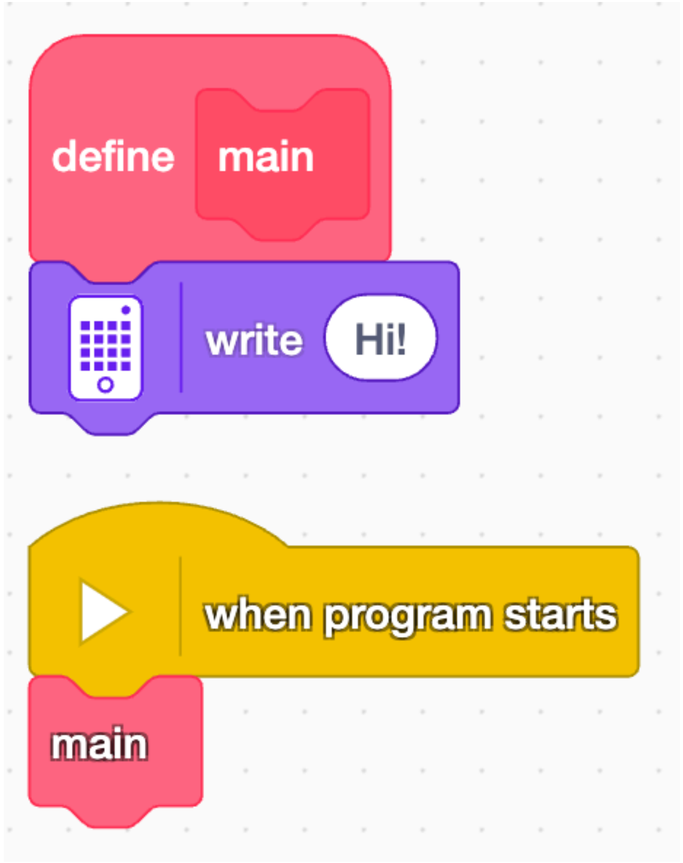
# A Python Program

```
from hub import light_matrix

def main():
    # write your code here
    light_matrix.write("Hi!")

main()
```

# Program Direction

Programs run top to bottom one block at a time

Each block is one command



# Program Direction

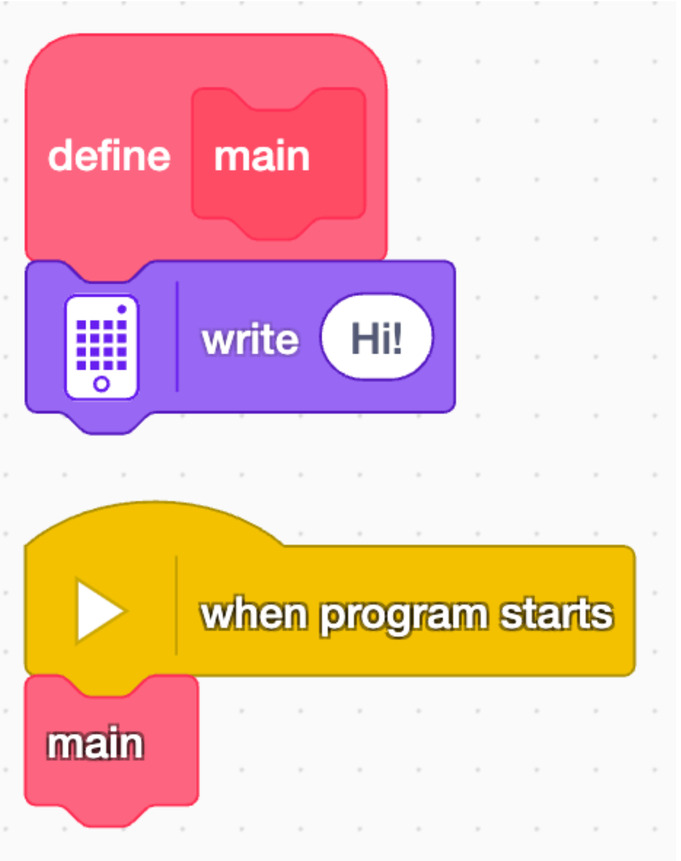
Programs run top to bottom one line at a time

Each line of text is called a *line of code*

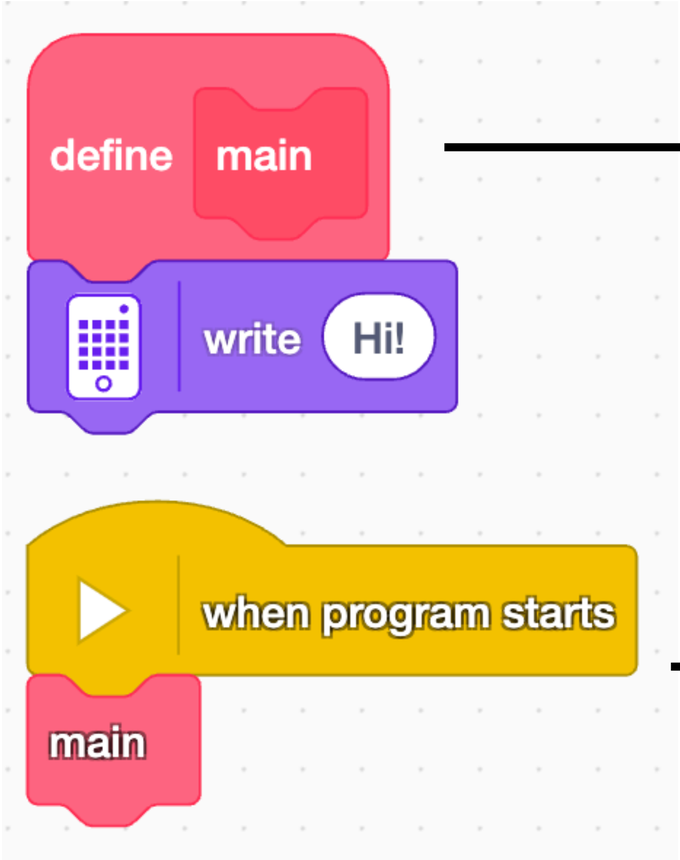
```
from hub import light_matrix

def main():
    # write your code here
    light_matrix.write("Hi!")

main()
```

# A Scratch Program

What does this program do?



# A Python Program




```
def main():
    light_matrix.write("Hi!")
```

```
main()
```

# Parts of a Program

Function definition ⟶

```
def main():
    light_matrix.write("Hi!")
```

Function call ⟶

```
main()
```

# Functions

definition keyword

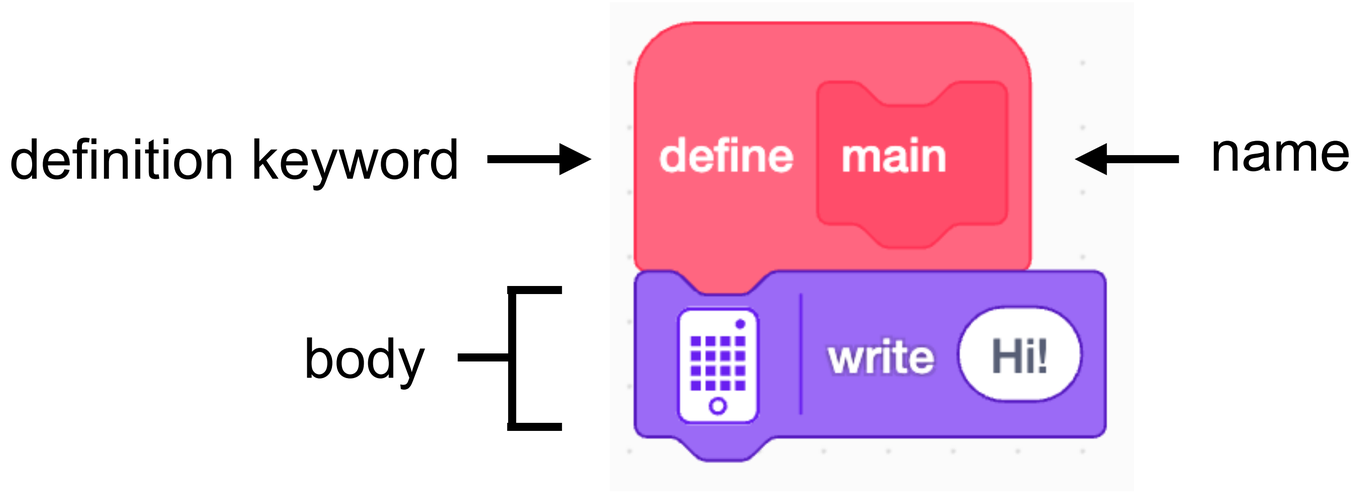
name

ending

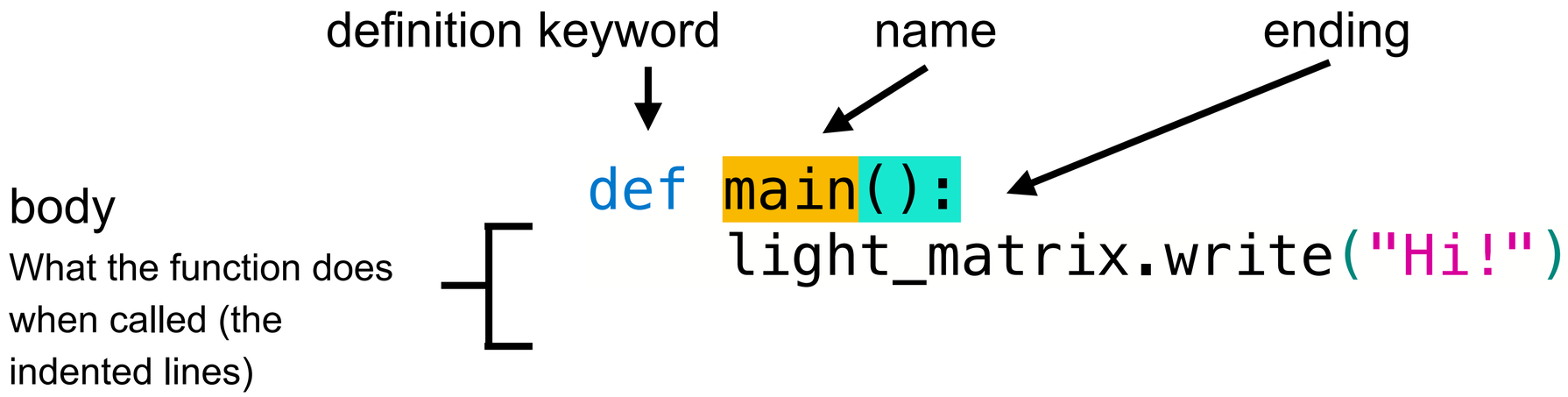
```
def main():
    light_matrix.write("Hi!")
```

body

What the function does when called (the indented lines)



# Functions

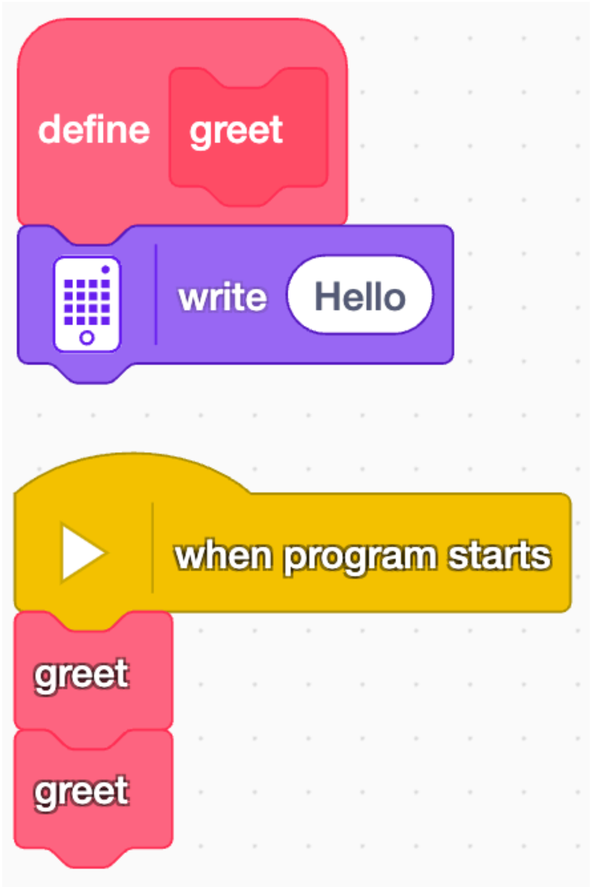



**Function Naming**

- Letters, numbers and some symbols are allowed
- No spaces allowed
- Convention is to not use uppercase letters

```
def raise_arm():
def Make Tea():
def climb@rock():
```

# Functions

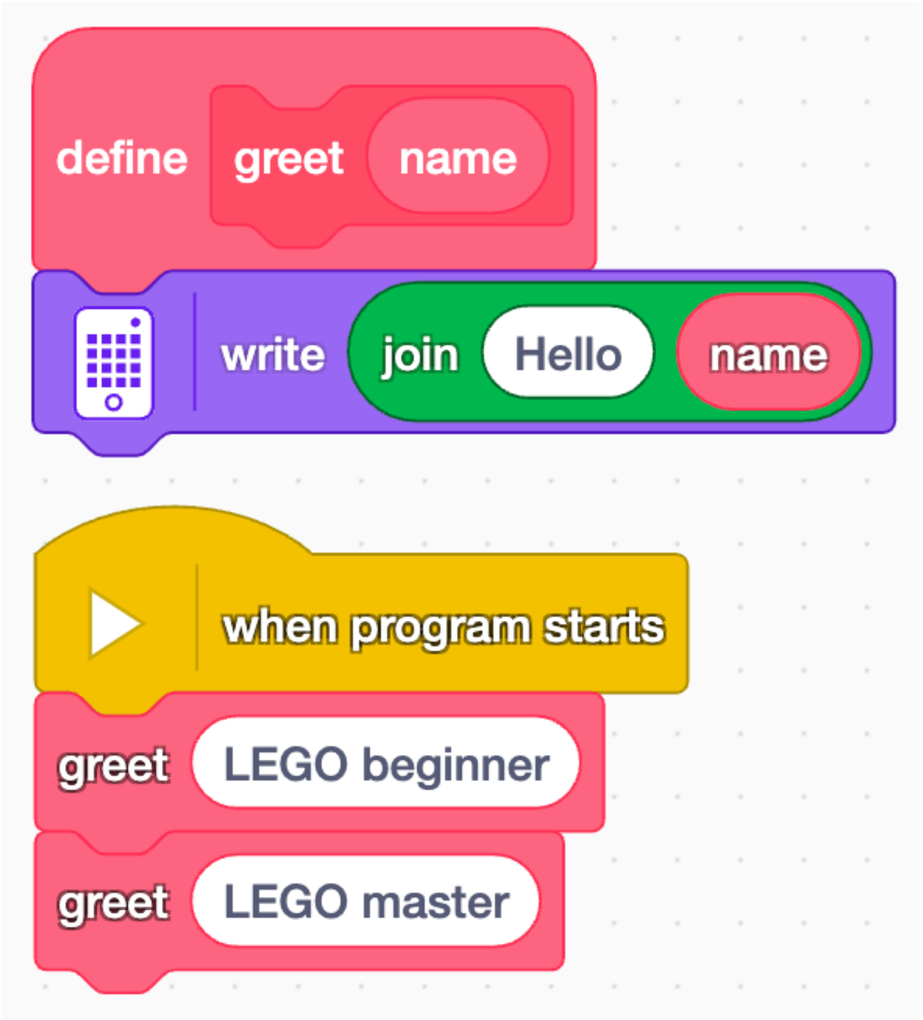



```
def greet():
    light_matrix.write("Hello")

greet()
greet()
```

What will happen if we call *greet()* twice?

What if I want a different greeting for each call?

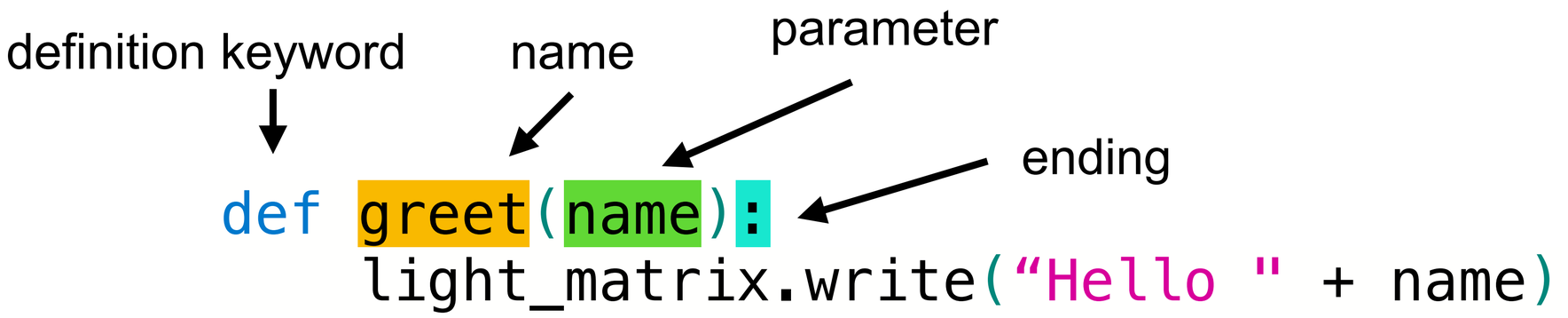
# Functions with Parameters




```
def greet(name):
    light_matrix.write(“Hello " + name)
```

```
greet("LEGO beginner")
greet("LEGO master")
```

# Functions with Parameters




function call with parameter
“LEGO beginner”

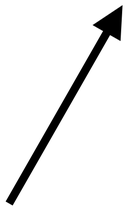
function call with parameter
“LEGO master”

```
greet("LEGO beginner")
greet("LEGO master")
```

# Parts of a Program

**Module Import**

- Used to organize related code
- Allows you to use code written by other people
- This module helps control the light matrix on the LEGO controller

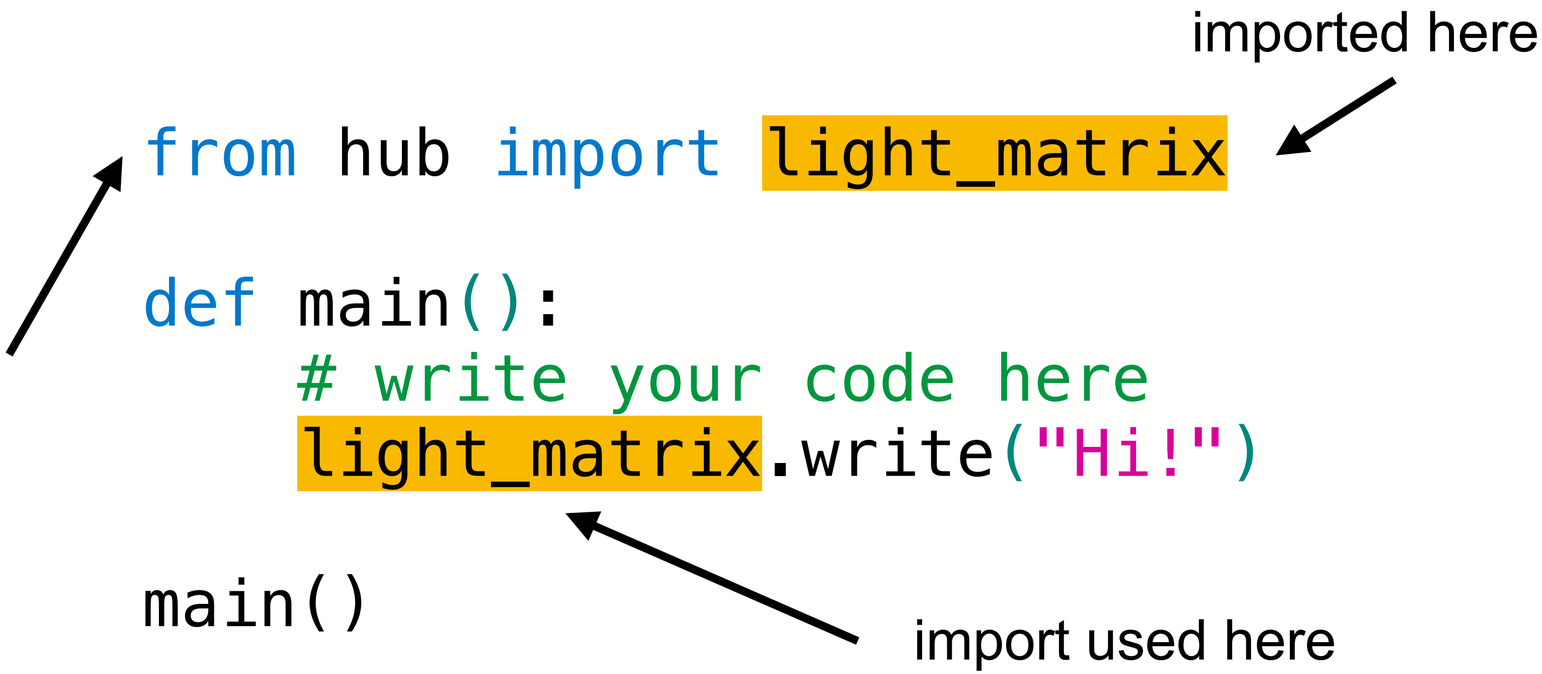


```
from hub import light_matrix

def main():
    # write your code here
    light_matrix.write("Hi!")

main()
```

# Parts of a Program

**Module Import**

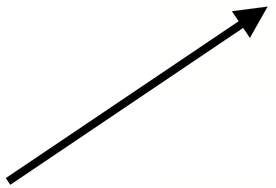
- Used to organize related code
- Allows you to use code written by other people
- This module helps control the light matrix on the LEGO controller



# Parts of a Program

**Comment**

- Text added to help another person understand your program
- You can put comments anywhere
- Anything to the right of “#” is ignored by your program

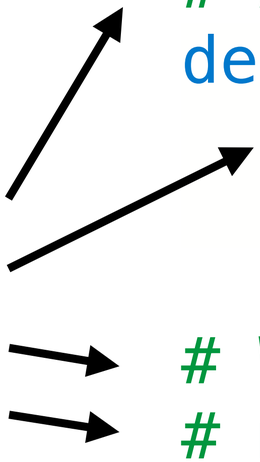


```
from hub import light_matrix

def main():
    # write your code here
    light_matrix.write("Hi!")

main()
```

# Parts of a Program

**Comments**

- Text added to help another person understand your program
- You can put comments anywhere
- Anything to the right of “#” is ignored by your program



```
from hub import light_matrix

# A comment
def main():
    # Another comment
    light_matrix.write("Hi!")

# Will this next line get run?
# main()
```

# The Await Keyword

Import *runloop* module

*await* - Wait for *write()* to finish before running next line

*async* - Lets you use *await* in a function

Lets *main()* use *await*

```
from hub import light_matrix
import runloop

async def main():
    await light_matrix.write("Hi!")
    await light_matrix.write("Ho!")

runloop.run(main())
```

Most function calls that control something connected to the LEGO controller should have *await* in front

# Python is Picky

**Indentation**

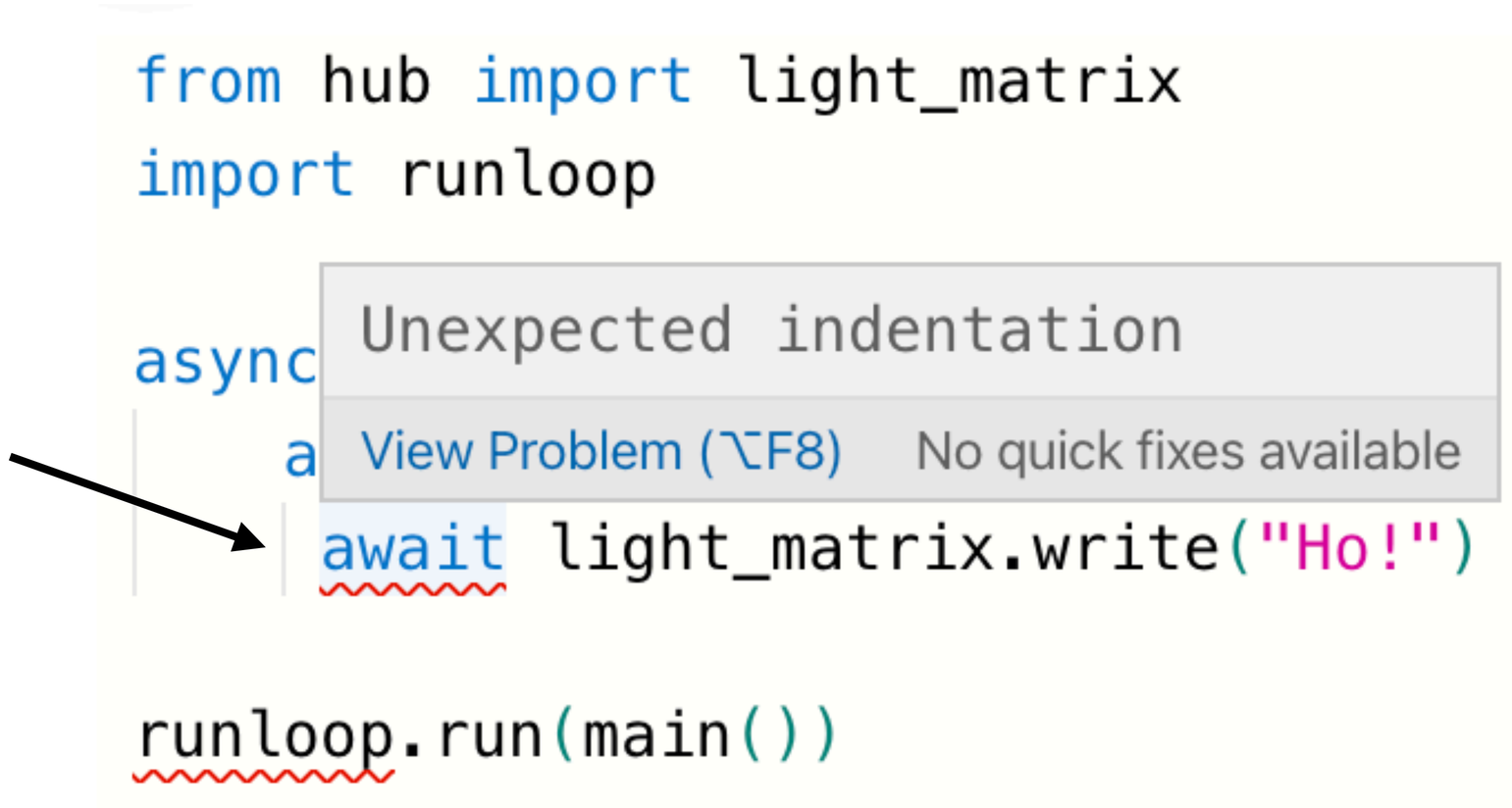
- Python groups code together by indentation
- What happens if you don’t indent correctly?

```
from hub import light_matrix
import runloop

async def main():
    await light_matrix.write("Hi!")
     await light_matrix.write("Ho!")

runloop.run(main())
```

# Python is Picky

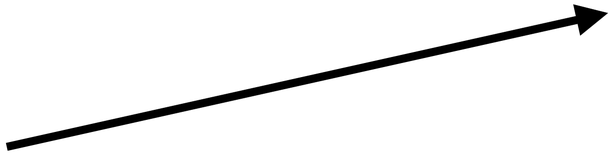
**Indentation**

- The editor will catch some mistakes and let you know when it's confused



# Python is Picky

```
from hub import light_matrix
import runloop

async def main():
    # Have light matrix write "Hi”
    await light_matrix.write("Hi!")

runloop.run(Main())
```



**Capitalization**

- Except for comments, capitalization matters!
- Python is a *case-sensitive* language
- Will this call to *Main()* work?

# Python is Picky

```
from hub import light_matrix
import runloop

async def ma
    await li
    await li

runloop.run(Main())
```

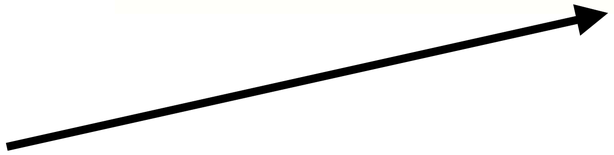
Main: Unknown

"Main" is not defined

View Problem (⌥F8)  No quick fixes available

**Capitalization**

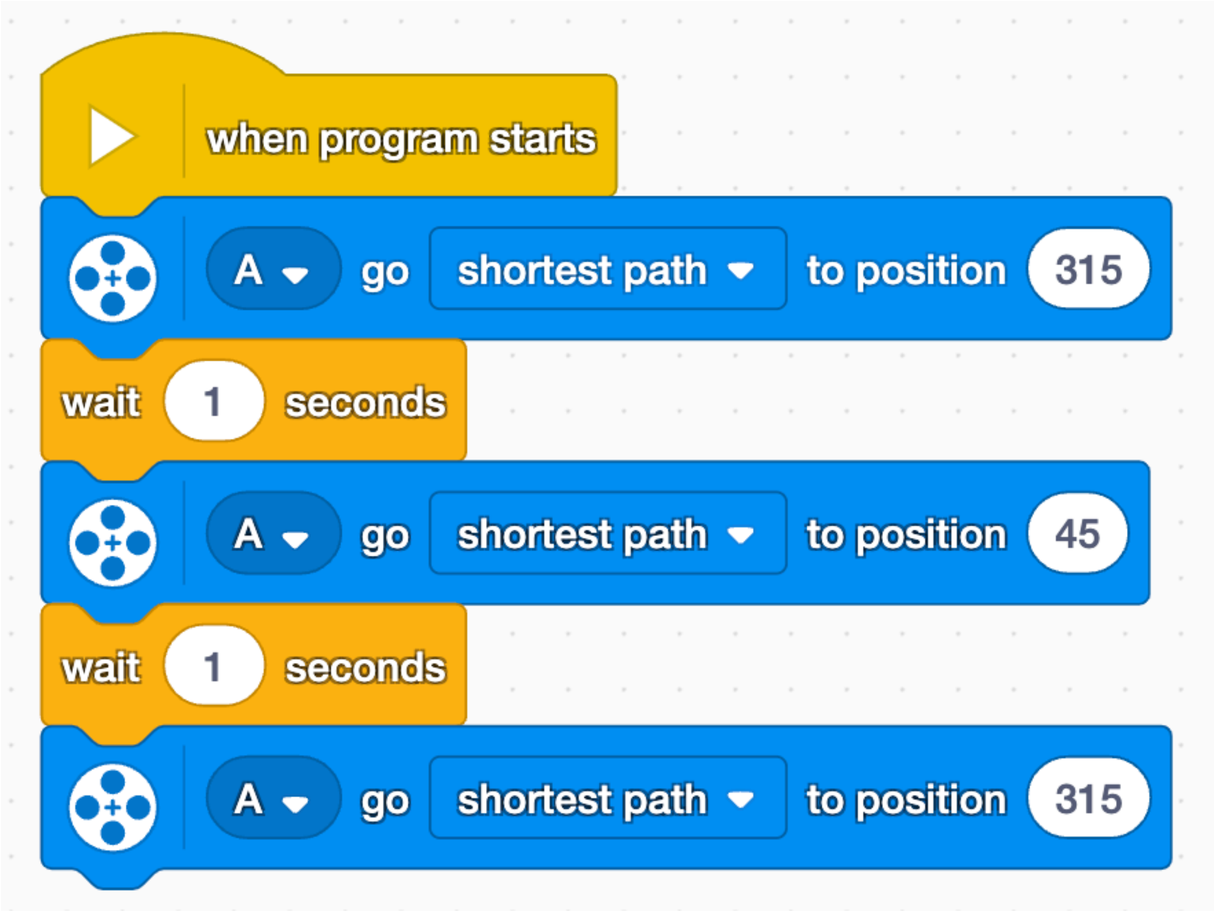
- Will this call to *Main()* work?
- Nope, *main()* is different from *Main()*



# Putting It All Together

```
from hub import light_matrix
import runloop

async def greet(name):
    await light_matrix.write("Hello " + name)

async def main():
    await greet("LEGO beginner")
    await greet("LEGO master")

runloop.run(main())
```

# Finally, Motors!




← Let’s write this in Python

# Motors

## New Imports

- *motor* - motor control module
- *port* - port definitions (A-F)

## Code Completion

- *motor* module contains a bunch of values and functions
- Access the values and functions inside the *motor* module by typing the name after *motor.* (note the period)
- The editor will show you what's inside *motor* after the period

```
1  from hub import port
2  import motor
3  import runloop
4
5  async def main():
6      motor.
7
8  runloop.ru
9
10
```

```
run_for_degrees
run_for_time
run_to_absolute_position
run_to_relative_position
```

# Motors

```
motor.run_to_absolute_position()

runloop.run(main())
```

```
(port: int, position: int, velocity: int, *,
direction: int = motor.SHORTEST_PATH, stop: int =
BRAKE, acceleration: int = 1000, deceleration: int =
1000) -> Awaitable[Unknown]
```

port: int`: A port from the `port` submodule in the `hub` module

Turn a motor to an absolute position. When awaited returns a status of the movement that corresponds to one of the following constants:

`motor.READY` `motor.RUNNING` `motor.STALLED` `motor.CANCELED` `motor.ERROR` `motor.DISCONNECTED`

## Run to Absolute Position

- Let's try *motor.run_to_absolute_position()*
- Code completion will show the required parameters
- Or you can refer to the *Help* panel

# Motors

## Run to Absolute Position

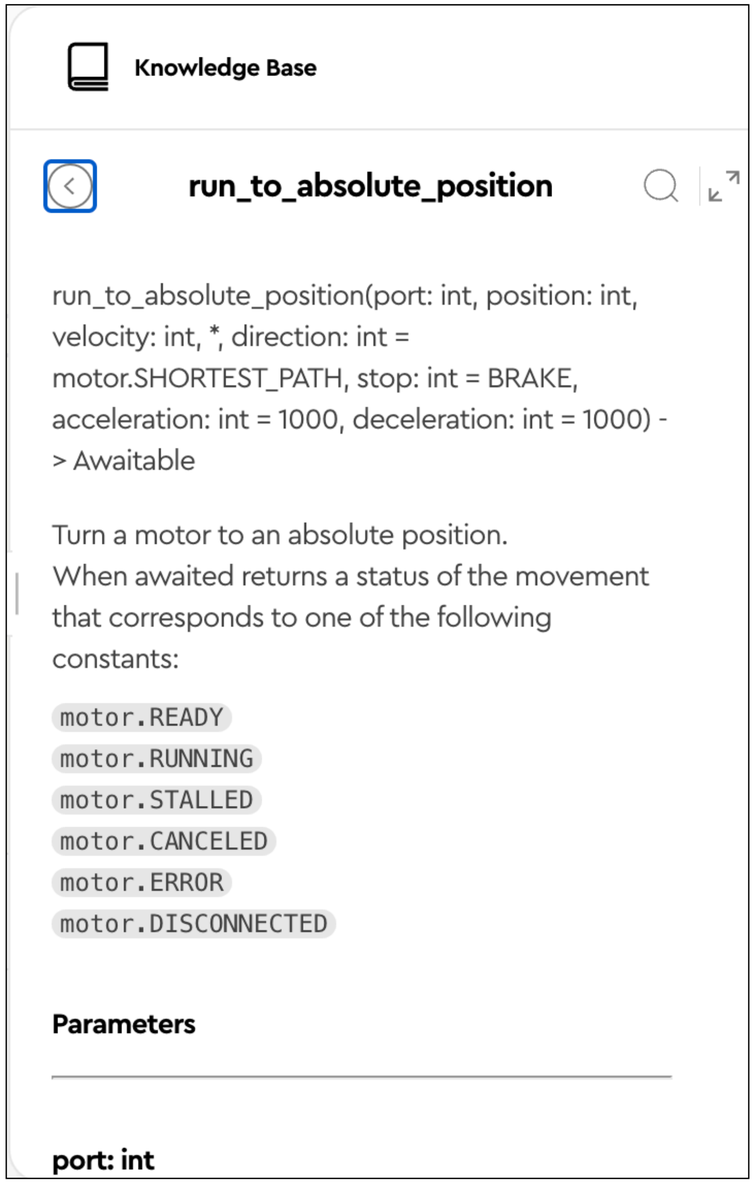
- In the *Help* panel:
  API Modules > Motor > run_to_absolute_position

```
await motor.run_to_absolute_position(port.A, 45, 300)
```

port (A-F)

position (degrees)

speed (-1110 to 1110)



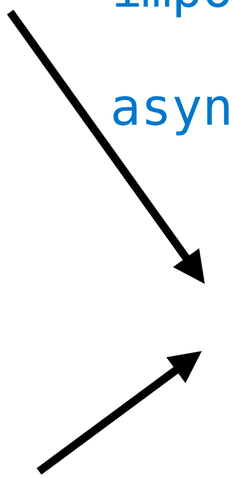
# Motors

**Await**

*await* needs to be here otherwise the program will not wait until the motor stops before moving on

**Sleep in Milliseconds**

sleep_ms will cause your program to pause on this line for this number of milliseconds



```
from hub import port
import motor
import runloop

async def main():
    await motor.run_to_absolute_position(port.A, 315, 300)
    await runloop.sleep_ms(1000)
    await motor.run_to_absolute_position(port.A, 45, 300)
    await runloop.sleep_ms(1000)
    await motor.run_to_absolute_position(port.A, 315, 300)

runloop.run(main())
```

# Motors


```python
from hub import port
import motor
import runloop

async def main():
    await motor.run_to_absolute_position(port.A, 315, 300)
    await runloop.sleep_ms(1000)
    await motor.run_to_absolute_position(port.A, 45, 300)
    await runloop.sleep_ms(1000)
    await motor.run_to_absolute_position(port.A, 315, 300)

runloop.run(main())
```

# Motors

```
from hub import port
import motor
import runloop

async def main():
    await motor.run_to_absolute_position(port.A, 315, 300)
    await runloop.sleep_ms(1000)
    await motor.run_to_absolute_position(port.A, 45, 300)
    await runloop.sleep_ms(1000)
    await motor.run_to_absolute_position(port.A, 315, 300)

runloop.run(main())
```

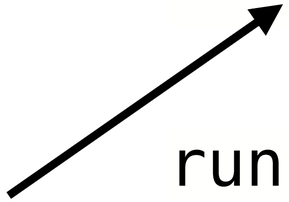
# print() is Helpful

**Program Execution**

- Where does your program run?
- Write program on your computer, runs on LEGO controller
- Helpful to know what your program is doing from your computer

**print()**

- Use *print()* to send text (called a *string* in Python) from the LEGO controller back to your computer



```
from hub import light_matrix
import runloop

async def main():
    await light_matrix.write("Hi!")
    print("I wrote Hi!")

runloop.run(main())
```

# print() is Helpful

```
1 from hub import light_matrix
2 import runloop
3
4 async def main():
5     await light_matrix.write("Hi!")
6     print("I wrote Hi!")
7
8 runloop.run(main())
9
```

Console

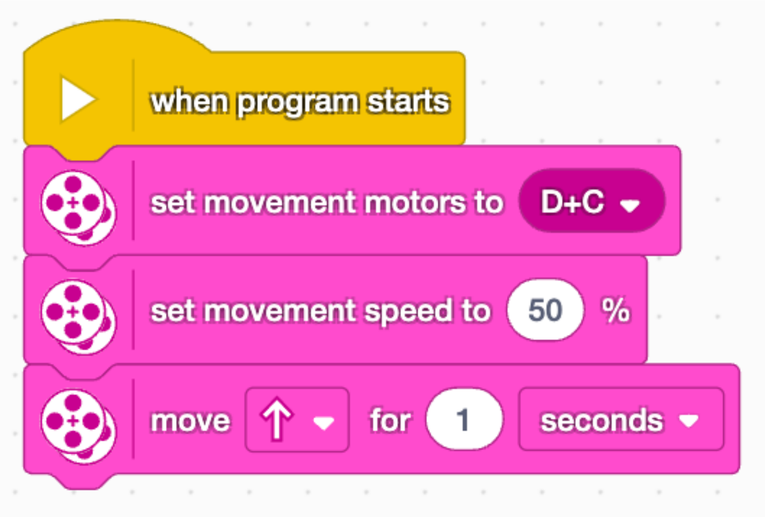
5:23:50 PM: Compiled

I wrote Hi!

**print()**

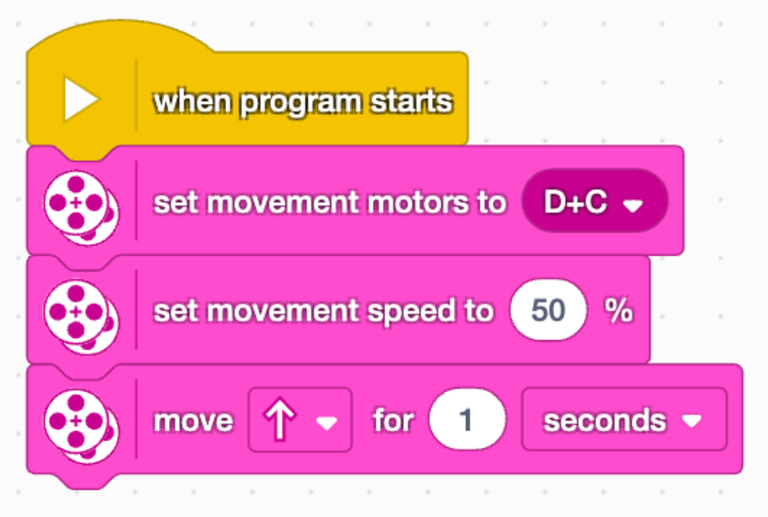
- Use *print()* to send text from the LEGO controller back to your computer

# Movement




```
import motor_pair
import runloop

async def main():
    motor_pair.pair(motor_pair.PAIR_1, port.D, port.C)
    motor_pair.move_tank_for_time(motor_pair.PAIR_1, 550, 550, 1000)

runloop.run(main())
```

# Movement



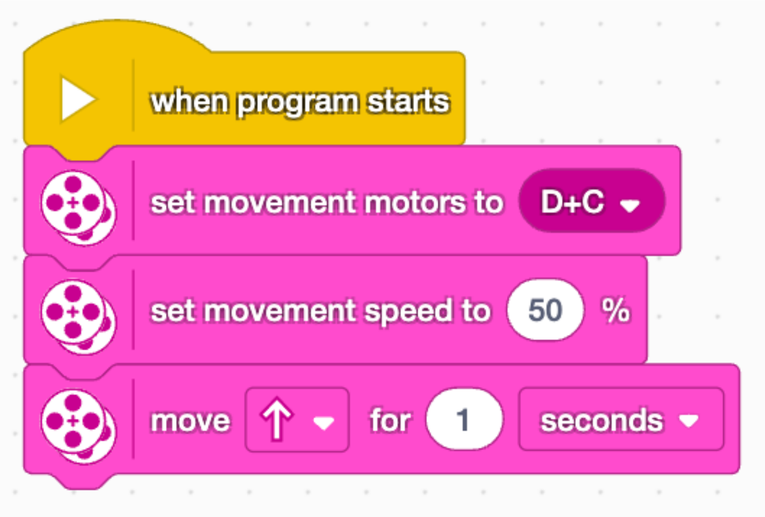

motor pair name

left motor

right motor

```
import motor_pair
import runloop

async def main():
    motor_pair.pair(motor_pair.PAIR_1, port.D, port.C)
    motor_pair.move_tank_for_time(motor_pair.PAIR_1, 550, 550, 1000)

runloop.run(main())
```

# Movement




```
import motor_pair
import runloop

async def main():
    motor_pair.pair(motor_pair.PAIR_1, port.D, port.C)
    motor_pair.move_tank_for_time(motor_pair.PAIR_1, 550, 550, 1000)

runloop.run(main())
```

right motor speed
(-1100 to 1100)

time in milliseconds
(1 sec = 1000 ms)

motor pair name

left motor speed
(-1100 to 1100)