

Explore Python

with LEGO® Education SPIKE™ Prime

Table of Contents



Help!	2	Brain Game.....	18
Hopper Race.....	3	The Coach	19
Super Cleanup	4	Training Camp 1	20
Broken.....	5	Training Camp 2	21
Design for Someone.....	6	Training Camp 3	22
Place your Order	7	Advanced Driving Base	23
Out of Order.....	8	My Code, Our Program	24
Track your Packages	9	Time for an Upgrade	25
Keep it Safe.....	10	Mission Ready	26
Keep it really safe!.....	11	Pass the Brick	27
Automate it!.....	12	Ideas, the LEGO way!	28
Break Dance	13	What is this?	29
Repeat 5 Times	14	Going the Distance	30
Rain or shine?	15	Goal!	31
Wind Speed	16		
Veggie Love	17		

Help!

<https://education.lego.com/en-us/lessons/prime-invention-squad/help>



```
from hub import port, button
from app import sound, display
import runloop
import color_sensor
import color

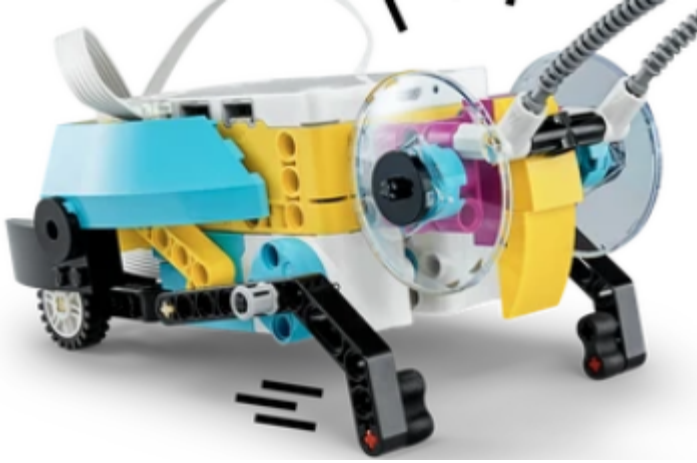
async def main():
    # This is story #1: Kiki is going for a walk. She's outside and happy, until...
    await runloop.until(lambda: button.pressed(button.LEFT) > 0) #1
    await runloop.until(lambda: color_sensor.color(port.B) is color.BLUE)
    await sound.play('Traffic') #2
    await runloop.until(lambda: color_sensor.color(port.B) is color.YELLOW)
    await sound.play('Ring Tone')
    await runloop.until(lambda: color_sensor.color(port.B) is color.GREEN)
    await sound.play('Dog Bark 1')
    await sound.play('Dog Bark 1')
    # This is story #2.
    await runloop.until(lambda: button.pressed(button.RIGHT) > 0)
    await runloop.until(lambda: color_sensor.color(port.B) is color.BLUE)
    await sound.play('Door Knock')
    await runloop.until(lambda: color_sensor.color(port.B) is color.YELLOW)
    await sound.play('Glass Breaking')
    await runloop.until(lambda: color_sensor.color(port.B) is color.GREEN)
    await sound.play('Dog Bark 3')
runloop.run(main())
```

#1 pressed() returns int not boolean

#2 slanted quotes all replaced with '

Hopper Race

<https://education.lego.com/en-us/lessons/prime-invention-squad/hopper-race>

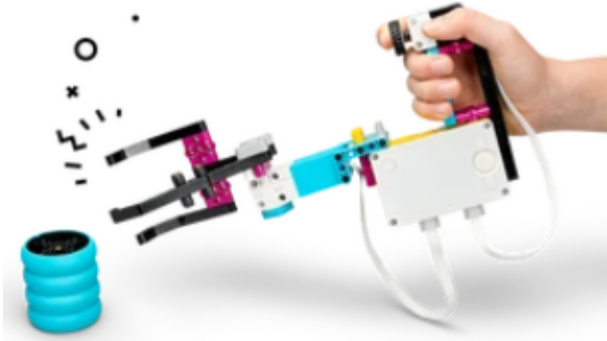


```
from hub import light_matrix, port
import runloop
import motor_pair
async def main():
    # Pair your motors on port E and F.
    motor_pair.pair(motor_pair.PAIR_1, port.E, port.F)
    # Count down from three.
    await light_matrix.write('3') #1
    await runloop.sleep_ms(1000)
    await light_matrix.write('2')
    await runloop.sleep_ms(1000)
    await light_matrix.write('1')
    await runloop.sleep_ms(1000)
    light_matrix.clear()
    # Adjust this to change the distance your Hopper will move.
    # -----v
    motor_pair.move_for_time(motor_pair.PAIR_1, duration=10000, steering=0,
        velocity=500) #2
runloop.run(main())

#1 replaced all slanted quotes with '
#2 added parameter names for clarity
```

Super Cleanup

<https://education.lego.com/en-us/lessons/prime-invention-squad/super-cleanup>



```
from hub import port
import runloop
import motor
import force_sensor

async def main():
    while True:
        # Wait for the force sensor to be pressed.
        await runloop.until(lambda: force_sensor.pressed(port.E))
        motor.run(port.A, velocity=-750) #1 note run is async
        # Wait for the force sensor to be released.
        await runloop.until(lambda: not force_sensor.pressed(port.E))
        motor.run(port.A, velocity=750)
runloop.run(main())

#1 added parameter name for clarity
```

Broken (fix the code)

<https://education.lego.com/en-us/lessons/prime-invention-squad/broken>

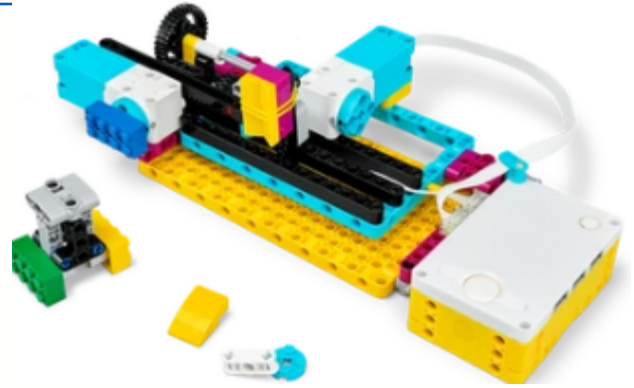
```
from hub import port, button
import runloop
import motor

# X motor is in port A.
# Y motor is in port C.

async def main():
    # Wait for the left button to be pressed.
    await runloop.until(lambda: button.pressed(button.LEFT) > 0) #1
    await motor.run_for_time(port.A, 1500, -1000)
    await runloop.sleep_ms(1000)
    # This block of code should 'cut' a square.
    await motor.run_for_degrees(port.A, 400, 1000)
    await motor.run_for_degrees(port.C, 575, 1000)
    await motor.run_for_degrees(port.A, -400, 1000)
    await motor.run_for_degrees(port.C, -575, 1000)
    # Wait for the right button to be pressed.
    await runloop.until(lambda: button.pressed(button.RIGHT) > 0)
    await motor.run_for_time(port.A, 1500, 1000)
    await runloop.sleep_ms(1000)
    # This block of code should 'cut' a rectangle.
    await motor.run_for_degrees(port.A, -60, 1000)
    await motor.run_for_degrees(port.A, -400, 1000)
    await motor.run_for_degrees(port.C, -800, 1000)
    await motor.run_for_degrees(port.A, 400, 1000)
    await motor.run_for_degrees(port.C, 800, 1000)

runloop.run(main())
```

#1 pressed is an int function, not boolean



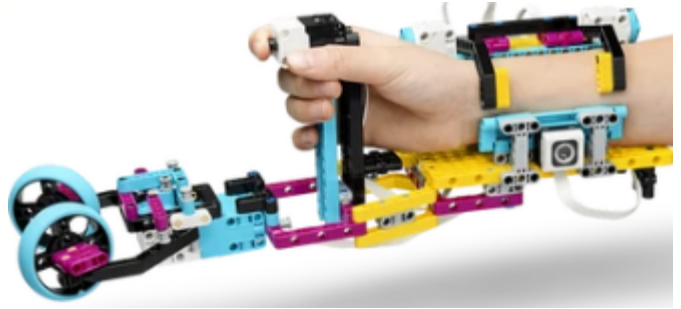
Design for Someone

<https://education.lego.com/en-us/lessons/prime-invention-squad/design-for-someone>

```
from hub import light_matrix, port, sound, button
import runloop
import motor
import force_sensor

async def main():
    await motor.run_to_absolute_position(port.A, 0, 750)
    await motor.run_to_absolute_position(port.E, 0, 750)
    await sound.beep(262, 500)
    await sound.beep(523, 500)
    # Make the prosthesis grab onto someone's arm.
    await motor.run_for_time(port.A, 1000, 750, stop=motor.HOLD)
    await motor.run_for_time(port.E, 1000, -750, stop=motor.HOLD)
    while True:
        if button.pressed(button.RIGHT): #1
            # Make the prosthesis let go.
            await motor.run_to_absolute_position(port.A, 0, 750)
            await motor.run_to_absolute_position(port.E, 0, 750)
            break
        if force_sensor.force(port.B) > 50:
            light_matrix.show_image(light_matrix.IMAGE_SQUARE)
        # You can add code to make a gripper open and close here.
        else:
            light_matrix.clear()
    runloop.sleep_ms(10)
    runloop.run(main())

#1 fixed missing indents
```



Place your Order

<https://education.lego.com/en-us/lessons/prime-kickstart-a-business/place-your-order>

```

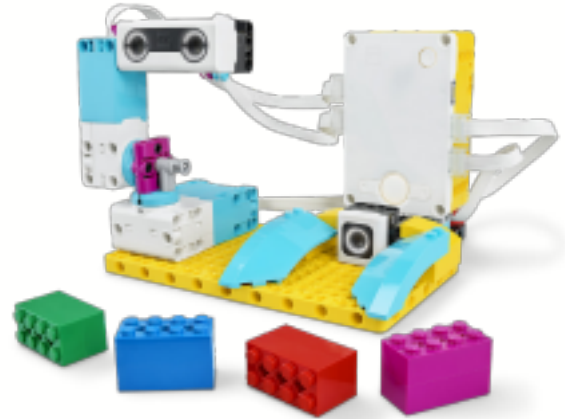
from hub import light_matrix, port
from app import sound
import runloop
import motor
import color_sensor
import color
import distance_sensor
async def main():
    # Arm motor is in port A.
    # Base motor is in port F.
    await motor.run_to_absolute_position(port.A, 350, 500)
    await motor.run_to_absolute_position(port.F, 350, 500)
    sound.play('Connect') #1
    pixels = [100] * 4
    distance_sensor.show(port.C, pixels)
    for x in range(5): #2
        light_matrix.show_image(light_matrix.IMAGE_HEART)
        await runloop.sleep_ms(500)
        light_matrix.show_image(light_matrix.IMAGE_HEART_SMALL)
        await runloop.sleep_ms(500)

    light_matrix.show_image(light_matrix.IMAGE_HEART)
    while True:
        # Wait for the color sensor to see magenta.
        light_matrix.show_image(light_matrix.IMAGE_HEART) #3
        await runloop.until(lambda: color_sensor.color(port.D) is color.MAGENTA)
        sound.play('Connect') #3
        light_matrix.show_image(light_matrix.IMAGE_HAPPY) #3
        await motor.run_for_degrees(port.A, 30, 500)
        await motor.run_for_degrees(port.A, -60, 500)
        await motor.run_for_degrees(port.A, 60, 500)
        await motor.run_for_degrees(port.A, -30, 500)

runloop.run(main())

#1 replace slanted quote with '
#2 10 was too many
#3 change sound and Happy image display to magenta detect

```



Out of Order

<https://education.lego.com/en-us/lessons/prime-kickstart-a-business/out-of-order>

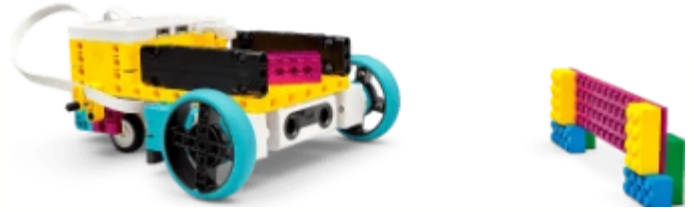
```
from app import display
from hub import port, button
import runloop
import motor
import motor_pair
import distance_sensor

# Back motor is in port C.
# Pair motors with left in port A and right in port E
motor_pair.pair(motor_pair.PAIR_1, port.A, port.E)

async def main():
    # Wait until the left button is pressed.
    await runloop.until(lambda: button.pressed(button.LEFT) > 0) #1
    await motor.run_to_absolute_position(port.C, 0, 1000)
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=500)
    # Adjust the distance in mm here -----v
    await runloop.until(lambda: distance_sensor.distance(port.B) < 150)
    motor_pair.stop(motor_pair.PAIR_1)
    # Wait until the right button is pressed.
    await runloop.until(lambda: button.pressed(button.RIGHT) > 0)
    await motor.run_to_absolute_position(port.C, 0, 1000)
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=500)
    # Adjust the distance in mm here -----v
    await runloop.until(lambda: distance_sensor.distance(port.B) < 150)
    motor_pair.stop(motor_pair.PAIR_1)
    # Adjust the position of the turn here -----v
    await motor.run_to_absolute_position(port.C, 20, 1000)
    await runloop.sleep_ms(1000)
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, -1028, 0, velocity=500)
    await motor.run_to_absolute_position(port.C, 0, 1000)
    runloop.sleep_ms(1000)
    # Adjust the distance in degrees here -----v
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 1028, 0, velocity=500)

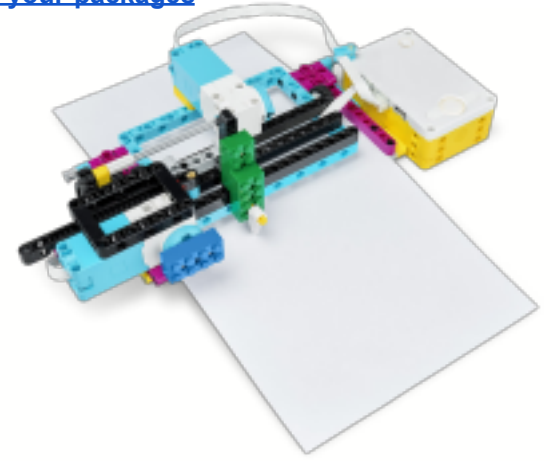
runloop.run(main())

#1 pressed is an int function, not boolean
```



Track your Packages

<https://education.lego.com/en-us/lessons/prime-kickstart-a-business/track-your-packages>



```
from hub import port, button
import runloop
import motor

# The horizontal motor is in port A.
# The vertical motor is in port C.

async def main():
    await runloop.until(lambda: button.pressed(button.LEFT) > 0) #1
    await motor.run_for_time(port.A, 1000, 750)
    await runloop.sleep_ms(1000)
    # This will track your package on map #1
    await motor.run_for_degrees(port.C, 475, 750)
    await motor.run_for_degrees(port.A, -545, 750)
    await motor.run_for_degrees(port.C, 950, 750)
    await motor.run_for_degrees(port.A, 550, 750)
    await motor.run_for_degrees(port.C, 380, 750)
    # Run both motors at the same time to move diagonally
    motor.run(port.C, 750)
    await motor.run_for_degrees(port.A, -540, 500)
    motor.stop(port.C)
    await motor.run_for_degrees(port.C, 175, 750)

runloop.run(main())

#1 pressed is an int function, not boolean
```

Keep it Safe

<https://education.lego.com/en-us/lessons/prime-kickstart-a-business/keep-it-safe>

```
from hub import port, sound, light_matrix, button
import motor, runloop

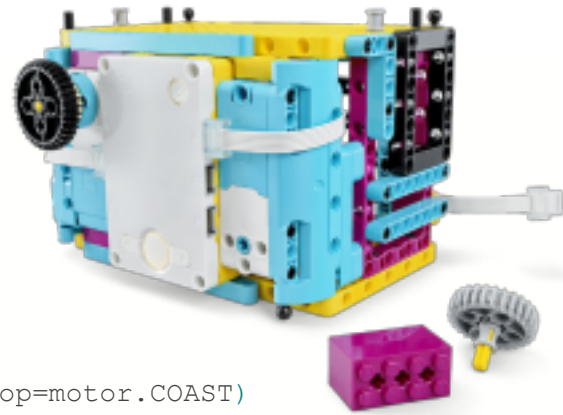
async def main():
    await sound.beep(262, 500)
    await sound.beep(523, 500)

    # This locks the safe.
    await motor.run_to_absolute_position(port.C, 270, 500)
    await motor.run_to_absolute_position(port.B, 0, 750, stop=motor.COAST)
    motor.reset_relative_position(port.B, 0)
    light_matrix.show_image(light_matrix.IMAGE_NO)

    # This unlocks the safe.
    await runloop.until(lambda: button.pressed(button.LEFT) > 0) #1
    await sound.beep(523, 500)
    await runloop.until(lambda: motor.relative_position(port.B) > 180)
    await sound.beep(262, 500)
    await motor.run_for_time(port.C, 1000, 500)
    light_matrix.show_image(light_matrix.IMAGE_NO)
    await runloop.sleep_ms(2000)
    light_matrix.show_image(light_matrix.IMAGE_YES)
    await runloop.sleep_ms(5000)

runloop.run(main())
```

#1 pressed is an int function, not boolean



Keep it really safe!

<https://education.lego.com/en-us/lessons/prime-kickstart-a-business/keep-it-really-safe>

```

from hub import port, light_matrix, button, sound
import runloop
import app
import motor
import time

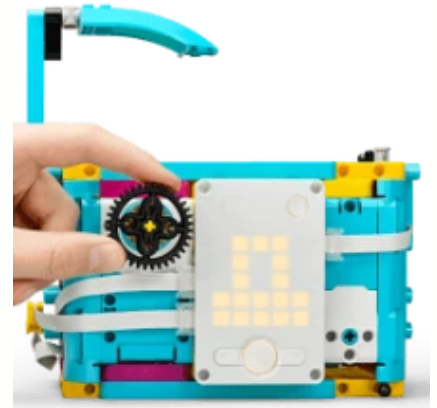
# The lock motor is in port C. The dial motor is in port B. The dial cover motor is in port E.
# This function unlocks the safe and is an async function to work with the async main function.
async def unlock():
    start_time = time.ticks_ms()
    # While the left button is not pressed or the dial motor is less than 180 degrees,
    # Move the dial cover closer to the dial.
    while not button.pressed(button.LEFT) or motor.relative_position(port.B) < 180:
        await sound.beep(262, 200)
        await motor.run_for_degrees(port.E, 15, 500)
        await runloop.sleep_ms(800)
        if time.ticks_diff(time.ticks_ms(), start_time) > 5000:
            await app.sound.play('Bonk') #1 If the time gets to five seconds, play the bonk sound.
            return
    # If the left button is pressed and the dial is greater than 180 degrees, open the safe.
    light_matrix.show_image(light_matrix.IMAGE_YES)
    await motor.run_to_absolute_position(port.E, 0, 500)
    await motor.run_for_time(port.C, 1000, 500)
    await app.sound.play('Wand')

async def main():
    await sound.beep(262, 200)
    await sound.beep(523, 200)
    # This locks the safe.
    await motor.run_for_time(port.C, 1000, -500)
    await motor.run_to_absolute_position(port.B, 0, 500, stop=motor.COAST)
    motor.reset_relative_position(port.B, 0)
    await motor.run_to_absolute_position(port.E, 0, 500)
    light_matrix.show_image(light_matrix.IMAGE_NO)
    await unlock() # This unlocks the safe by calling the unlock function.

runloop.run(main())

#1 change slanted quote to '

```



Automate it!

<https://education.lego.com/en-us/lessons/prime-kickstart-a-business/automate-it>

```
from hub import port
from app import sound #2
import runloop
import motor
import color_sensor
import color

# The base motor is in port A. The arm motor is in port F. The color sensor is in port D.
async def check_color(): # This will check the color of the package.
    await motor.run_to_absolute_position(port.F, 235, 250)
    await runloop.sleep_ms(4000)
    if color_sensor.color(port.D) is color.MAGENTA: # If magenta, place the bricks in the bin.
        await motor.run_to_absolute_position(port.A, 0, 250)
        await motor.run_to_absolute_position(port.F, 25, 250)
        await sound.play('Triumph') #1 2
        await motor.run_to_absolute_position(port.F, 240, 250)
    else: # If it does not sense magenta it will drop the bricks.
        await sound.play('Oops')
        await motor.run_to_absolute_position(port.F, 25, 250)
        for x in range(3):
            await motor.run_for_degrees(port.F, -100, 1000)
            await motor.run_for_degrees(port.F, 100, 1000)
async def main(): # This powers up the robot and makes it grab one package from each side
    await motor.run_to_absolute_position(port.A, 0, 250)
    await motor.run_to_absolute_position(port.F, 240, 250)
    await motor.run_to_absolute_position(port.A, 90, 250)
    await motor.run_to_absolute_position(port.F, 25, 250)
    # This calls the check_color function await check_color()
    await motor.run_to_absolute_position(port.A, 0, 250)
    await motor.run_to_absolute_position(port.F, 240, 250)
    await motor.run_to_absolute_position(port.A, 270, 250)
    await motor.run_to_absolute_position(port.F, 25, 250)
    await check_color()
    await motor.run_to_absolute_position(port.A, 0, 250)
    await motor.run_to_absolute_position(port.F, 240, 250)
runloop.run(main())

#1 replace slanted quote with '
#2 import sound from app
```



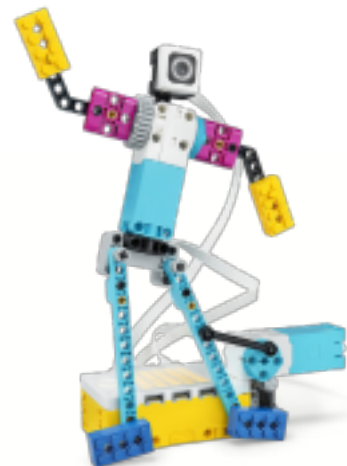
Break Dance

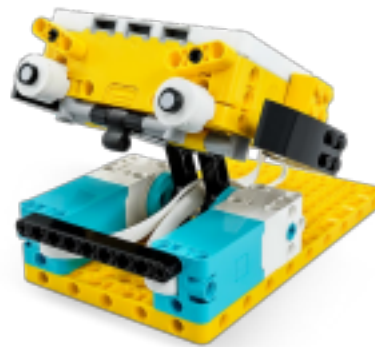
<https://education.lego.com/en-us/lessons/prime-life-hacks/break-dance>

```
from hub import light_matrix, port
import runloop
import motor

# The arm motor is in port B.
# The leg motor is in port F.
async def main():
    # Move arms and legs to 0 position.
    motor.run_to_absolute_position(port.B, 0, -800)
    await motor.run_to_absolute_position(port.F, 0, -800)
    await runloop.sleep_ms(1000)
    for x in range(10):
        await light_matrix.write('1') #1
    # Move arms and legs at the same time.
    motor.run_for_degrees(port.F, 360, -800)
    await motor.run_for_degrees(port.B, 360, -800)
    await runloop.sleep_ms(450)
    await light_matrix.write('2')
    # Move arms and legs at the same time.
    motor.run_for_degrees(port.F, 360, -800)
    await motor.run_for_degrees(port.B, 360, -800)
    await runloop.sleep_ms(450)
    await light_matrix.write('3')
    # Move arms and legs at the same time.
    motor.run_for_degrees(port.F, 360, -800)
    await motor.run_for_degrees(port.B, 360, -800)
    await runloop.sleep_ms(450)
runloop.run(main())
```

#1 replace slanted quote with ' '





Repeat 5 Times

<https://education.lego.com/en-us/lessons/prime-life-hacks/repeat-5-times>

```
from hub import light_matrix, port, motion_sensor
from app import sound
import runloop
import motor_pair

# Leg motors are paired with the left motor in port F and the right motor in port B.
motor_pair.pair(motor_pair.PAIR_1, port.F, port.B)

async def main():
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=-500)
    await runloop.until(lambda: motion_sensor.up_face() is motion_sensor.LEFT)
    motor_pair.stop(motor_pair.PAIR_1)
    await sound.play('Sport Whistle 1') #1
    for count in range(5):
        motor_pair.move(motor_pair.PAIR_1, 0, velocity=500)
        await runloop.until(lambda: motion_sensor.up_face() is motion_sensor.TOP)
        motor_pair.stop(motor_pair.PAIR_1)
        await sound.play('Male Jump 1')
        await light_matrix.write(str(count + 1))
        await runloop.sleep_ms(500)
        motor_pair.move(motor_pair.PAIR_1, 0, velocity=-500)
        await runloop.until(lambda: motion_sensor.up_face() is motion_sensor.LEFT)
        motor_pair.stop(motor_pair.PAIR_1)
        await sound.play('Sport Whistle 2')
    runloop.run(main())

#1 replace slanted quote with `
```

Rain or shine?



<https://education.lego.com/en-us/lessons/prime-life-hacks/rain-or-shine>

Currently, it isn't possible to use the weather forecast functions with our Python programming language.

Here is a program that will make the weather caster move!

```
from hub import light_matrix, port, sound
import app
import runloop
import motor

# The umbrella motor is in port B.
# The sunglasses motor is in port F.
async def main():
    YOUR_LOCAL_FORECAST = 'sunny' #1
    # This gets the robot in the correct starting position.
    await motor.run_to_absolute_position(port.B, 45, 750)
    await motor.run_to_absolute_position(port.F, 300, 750)
    await sound.beep(262, 200)
    await sound.beep(523, 200)
    # If it is sunny, then put on sunglasses.
    if YOUR_LOCAL_FORECAST == 'sunny':
        await motor.run_to_absolute_position(port.F, 0, 750)
        light_matrix.show_image(light_matrix.IMAGE_SQUARE)
        await runloop.sleep_ms(2000)
        await motor.run_to_absolute_position(port.F, 300, 750)
    elif YOUR_LOCAL_FORECAST == 'rainy': # Or if it is rainy, lift the umbrella.
        await motor.run_to_absolute_position(port.B, 340, 750)
        await app.sound.play('Rain')
        await motor.run_to_absolute_position(port.B, 45, 750)
    else:
        # Otherwise show X.
        light_matrix.show_image(light_matrix.IMAGE_NO)
runloop.run(main())
```

Wind Speed

<https://education.lego.com/en-us/lessons/prime-life-hacks/wind-speed>



Currently, it isn't possible to use the weather forecast functions with our Python programming language.

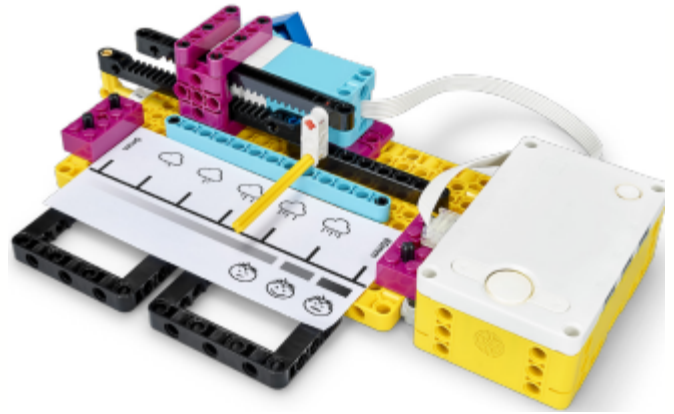
```
from hub import port
import runloop
import motor
# The tilt motor is in port A.
async def main():
    WIND_SPEED_FORECAST = 8
    # This moves the tilt motor into position.
    motor.run_to_absolute_position(port.A, 5, 200)
    # If the wind speed is less than 5.5 move the indicator to the correct position.
    if WIND_SPEED_FORECAST < 5.5:
        await motor.run_for_degrees(port.A, 30, 200)
        await runloop.sleep_ms(1000)
        await motor.run_for_degrees(port.A, -30, 200)
    else:
        await motor.run_for_degrees(port.A, 60, 200)
        await runloop.sleep_ms(1000)
        await motor.run_for_degrees(port.A, -60, 200)
runloop.run(main())
```

Veggie Love

<https://education.lego.com/en-us/lessons/prime-life-hacks/veggie-love>

Currently, it isn't possible to use the weather forecast functions with our Python programming language.

Here is a program that will make the weather caster move!



```
from hub import light_matrix, port, button
import runloop
import motor
# The pointer motor is in port E.
WEEK_RAIN = 50
ROTATION = 0
async def main():
# Wait until the left button is pressed.
    await runloop.until(lambda: button.pressed(button.LEFT) > 0) #1
    await motor.run_for_time(port.E, 2000, -500)
    motor.reset_relative_position(port.E, 0)
    await motor.run_for_time(port.E, 2000, 500)
    await light_matrix.write(str(abs(motor.relative_position(port.E))))

# Calculate the rotation.
    ROTATION = int(WEEK_RAIN * abs(motor.relative_position(port.E)) / 60)
    print(ROTATION)
# Wait until the right button is pressed.
    await runloop.until(lambda: button.pressed(button.RIGHT) > 0)
    motor.reset_relative_position(port.E, 0)
    await motor.run_for_degrees(port.E, ROTATION, -500)

    await light_matrix.write(str(WEEK_RAIN))
    print(WEEK_RAIN)
runloop.run(main())

#1 pressed is an int function, not boolean
```

Brain Game

<https://education.lego.com/en-us/lessons/prime-life-hacks/brain-game/>

```
from hub import light_matrix, button, port
import runloop
import motor
import color_sensor
import color
from app import sound
async def main():
    candy1 = [] #create lists
    candy2 = []
    await runloop.until(lambda: button.pressed(button.LEFT) > 0) #2 eat candy stick 1
    light_matrix.clear()
    candy1.clear()
    await motor.run_for_time(port.A, 2000, -500)
    await sound.play('Bite') #1
    await sound.play('Bite')
    for x in range(5): #read and record sequence of colors in the list called candy 1
        candy1.append(color_sensor.color(port.B))
        await runloop.sleep_ms(1000)
        await motor.run_for_degrees(port.A, 95, 500)
    await runloop.until(lambda: button.pressed(button.RIGHT) > 0) # eat candy stick 2
    candy2.clear()
    await motor.run_for_time(port.A, 2000, -500)
    await sound.play('Bite')
    await sound.play('Bite')
    for x in range(5): #read and record sequence of colors in the list called candy 2
        candy2.append(color_sensor.color(port.B))
        await runloop.sleep_ms(1000)
        await motor.run_for_degrees(port.A, 95, 500)
    candy1_red_index = candy1.index(color.RED) #light up position of red bricks if a match
    candy2_red_index = candy2.index(color.RED)
    if candy1_red_index == candy2_red_index:
        for x in range(5):
            light_matrix.set_pixel(x, candy1_red_index, 100)
            await sound.play('Win')
    else:
        light_matrix.show_image(light_matrix.IMAGE_NO)
        await sound.play('Oops')
runloop.run(main())
```



The Coach

<https://education.lego.com/en-us/lessons/prime-life-hacks/the-coach>

```
from hub import port
import runloop
import motor
import time

# The left leg motor is in port F.
# The right leg motor is in port B.
async def main():
    # Set the coach to stand straight up.
    motor.run_to_absolute_position(port.F, 0, 800)
    motor.run_to_absolute_position(port.B, 0, 800)
    start_time = time.ticks_ms()

    # The coach will dance for five seconds.
    while time.ticks_diff(time.ticks_ms(), start_time) < 5000:
        motor.set_duty_cycle(port.B, -7000)
        motor.set_duty_cycle(port.F, 7000)
        await runloop.sleep_ms(200)
        motor.set_duty_cycle(port.B, 7000)
        motor.set_duty_cycle(port.F, -7000)
        await runloop.sleep_ms(200)

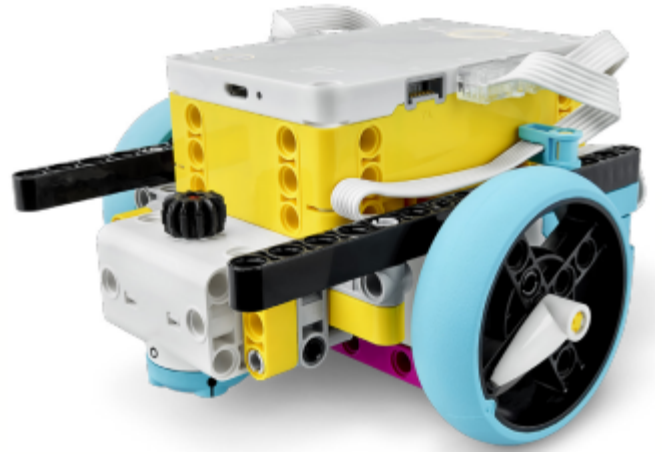
    motor.stop(port.B)
    motor.stop(port.F)

runloop.run(main())
```



Training Camp 1

<https://education.lego.com/en-us/lessons/prime-competition-ready/training-camp-1-driving-around>



```
from hub import port
import runloop
import motor_pair

# The left wheel motor is in port C.
# The right wheel motor is in port D.

motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)

async def main():
    # Loop four times to travel in a square.
    for x in range(4):
        # (Distance / 17.5) * 360 will get you the degrees needed to travel the distance needed.
        # To travel 10 cm, calculate (10 / 17.5 * 360 = 206)
        # Adjust the degrees here -----v
        await motor_pair.move_for_degrees(motor_pair.PAIR_1, 206, 0, velocity=300)
        await runloop.sleep_ms(500)
        # Make a right turn.
        await motor_pair.move_for_degrees(motor_pair.PAIR_1, 182, 100, velocity=300)

runloop.run(main())
```

Training Camp 2

<https://education.lego.com/en-us/lessons/prime-competition-ready/training-camp-2-playing-with-objects>



```

from hub import port, sound, button
import runloop
import motor_pair
import motor
import distance_sensor

# The left wheel motor is in port C.
# The right wheel motor is in port D.
# The arm motor is in port E.
# The distance sensor is in port F.
motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)

async def main():
    # Move the arm motor up and then partially down.
    await motor.run_for_time(port.E, 1000, -200)
    await motor.run_for_degrees(port.E, 75, 200)
    await sound.beep(262, 200)
    await sound.beep(523, 200)

    # Wait for the right button to be pressed.
    await runloop.until(lambda: button.pressed(button.RIGHT) > 0) #2
    await runloop.sleep_ms(1000)

    # Drive base will travel forward until the distance sensor senses closer than 100 mm.
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=300)
    await runloop.until(lambda: distance_sensor.distance(port.F) < 100)
    print(distance_sensor.distance(port.F))
    motor_pair.stop(motor_pair.PAIR_1)

    # Move the arm all the way down.
    await motor.run_for_degrees(port.E, -75, 200)
    await sound.beep(600, 500)
    await sound.beep(720, 500)

    # (Distance / 17.5) * 360 will get you the degrees needed to travel the distance needed.
    # To travel 20 cm, calculate (20 / 17.5 * 360 = 411)
    # To move the drive base in reverse you make the degrees negative
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, -411, 0, velocity=300)

runloop.run(main())

```

Training Camp 3

<https://education.lego.com/en-us/lessons/prime-competition-ready/training-camp-3-react-to-lines>

```
from hub import port, button

import runloop

import motor_pair
import color_sensor
import color

# The left leg motor is in port C.
# The right leg motor is in port D.
# The color sensor is in port B.

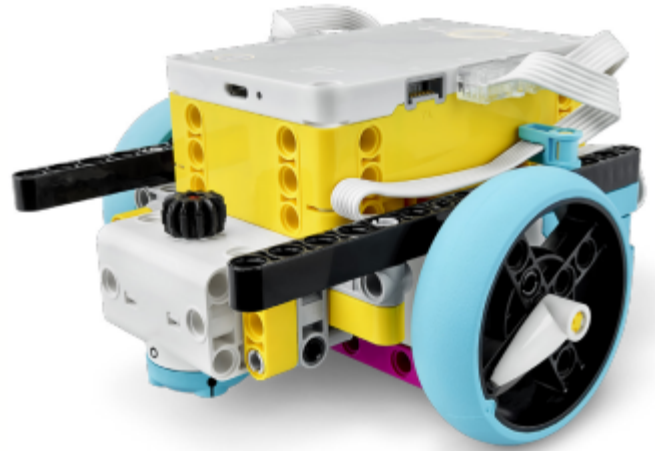
motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)

async def main():
    # Set the velocity.
    velocity_value = 400

    while True:
        # If the left button is pressed, the drive base stops at a black line.
        if button.pressed(button.LEFT):
            motor_pair.move(motor_pair.PAIR_1, 0, velocity=velocity_value)
            await runloop.until(lambda: color_sensor.color(port.B) is color.BLACK)
            motor_pair.stop(motor_pair.PAIR_1)

        # If the right button is pressed, the drive base follows a black line,
        if button.pressed(button.RIGHT):
            while True:
                motor_pair.move_tank(motor_pair.PAIR_1, 0, velocity_value)
                await runloop.until(lambda: color_sensor.color(port.B) is color.BLACK)
                motor_pair.move_tank(motor_pair.PAIR_1, velocity_value, 0)
                await runloop.until(lambda: color_sensor.color(port.B) is color.WHITE)

runloop.run(main())
```



Advanced Driving Base

<https://education.lego.com/en-us/lessons/prime-competition-ready/assembling-an-advanced-driving-base>



```
from hub import port, motion_sensor
import runloop
import motor_pair

# The drive motors are in port A (left) and port E (right).
# The wheel circumference is 27.63 cm.
# Pair the motors.
motor_pair.pair(motor_pair.PAIR_1, port.A, port.E)
# Create a function to make a right turn 90 degrees.
async def right_turn():
    motion_sensor.reset_yaw(0)
    motor_pair.move(motor_pair.PAIR_1, 100, velocity= 50)
    await runloop.until(lambda: motion_sensor.tilt_angles()[0] <= -900)
    motor_pair.stop(motor_pair.PAIR_1)
# Create a function to make a left turn 90 degrees.
async def left_turn():
    motion_sensor.reset_yaw(0)
    motor_pair.move(motor_pair.PAIR_1, 100, velocity= 50)
    await runloop.until(lambda: motion_sensor.tilt_angles()[0] >= 900)
    motor_pair.stop(motor_pair.PAIR_1)
async def main():
    # 20 cm / (circumference / 360) = 260 degrees.
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 260, 0, velocity=100)
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, -260, 0, velocity=100)
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 260, -40, velocity=100)
    await right_turn()
    await runloop.sleep_ms(1000)
    await left_turn()
runloop.run(main())
```

My Code, Our Program

<https://education.lego.com/en-us/lessons/prime-competition-ready/my-code-oi>



```
from hub import port, sound
import runloop
import motor_pair

# The drive motors are in port A (left) and port E (right).
# The wheel circumference is 27.63 cm.
# Pair the motors.
motor_pair.pair(motor_pair.PAIR_1, port.A, port.E)

async def square():
    for x in range(4):
        await motor_pair.move_for_degrees(motor_pair.PAIR_1, 540, 0, velocity= 500)
        await motor_pair.move_for_degrees(motor_pair.PAIR_1, 145, 100, velocity= 500)

async def triangle():
    for x in range(3):
        await motor_pair.move_for_degrees(motor_pair.PAIR_1, 540, 0, velocity= 500)
        await motor_pair.move_for_degrees(motor_pair.PAIR_1, 200, 100, velocity= 500)

async def circle():
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 1500, 40, velocity= 500)

async def main():
    await square()
    await sound.beep(262, 200)
    await triangle()
    await sound.beep(262, 200)
    await circle()
    await sound.beep(262, 200)

runloop.run(main())
```

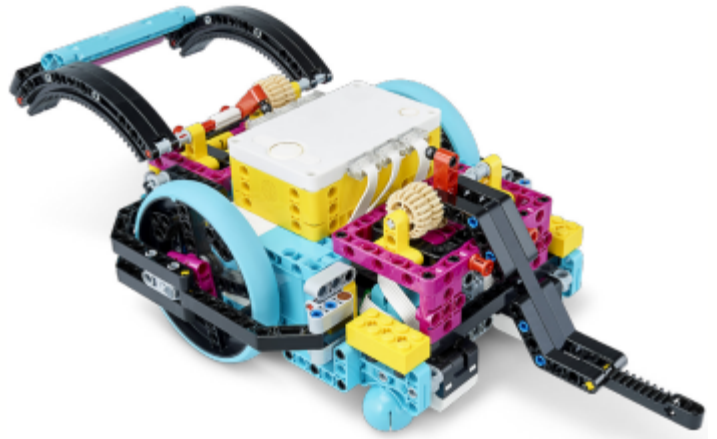
Time for an Upgrade

<https://education.lego.com/en-us/lessons/prime-competition-ready/time-for-an-upgrade>

```
from hub import port, sound
import runloop
import motor

# The lift arm motor is in port D.
# The dozer blade motor is in port C

async def main():
    await motor.run_for_time(port.D, 500, -1000)
    await motor.run_for_time(port.C, 500, -1000)
    await motor.run_for_degrees(port.D, 70, 1000)
    await motor.run_for_degrees(port.C, 70, 1000)
    await sound.beep(262, 200)
    await motor.run_for_degrees(port.D, 180, 1000)
    await motor.run_for_degrees(port.D, -180, 1000)
    await motor.run_for_degrees(port.C, 180, 1000)
    await motor.run_for_degrees(port.C, -180, 1000)
    await sound.beep(262, 200)
    await motor.run_for_degrees(port.D, 180, 150)
    await motor.run_for_degrees(port.D, -180, 150)
    await motor.run_for_degrees(port.C, 180, 150)
    await motor.run_for_degrees(port.C, -180, 150)
runloop.run(main())
```



Mission Ready

<https://education.lego.com/en-us/lessons/prime-competition-ready/mission-ready>

```
from hub import port
```

```
import runloop
```

```
import motor_pair
```

```
import motor
```

```
# The dozer blade motor is in port C.
```

```
# The lift arm motor is in port D.
```

```
# Drive motors are in port A(left) and port E(right).The wheel circumference is 27.63 cm.
```

```
async def main():
```

```
    motor_pair.pair(motor_pair.PAIR_1, port.A, port.E) # Pair the motors.
```

```
    motor.run_for_time(port.C, 1000, -1000) # Prepare the dozer blade and lift arm.
```

```
    await motor.run_for_time(port.D, 1000, -1000)
```

```
    await motor.run_for_degrees(port.C, 70, 1000)
```

```
    await motor.run_for_degrees(port.D, 20, 1000)
```

```
# Move forward to push in the lock bar.
```

```
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, -26, 0, velocity= 250)
```

```
# Back up and lift the dozer blade.
```

```
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 137 , 0, velocity= 250)
```

```
    await motor.run_for_degrees(port.C, 180, 400)
```

```
# Move forward and pull down the lever and then move the dozer blade back up.
```

```
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, -78, 0, velocity= 250)
```

```
    await motor.run_for_degrees(port.C, -180, 600)
```

```
    await motor.run_for_degrees(port.C, 180, 600)
```

```
# Back up and lower the dozer blade.
```

```
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 91, 0, velocity= 250)
```

```
    await motor.run_for_degrees(port.C, -180, 600)
```

```
# Turn to drive around the mission.
```

```
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 146, -100, velocity= 250)
```

```
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 100, 0, velocity= 250)
```

```
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 600, -25, velocity= 250)
```

```
# Drive to the other side of the mission.
```

```
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 1250, 0, velocity= 250)
```

```
# Turn and drive forward to align to the gates.
```

```
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 300, -50, velocity= 250)
```

```
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 228, 0, velocity= 250)
```

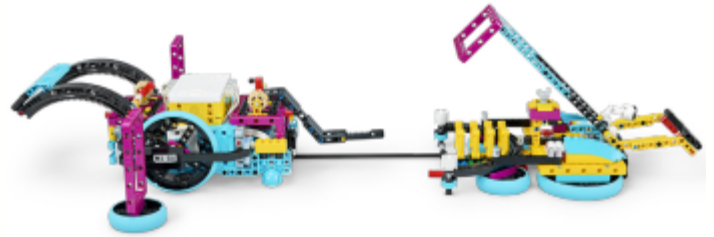
```
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 140, -100, velocity= 250)
```

```
# Drive forward to place the lift arm through the gates.
```

```
    await motor_pair.move_for_degrees(motor_pair.PAIR_1, 450, 0, velocity= 250)
```

```
    await motor.run_for_degrees(port.D, 100, 1000) # Lift the arm to get the gates.
```

```
runloop.run(main())
```



Pass the Brick

<https://education.lego.com/en-us/lessons/prime-extra-resources/pass-the-brick>

```
from hub import port, button
import runloop
import motor

async def main():
    # This will make the hand open once to start.
    motor.run_for_time(port.F, 1000, 750)
    while True:
        # Wait for the left button to be pressed.
        await runloop.until(lambda: button.pressed(button.LEFT) > 0) #1
        motor.run(port.F, -750)
        # Wait for the left button to be released.
        await runloop.until(lambda: button.pressed(button.LEFT) <= 0)
        motor.run(port.F, 750)
runloop.run(main())
```



Ideas, the LEGO way!

<https://education.lego.com/en-us/lessons/prime-extra-resources/ideas-the-lego-way>



```
from hub import light_matrix, button, sound
import runloop

async def main():
    while True:
```

```
# If the left button is pressed, count down 3, 2, 1... one second at a time.
```

```
    if button.pressed(button.LEFT):
        await light_matrix.write('3')
        await runloop.sleep_ms(1000)
        await light_matrix.write('2')
        await runloop.sleep_ms(1000)
        await light_matrix.write('1')
        await runloop.sleep_ms(1000)
        light_matrix.clear()
        await sound.beep(262, 500)
        await sound.beep(523, 500)
```

```
# If the right button is pressed, count down 5, 4, 3, 2, 1... six seconds at a time.
```

```
    if button.pressed(button.RIGHT):
        await light_matrix.write('5')
        await runloop.sleep_ms(6000)
        await light_matrix.write('4')
        await runloop.sleep_ms(6000)
        await light_matrix.write('3')
        await runloop.sleep_ms(6000)
        await light_matrix.write('2')
        await runloop.sleep_ms(6000)
        await light_matrix.write('1')
        await runloop.sleep_ms(6000)
        light_matrix.clear()
        await sound.beep(262, 500)
        await sound.beep(523, 500)
```

```
runloop.run(main())
```

What is this?

<https://education.lego.com/en-us/lessons/prime-extra-resources/what-is-this>

```
from hub import port
import runloop
import motor

async def main():
    # Create a loop that repeats five times.
    for x in range(5):
        # Run the motor for two seconds at 500 degrees per second forward.
        await motor.run_for_time(port.F, 2000, 500)
        # Run the motor for two seconds at 500 degrees per second backward.
        await motor.run_for_time(port.F, 2000, -500)
runloop.run(main())
```



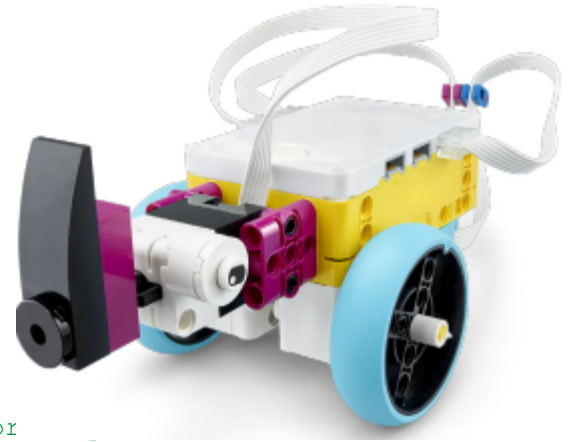
Going the Distance

<https://education.lego.com/en-us/lessons/prime-extra-resources/going-the-distance>

```
from hub import port
import runloop
import motor_pair
import force_sensor

async def main():
    # Pair motors with the left motor on port B and right motor
    motor_pair.pair(motor_pair.PAIR_1, port.B, port.A)
    # Move motor pair straight forward at 500 degrees per second.
    motor_pair.move(motor_pair.PAIR_1, 0, velocity=500)
    # Wait for the force sensor to be pressed.
    await runloop.until(lambda: force_sensor.pressed(port.C))
    # Stop the rhino when the button is pressed.
    motor_pair.stop(motor_pair.PAIR_1)

runloop.run(main())
```



```
from hub import port, button
import runloop
import motor

async def main():
    while True:
        # Set the motor to the 0 degree position.
        motor.run_to_absolute_position(port.A, 0, 1000)
        # Wait for the left button to be pressed to run the motor for 360 degrees.
        await runloop.until(lambda: button.pressed(button.LEFT) > 0)
        await motor.run_for_degrees(port.A, 360, 1000)
        # Wait for one second before being able to press the button again.
        await runloop.sleep_ms(1000)
runloop.run(main())
```

A collection of colorful LEGO bricks, including a large purple frame, a yellow brick, a blue brick, a green brick, and a red brick, scattered on a white background.