HUB BUTTONS (HUB.BUTTONS)

Reads states of the the hub's buttons.

Buttons: CENTER, LEFT, RIGHT, .BLUETOOTH

`hub.buttons.pressed(Button)`

Returns a set of buttons currently pressed (Button.CENTER, Button.LEFT, Button.RIGHT, Button.BLUETOOTH).

from pybricks.parameters import Button

```
from pybricks.hubs import PrimeHub
from pybricks.parameters import Button
hub = PrimeHub()

# Prints a message if the centre button
#is in the set of buttons pressed

if Button.CENTER in hub.buttons.pressed():
    print("Center button pressed!")
```

STATUS/HUB LIGHT (HUB.LIGHT)

Controls the RGB LED status light surrounding the center button.

`hub.light.on(color)`

Turns on the light with a specific color. (Color.RED)

`hub.light.off()`

Turns off the status light.

`hub.light.blink(color, durations)`

Blinks the light indefinitely using a list of on/off duration times in millisecond

`hub.light.animate(colors, interval)`

Animates the light through a sequence of colors in the background.

from pybricks.parameters import Color

```
from pybricks.hubs import PrimeHub
from pybricks.parameters import Color
hub = PrimeHub()

# Blink Red on for 500ms, off for 200ms
hub.light.blink(Color.RED, [500, 200])
```

```
from pybricks.hubs import PrimeHub

from pybricks.parameters import Color
hub = PrimeHub()

# Cycle through colors every 300ms
hub.light.animate([Color.RED, Color.GREEN, Color.BLUE], interval=300)
```

HUB INITIALIZATION

Initializes the SPIKE Prime hub.

```
from pybricks.hubs import PrimeHub
hub = PrimeHub()
```

Default Spike Prime hub initialization

```
from pybricks.hubs import PrimeHub
hub = PrimeHub()      #Intialise hub
```

SPEAKER (HUB.SPEAKER)

Generates audio messages and tones.

`hub.speaker.volume(volume)`

Reports or sets speaker volume (0 (None)to 100%).

`hub.speaker.beep(frequency, duration)`

Plays a single tone at a specific frequency (Hz) for a duration in milliseconds.

`hub.speaker.play_notes(notes, tempo)`

Plays a sequence of musical notes string notation.

```
from pybricks.hubs import PrimeHub
from pybricks.tools import wait

hub = PrimeHub()

# Set speaker volume to 80%
hub.speaker.volume(80)

# Play a 1000 Hz beep for 200 milliseconds
hub.speaker.beep(frequency=1000, duration=200)

wait(500)
```

SYSTEM & POWER

Monitors battery & Manages System

`hub.battery.voltage()`

Returns battery voltage in millivolts.

`hub.charger.connected()`

Returns True if a USB charger is plugged in.

`hub.system.shutdown()`

Stops the program and turns off the hub

```
from pybricks.hubs import PrimeHub
hub = PrimeHub()

# Plays C4 for quarter note, E4, G4
hub.speaker.play_notes(["C4/4", "E4/4", "G4/4"], tempo=120
```

ROBOTS GOT TALENTS®

LIGHT MATRIX DISPLAY

Controls the 5x5 pixel grid on the hub.

`hub.display.off()`
Clears the display and turns off all pixels

`hub.display.pixel(R, C, brightness)`
Turns on an individual pixel on the matrix, defined by its row and column (0–4) with a brightness percentage (0–100)

`hub.display.number(number)`
Displays a number from -99 to 99. (-) will appear on the center of the matrix to mark a -ve number

`hub.display.char(char)`
Displays a single charecter (A,1,#) on the matrix.

`hub.display.text(text, on=, off=)`
displays a text string character by character.

`hub.display.icon(list/predef. icon)`
Displays a static 5x5 pixel image on the matrix. Accepts a built-in Pybricks icon or a 5x5 nested list of brightness percentages (0 to 100).

`hub.display.animate(matrices, interval)`
Displays a continuously looping animation from a sequence of 2D matrix images.

```
from pybricks.hubs import PrimeHub
from pybricks.tools import wait
hub = PrimeHub()

# Turn on individual pixels at 80% brightness  (Top-left corner)
hub.display.pixel(0, 0, brightness=80)
wait(1500)

# Turn off all display pixels
hub.display.off()
wait(1000)

# Scroll a word across the LED matrix
# Displays each character for 400ms
#with a 100ms pause between letters
hub.display.text("RGT", on=400, off=100)
```

```
from pybricks.hubs import PrimeHub
from pybricks.tools import wait
hub = PrimeHub()

heart_symbol = [
    [0,  100, 0,  100, 0],
    [100, 100, 100, 100, 100],
    [100, 100, 100, 100, 100],
    [0,  100, 100, 100, 0],
    [0,   0,  100,  0,  0],
]   # Custom 5x5 Symbol (Heart Pattern)

# Display custom icon for 3 seconds
hub.display.icon(heart_symbol)
wait(3000)
```

IMU SENSOR (HUB.IMU)

Interacts with the built-in 6-axis gyro and accelerometer.

`hub.imu.ready()`
Returns True if the IMU is fully calibrated and ready.

`hub.imu.stationary()`
Returns True if the hub has been stationary for > 1s.

`hub.imu.up()`
Detects which side of the hub faces upward (Side.TOP, Side.BOTTOM, Side.LEFT, Side.RIGHT, Side.FRONT, Side.BACK).

`hub.imu.tilt()`
Gets the pitch and roll angles in degrees as a tuple.

`hub.imu.heading()`
Gets the cumulative yaw/heading angle in degrees relative to starting orientation (positive = clockwise).

`hub.imu.reset_heading(angle)`
Resets the heading angle to a specified value.

`hub.imu.rotation(axis)`
Reports the rotation around a specific axis (Axis.X, Axis.Y, or Axis.Z).

`hub.imu.acceleration(axis=)`
Reports the acceleration in mm/s^2 along an axis.

`hub.imu.angular_velocity(axis=)`
Reads rotation speed in deg/s along a specific axis

`hub.imu.orientation()`
Returns the full 3D orientation of the hub as a 3x3 rotation matrix object

```
from pybricks.hubs import PrimeHub
from pybricks.tools import wait
hub = PrimeHub()

# Reset gyro heading to 0 degrees at start
hub.imu.reset_heading(0)

# Check if IMU is stationary and calibrated
print("IMU Set:", hub.imu.ready())

# Read and print orientation metrics 10 times
for _ in range(10):
    side = hub.imu.up()         # Detects facing side
    pitch, roll = hub.imu.tilt()    # Reads pitch and roll in degrees
    heading = hub.imu.heading()   # Reads hub heading (z-axis)
    print("Up:", side, "\nTilt:", pitch, roll, "\nHeading:", heading)
    wait(500)
```

ROBOTS GOT TALENTS®

FORCE SENSOR

Measures the force and monitors the status of the force sensor.

`force_sensor.force()`
Reports the measured force applied to the force sensor in Newtons (0-10N).

`force_sensor.pressed(force=X)`
Returns True if the applied force exceeds the defines threshold in Newtons.

`force_sensor.touched()`
Returns True if the force sensor is pressed.

```
from pybricks.pupdevices import ForceSensor
from pybricks.parameters import Port
from pybricks.tools import wait

sensor = ForceSensor(Port.E)

# Wait until the force sensor is pressed with 2 Newtons

while not sensor.pressed(force=2):
    print("Force:", sensor.force(), "N")
    wait(200)
```

DISTANCE SENSOR

Measures distance to an objects and controls the light rings around the sensor's eyes.

`distance_sensor.distance()`
Returns the distance to the nearest object in (mm)

`distance_sensor.presence()`
Returns True if another ultrasonic sensor is detected nearby transmitting signals.

`distance_sensor.lights.on(brightness)`
Adjust the four LED segments (light rings) around the distance sensor eyes to a specified % (0-100)

`distance_sensor.lights.off()`
Turns off all four LED light segments.

```
from pybricks.pupdevices import UltrasonicSensor
from pybricks.parameters import Port
from pybricks.tools import wait

distance_sensor = UltrasonicSensor(Port.C)

#Adjust the light ring brightness to 80%
distance_sensor.lights.on(80)

#Detects & print distance to nearest object 5 times (1 per s)
for _ in range(5):
    dist_mm = distance_sensor.distance()
    print("Obstacle Distance:", dist_mm, "mm")
    wait(1000)

distance_sensor.lights.off()    #Turn light rings off
```

PUPDEVICES INITIALIZATION

Initializes the Spike Prime motors, & sensors connected to the hub ports.

```
from pybricks.pupdevices import Motor, ColorSensor
from pybricks.parameters import Port
motor = Motor(Port.A)
color_sensor = ColorSensor(Port.B)
```

Default initialization for motor & color sensor

from pybricks.parameters import Port

```
from pybricks.pupdevices import Motor, ColorSensor
from pybricks.pupdevices import UltrasonicSensor, ForceSensor
from pybricks.parameters import Port

motor = Motor(Port.A)   # Motor Initialization

color_sensor = ColorSensor(Port.B)    # Sensors Initialization
distance_sensor = UltrasonicSensor(Port.C)
force_sensor = ForceSensor(Port.D)
```

COLOR SENSOR

Reads surface color, ambient light intensity, and reflected light & controls built-in light

`color_sensor.color()`
Returns the detected color from (Color.RED, Color.BLUE, Color.GREEN, Color.YELLOW, Color.BLACK, Color.WHITE, Color.NONE).

`color_sensor.reflection()`
Returns the reflected light intensity as a percentage (0-100%).

`color_sensor.ambient()`
Returns the ambient light intensity as a percentage.

`color_sensor.hsv()`
Returns the detected color as Hue (0–360), Saturation (0–100), and Value (0–100).

`color_sensor.lights.off()`
Turns off the sensor's built-in light source.

`color_sensor.lights.on(brightness)`
Adjust the sensor's built-in light source to a defined brightness percentage (0-100).

```
from pybricks.pupdevices import ColorSensor
from pybricks.parameters import Port, Color
from pybricks.tools import wait

color_sensor = ColorSensor(Port.B)

# Read the detected colour & reflected light intensity %
detected = color_sensor.color()
reflectance = color_sensor.reflection()
print("Detected Color:", detected)
print("Reflected Light:", reflectance, "%")
```

ROBOTS GOT TALENTS®

MOTOR

Controls single SPIKE motors for precise , speed, positioning, and torque control.

`Motor(port)`

`Motor(port, positive_direction=Direction.CLOCKWISE)`

Initializes a motor connected to a specific port. The expanded mode defines the direction rotation as Direction.CLOCKWISE or .COUNTERCLOCKWISE

`motor.speed()`

Returns the current speed of the motor in degrees per second.

`motor.angle()`

Returns the current absolute or relative angular position of the motor in degrees.

`motor.reset_angle(angle=0)`

Resets the motor's internal angle counter to a specified value.

`motor.run(speed)`

Runs the motor indefinitely at a constant speed (deg/s)

`motor.run_time(speed, time)`

`motor.run_time(speed, time, then=Stop.HOLD, wait=True)`

Runs the motor at a given speed for a specified duration in milliseconds.

`motor.run_angle(speed, rotation_angle)`

`motor.run_angle(speed, rotation_angle, then=Stop.HOLD, wait=True)`

Rotates the motor by a specified number of degrees in a specified speed.

`motor.run_target(speed, target_angle)`

`motor.run_target(speed, target_angle, then=Stop.HOLD, wait=True)`

Runs the motor to an absolute target angle in degrees.

`motor.stop()`

Stops the motor and lets it coast freely (Stop.COAST).

`motor.hold()`

Stops the motor and actively holds its current position (Stop.HOLD).

```
from pybricks.pupdevices import Motor
from pybricks.parameters import Port, Direction

# Initialize motor with COUNTERCLOCKWISE set as its (forward) direction.
# This permanently flips the motor's default rotation.
motor = Motor(Port.A, positive_direction=Direction.COUNTERCLOCKWISE)

# Method 1: Rotate backwards relative to motor setup using negative speed.
motor.run_time(-300, 2000)

# Method 2: Rotate backwards relative to motor setup using rotation angle.
motor.run_angle(speed=200, rotation_angle=-180)
```

```
from pybricks.pupdevices import Motor
from pybricks.parameters import Port

motor = Motor(Port.A)

#Rotate for 2 s (2000 ms) at speed of 300 deg/s
motor.run_time(300, 2000)
```

```
from pybricks.pupdevices import Motor
from pybricks.parameters import Port

motorA = Motor(Port.A)

# Rotate 360 degrees from the start position
motor.run_angle(speed=250, rotation_angle=360)
```

```
from pybricks.pupdevices import Motor
from pybricks.parameters import Port, Stop

motor = Motor(Port.A)

# Reset motor relative position to 0 degrees
motor.run_target(200, 0, then=Stop.HOLD)
```

```
from pybricks.pupdevices import Motor
from pybricks.parameters import Port
from pybricks.tools import wait

example_motor = Motor(Port.A)

# Rotate clockwise at 500 degrees per second.
example_motor.run(500)

wait(3000)    # Wait for 3 seconds.

# Rotate counterclockwise at 500 degrees/s
example_motor.run(-500)

wait(3000)    # Wait for 3 seconds.

# Stop the motor
example_motor.stop()
```

```
from pybricks.parameters import Port
from pybricks.pupdevices import Motor
from pybricks.tools import wait

# Initialize motors on port A and B.
left_arm = Motor(Port.A)
right_arm = Motor(Port.B)

# Both motors run at 500 degrees per second
left_arm.run(500)
right_arm.run(500)

# Both motors run together for 3 seconds.
wait(3000)

# Stop both motors at the end.
left_arm.stop()
right_arm.stop()
```

ROBOTS GOT TALENTS®