EdPy app documentation

A complete reference to the documentation available in the EdPy app.

Includes Edison V1, V2 and V3 compatibility tags.

3.2.2.18.5 rev 5

Contents

Compatibility tags show which Edison robot versions each entry is compatible with.

V1 Edison V1 V2 Edison V2 V3 Edison V3

EdPy functions

Ed.Exit()	V3	3
Ed.ReadUSB()	V3	4
Ed.WriteUSB()	V3	5
Ed.Random()	V3	6
Ed.List()	V1 V2 V3	7
Ed.LeftLed()	V1 V2 V3	9
Ed.RightLed()	V1 V2 V3	10
Ed.ObstacleDetectionBeam()	V1 V2 V3	11
Ed.LineTrackerLed()	V1 V2 V3	12
Ed.SendIRData()	V1 V2 V3	13
Ed.StartCountDown()	V1 V2 V3	14
Ed.TimeWait()	V1 V2 V3	15
Ed.RegisterEventHandler()	V1 V2 V3	16
Ed.PlayBeep()	V1 V2 V3	18
Ed.PlayMyBeep()	V1 V2 V3	19
Ed.PlayTone()	V1 V2 V3	20
Ed.PlayTune()	V1 V2 V3	22
Ed.TuneString()	V1 V2 V3	24
Ed.Drive()	V1 V2 V3	26
Ed.DriveLeftMotor()	V1 V2 V3	29
Ed.DriveRightMotor()	V1 V2 V3	32
Ed.ReadObstacleDetection()	V1 V2 V3	35
Ed.ReadKeypad()	V1 V2 V3	37
Ed.ReadClapSensor()	V1 V2 V3	38
Ed.ReadLineState()	V1 V2 V3	39
Ed.ReadRemote()	V1 V2 V3	40
Ed.ReadIRData()	V1 V2 V3	42
Ed.ReadLeftLightLevel()	V1 V2 V3	43
Ed.ReadRightLightLevel()	V1 V2 V3	44
Ed.ReadLineTracker()	V1 V2 V3	45
Ed.ReadCountDown()	V1 V2 V3	46
Ed.ReadMusicEnd()	V1 V2 V3	47
Ed.ReadDriveLoad()	V1 V2 V3	48
Ed.ReadDistance()	V2 V3	49
Ed.ReadRound()	V3	50
Ed.ReadTriangle()	V3	51
Ed.SetDistance()	V2 V3	52
Ed.ResetDistance()	V2 V3	53
Ed.AddLogEntry()	V3	54

Constants

Ed.ON	V1 V2 V3	55
Ed.OFF	V1 V2 V3	55
Ed.NOTE_#	V1 V2 V3	55
Ed.TEMPO_#	V1 V2 V3	56
Ed.FORWARD	V1 V2 V3	56

Ed.BACKWARD V1 V2 V3 57
Ed.FORWARD_RIGHT V1 V2 V3 57
Ed.BACKWARD_RIGHT V1 V2 V3 57
Ed.FORWARD_LEFT V1 V2 V3 57
Ed.BACKWARD_LEFT V1 V2 V3 57
Ed.SPIN_# V1 V2 V3 58
Ed.STOP V1 V2 V3 58
Ed.SPEED_# V1 V2 V3 59
Ed.DISTANCE_UNLIMITED V1 V2 V3 59
Ed.MOTOR_# V2 V3 59
Ed.TIME_# V1 V2 V3 60
Ed.OBSTACLE_# V1 V2 V3 60
Ed.LINE_ON_# V1 V2 V3 60
Ed.KEYPAD_# V1 V2 V3 60
Ed.CLAP_# V1 V2 V3 61
Ed.DRIVE_# V1 V2 V3 61
Ed.MUSIC_# V1 V2 V3 61
Ed.REMOTE_CODE_# V1 V2 V3 62
Ed.EVENT_# V1 V2 V3 62
Ed.CM V2 V3 63
Ed.INCH V2 V3 63
Ed.TIME V1 V2 V3 63
Ed.TICKS V3 64
Ed.V1 V1 64
Ed.V2 V2 64
Ed.V3 V3 64

EdPy variables

Ed.Tempo V1 V2 V3 65
Ed.DistanceUnits V1 V2 V3 65
Ed.EdisonVersion V1 V2 V3 67

Python language features

abs() V1 V2 V3 68
len() V1 V2 V3 68
range() V1 V2 V3 69
True V1 V2 V3 69
False V1 V2 V3 70
import V1 V2 V3 70
def V1 V2 V3 71
pass V1 V2 V3 71
for V1 V2 V3 72
while V1 V2 V3 72
if / elif / else V1 V2 V3 73
class V1 V2 V3 74
return V1 V2 V3 75

Ed.Exit()

EDPY FUNCTION

Compatible with V3

Parameters:

N/A

Returns:

N/A

Explanation:

Causes the program to end.

Examples:

Beep when the triangle button is pressed; exit when the round button is pressed.

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
while True:
    button = Ed.ReadKeypad()

    if button == Ed.KEYPAD_TRIANGLE:
        Ed.PlayBeep()
    elif button == Ed.KEYPAD_ROUND:
        Ed.Exit()
```

Watch out for:

This function causes Edison's program to end immediately, but it is not the only way a program can end. Programs without an infinite loop will end automatically once Edison has executed all of the program's instructions.

Ed.ReadUSB()

EDPY FUNCTION

Compatible with V3

Parameters:

N/A

Returns:

The last integer received from an Edison web app via USB, or `None` if no data has been received since the previous Ed.ReadUSB() call.

Explanation:

Reads the most recent integer received via USB and clears it.
Returns `None` if there is no USB data to receive.

Examples:

Use Edison as a musical keyboard by requesting notes with USB data.

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
while True:
    usbData = Ed.ReadUSB()
    while usbData == None:
        usbData = Ed.ReadUSB()
    if usbData == 2:
        Ed.PlayTone(Ed.NOTE_A_6, Ed.NOTE_WHOLE)
        while Ed.ReadMusicEnd() == Ed.MUSIC_NOT_FINISHED:
            pass
    elif usbData == 30:
        Ed.PlayTone(Ed.NOTE_B_6, Ed.NOTE_HALF)
        while Ed.ReadMusicEnd() == Ed.MUSIC_NOT_FINISHED:
            pass
    elif usbData == 18:
        Ed.PlayTone(Ed.NOTE_F_7, Ed.NOTE_QUARTER)
        while Ed.ReadMusicEnd() == Ed.MUSIC_NOT_FINISHED:
            pass
```

Watch out for:

Edison can only receive USB data while connected to a web app.

Ed.WriteUSB()

EDPY FUNCTION

Compatible with V3

Ed.WriteUSB(data)

Parameters:

Data:

An integer value to send to a web app via USB.

Returns:

N/A

Explanation:

Sends one integer value via USB for an Edison web app to receive.

Examples:

Send a simple value.

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
Ed.WriteUSB(28)
```

Send the current light level.

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM
#--------Your code below-----------
Ed.WriteUSB(Ed.ReadLeftLightLevel())
```

Watch out for:

Edison can only send USB data while connected to a web app.

Ed.Random()

EDPY FUNCTION

Compatible with V3

Ed.Random(start, stop)

Parameters:

Start:

An integer value that sets the lower bound of the random number range. Must be less than or equal to stop.

Stop:

An integer value that sets the upper bound of the random number range. Must be greater than or equal to start.

Returns:

A randomised integer between the start and stop values, inclusive. If start and stop are equal, that value is returned.

Explanation:

Generates a random value in the range specified by the start and stop values.

Examples:

Beep between 4 and 10 times.

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
for i in range(Ed.Random(4, 10)):
    Ed.PlayBeep()
    Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
```

Watch out for:

The stop value must be greater than or equal to the start value or Edison will be unable to generate a value in the required range.

Ed.List()

EDPY FUNCTION

Compatible with V1 V2 V3

`Ed.List(size)`

Parameters:

Size:

A positive integer that sets the number of integers in the new list.

The maximum size is 250 integers.

Returns:

An Edison list, which is a Python-style list.

`Ed.List(size, initialList)`

Parameters:

Size:

A positive integer that sets the number of integers in the new list.

The maximum size is 250 integers.

Initial List:

A Python-style list, for example [1, 2, 3], that sets the initial values in the new Ed.List.

On Edison V3, the initial list can contain a maximum of 120 values.

Returns:

An Edison list, which is a Python-style list.

Explanation:

Creates a list of Edison variables.

Examples:

Create an empty list and fill with zeros.

```
#--------Your code below-----------
zeros = Ed.List(5)
for x in range(5):
    zeros[x] = 0
```

Create a new list with pre-filled values.

```
#--------Your code below-----------
example = Ed.List(5, [1, 2, 3, 4, 5])
```

Watch out for:

The maximum list size is 250. On Edison V3, an initial list supplied when the list is created can contain a maximum of 120 values.

Additional new elements cannot be added to the end of the list. The list is a fixed size. Python lists are zero-indexed, meaning the first element in the list is at index 0. For example, using the pre-filled list in the example above, the following code would flash Edison's LED once.

```
while example[0] != 0:
    Ed.LeftLed(Ed.ON)
    Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
    Ed.LeftLed(Ed.OFF)
    Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
    example[0] = example[0] - 1
```

Ed.LeftLed()

EDPY FUNCTION

Compatible with V1 V2 V3

`Ed.LeftLed(state)`

Parameters:

State:

- Ed.ON - LED turns on
- Ed.OFF - LED turns off

Returns:

N/A

Explanation:

Turns Edison's left LED on or off.

Examples:

Quickly flash the left LED.

```
#--------Your code below-----------
Ed.LeftLed(Ed.ON)
Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
Ed.LeftLed(Ed.OFF)
Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
```

Left LED on while driving.

```
#--------Your code below-----------
Ed.LeftLed(Ed.ON)
Ed.Drive(Ed.FORWARD, Ed.SPEED_5, 10)
Ed.LeftLed(Ed.OFF)
```

Watch out for:

After turning Edison's LED on, call the function again with Ed.OFF to turn the LED off.

Ed.RightLed()

EDPY FUNCTION

Compatible with V1 V2 V3

`Ed.RightLed(state)`

Parameters:

State:

- Ed.ON - LED turns on
- Ed.OFF - LED turns off

Returns:

N/A

Explanation:

Turns Edison's right LED on or off.

Examples:

Quickly flash the right LED.

```
#--------Your code below-----------
Ed.RightLed(Ed.ON)
Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
Ed.RightLed(Ed.OFF)
Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
```

Right LED on while driving.

```
#--------Your code below-----------
Ed.RightLed(Ed.ON)
Ed.Drive(Ed.FORWARD, Ed.SPEED_5, 10)
Ed.RightLed(Ed.OFF)
```

Watch out for:

After turning Edison's LED on, call the function again with Ed.OFF to turn the LED off.

Ed.ObstacleDetectionBeam()

EDPY FUNCTION

Compatible with V1 V2 V3

Ed.ObstacleDetectionBeam(state)

Parameters:

State:

- Ed.ON - Obstacle detection functions are enabled.
- Ed.OFF - Obstacle detection functions are disabled.

Returns:

N/A

Explanation:

Turns Edison's infrared obstacle-detection system on or off. This is required to use other obstacle detection functions.

Examples:

Turn on the obstacle-detection beam and beep when an obstacle is detected.

```
#--------Your code below-----------
Ed.ObstacleDetectionBeam(Ed.ON)
while True:
    if Ed.ReadObstacleDetection() > Ed.OBSTACLE_NONE:
        Ed.PlayBeep()
```

Watch out for:

While the obstacle-detection beam needs to be turned on to enable Edison to detect an obstacle, this function is not used to detect obstacles. Use Ed.ReadObstacleDetection() to have Edison react to obstacles.

Edison's obstacle detection and IR messaging functions both use the same IR LEDs and IR receiver. Therefore, Edison cannot send or receive messages from other Edison robots if the obstacle-detection beam is turned on.

Ed.LineTrackerLed()

EDPY FUNCTION

Compatible with V1 V2 V3

`Ed.LineTrackerLed(state)`

Parameters:

State:

- Ed.ON - LED turns on
- Ed.OFF - LED turns off

Returns:

N/A

Explanation:

Turns Edison's line-tracking LED on or off. The line-tracking LED is required for Ed.ReadLineState() and recommended, but not required, for Ed.ReadLineTracker().

Examples:

Turn on the line-tracking LED and beep when a black surface is detected.

```
#--------Your code below-----------
Ed.LineTrackerLed(Ed.ON)
while True:
    if Ed.ReadLineState() == Ed.LINE_ON_BLACK:
        Ed.PlayBeep()
```

Watch out for:

Edison uses this first reading as the white-surface baseline for Ed.ReadLineState(). If the line-tracking LED is turned on while the robot is over a black line, Edison may be unable to detect a surface that reflects less light and therefore may never return Ed.LINE_ON_BLACK.

While the line-tracking LED needs to be turned on to enable Edison to detect a line, this function is not used to react to lines. Use Ed.ReadLineState() to have Edison react to lines.

Ed.SendIRData()

EDPY FUNCTION

Compatible with V1 V2 V3

`Ed.SendIRData(byte)`

Parameters:

Byte:

An integer from 0 to 255 to send to all nearby Edison robots.

Returns:

N/A

Explanation:

Sends one byte of data via infrared for nearby Edison robots to receive.

Examples:

Send a simple value.

```
#--------Your code below-----------
Ed.SendIRData(10)
```

Send line tracking data.

```
#--------Your code below-----------
Ed.LineTrackerLed(Ed.ON)
lineState = Ed.ReadLineState()
Ed.SendIRData(lineState)
```

Watch out for:

Only 8-bit values (0 to 255) can be sent. If a larger number is used, only its lowest 8 bits are sent.

Because Ed.ReadIRData() also returns 0 when no new message is available, use 1 to 255 when the receiving program must distinguish data from no data.

Ed.StartCountDown()

EDPY FUNCTION

Compatible with V1 V2 V3

`Ed.StartCountDown(time, units)`

Parameters:

Time:

The number of seconds or milliseconds to count down from.

The maximum value is 32,767.

Units:

- Ed.TIME_MILLISECONDS - Counts down in milliseconds.
- Ed.TIME_SECONDS - Counts down in seconds.

Returns:

N/A

Explanation:

Sets the countdown timer to a specified number of seconds or milliseconds. The counter then starts to count down in the background.

Examples:

Flash the left LED for 3,000 milliseconds.

```
#--------Your code below-----------
Ed.StartCountDown(3100, Ed.TIME_MILLISECONDS)
while Ed.ReadCountDown(Ed.TIME_MILLISECONDS) > 100:
    Ed.LeftLed(Ed.ON)
    Ed.TimeWait(50, Ed.TIME_MILLISECONDS)
    Ed.LeftLed(Ed.OFF)
    Ed.TimeWait(50, Ed.TIME_MILLISECONDS)
```

Watch out for:

The timer counts down only; it cannot be used as a count-up stopwatch.

The timer is not a clock and cannot tell time.

Ed.TimeWait()

EDPY FUNCTION

Compatible with V1 V2 V3

`Ed.TimeWait(time, units)`

Parameters:

Time:

A positive integer representing the number of seconds or milliseconds to wait.

The maximum value is 32,767.

Units:

- Ed.TIME_MILLISECONDS - Measures the wait time in milliseconds.
- Ed.TIME_SECONDS - Measures the wait time in seconds.

Returns:

N/A

Explanation:

Pauses program execution until the specified amount of time has elapsed.

Examples:

Turn on the left LED for three seconds, off for half a second and then back on.

```
#--------Your code below-----------
Ed.LeftLed(Ed.ON)
Ed.TimeWait(3, Ed.TIME_SECONDS)
Ed.LeftLed(Ed.OFF)
Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
Ed.LeftLed(Ed.ON)
```

Watch out for:

Time can only be set as an integer value. To wait for a period less than one second, use Ed.TIME_MILLISECONDS as the unit. For example, 3,500 milliseconds is equal to 3.5 seconds.

Ed.RegisterEventHandler()

EDPY FUNCTION

Compatible with V1 V2 V3

```
Ed.RegisterEventHandler(event, function)
```

Parameters:

Event:

- Ed.EVENT_TIMER_FINISHED - Calls the function when the countdown timer finishes.
- Ed.EVENT_REMOTE_CODE - Calls the function when Edison receives a remote code.
- Ed.EVENT_IR_DATA - Calls the function when Edison receives code from another Edison.
- Ed.EVENT_CLAP_DETECTED - Calls the function when Edison detects a clap.
- Ed.EVENT_OBSTACLE_ANY - Calls the function when Edison detects any obstacle.
- Ed.EVENT_OBSTACLE_LEFT - Calls the function when Edison detects an obstacle to the left.
- Ed.EVENT_OBSTACLE_RIGHT - Calls the function when Edison detects an obstacle to the right.
- Ed.EVENT_OBSTACLE_AHEAD - Calls the function when Edison detects an obstacle straight ahead.
- Ed.EVENT_DRIVE_STRAIN - Calls the function when Edison detects strain on the drive.
- Ed.EVENT_KEYPAD_TRIANGLE - Calls the function when Edison detects a triangle button press.
- Ed.EVENT_KEYPAD_ROUND - Calls the function when Edison detects a round button press.
- Ed.EVENT_LINE_TRACKER_ON_WHITE - Calls the function when Edison detects a white surface under the line tracker.
- Ed.EVENT_LINE_TRACKER_ON_BLACK - Calls the function when Edison detects a black surface under the line tracker.
- Ed.EVENT_LINE_TRACKER_SURFACE_CHANGE - Calls the function when Edison detects a surface change under the line tracker.
- Ed.EVENT_TUNE_FINISHED - Calls the function when Edison finishes playing a tune.

Function:

- The name of a user-created function, supplied as a string. This function is called when the event occurs.
- None - Disables the event, allowing event masking (Edison V3 only).

Returns:

N/A

Explanation:

Sets up or disables an event handler, causing Edison to call a function when a given event occurs.

Examples:

Flash the left LED forever and beep whenever an obstacle is detected without event masking.

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_VERY_SLOW

#--------Your code below-----------

def whenObsBeep():
    Ed.PlayBeep()
Ed.ObstacleDetectionBeam(Ed.ON)
Ed.RegisterEventHandler(Ed.EVENT_OBSTACLE_ANY, "whenObsBeep")
while True:
    Ed.LeftLed(Ed.ON)
    Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
    Ed.LeftLed(Ed.OFF)
    Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
```

Flash the left LED forever and beep whenever an obstacle is detected with event masking. (Edison V3 only)

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_VERY_SLOW

#--------Your code below-----------

def whenObsBeep():
    Ed.RegisterEventHandler(Ed.EVENT_OBSTACLE_ANY, None)
    Ed.PlayBeep()
    Ed.RegisterEventHandler(Ed.EVENT_OBSTACLE_ANY, "whenObsBeep")
Ed.ObstacleDetectionBeam(Ed.ON)
Ed.RegisterEventHandler(Ed.EVENT_OBSTACLE_ANY, "whenObsBeep")
while True:
    Ed.LeftLed(Ed.ON)
    Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
    Ed.LeftLed(Ed.OFF)
    Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
```

Watch out for:

Event handlers act as an interrupt, which means that when the event occurs, the main program is paused while the given function is run. When the function completes, the main program continues where it left off.

Edison V3 robots detect events even while an event-handler function is running. This can cause a second event to trigger the handler again immediately after it finishes. Repeated triggers can be avoided by disabling the event handler while the function is running. Disable the handler by calling Ed.RegisterEventHandler() at the start of the function with None as the function parameter. Re-enable the handler at the end of the function by calling Ed.RegisterEventHandler() with the function name.

Ed.PlayBeep()

EDPY FUNCTION

Compatible with V1 V2 V3

Parameters:

N/A

Returns:

N/A

Explanation:

Sounds a single beep at 3.5 kHz for 125 ms (0.125 seconds).

Examples:

Play a beep.

```
#--------Your code below-----------
Ed.PlayBeep()
```

Watch out for:

All of Edison's sounds occur in the background. As such, Edison moves on to the next line of code as soon as the sound starts without waiting for the sound to finish. To make Edison wait for the sound to finish, use the Ed.ReadMusicEnd() function in a loop.

Ed.PlayMyBeep()

EDPY FUNCTION

Compatible with V1 V2 V3

```
Ed.PlayMyBeep(period)
```

Parameters:

Period:

The period is how long it takes an acoustic wave to complete a full cycle. Changing this number causes a change in the frequency of the sound played because when the period increases, the frequency decreases. To convert a frequency into a period, divide the number 8,000,000 by the desired frequency.

For example, to play a 1 kHz (1,000 cycles per second) sound: 8,000,000/1,000 = 8,000

Returns:

N/A

Explanation:

Sounds a single beep with a given period for a duration of 50 ms (0.05 seconds).

Examples:

Play a beep at 1 kHz using a period value of 8,000.

```
#--------Your code below-----------
Ed.PlayMyBeep(8000)
```

Watch out for:

All of Edison's sounds occur in the background. As such, Edison moves on to the next line of code as soon as the sound starts without waiting for the sound to finish. To make Edison wait for the sound to finish, use the Ed.ReadMusicEnd() function in a loop.

The largest number Edison can use is 32,767, which means Edison cannot handle the number 8,000,000, which is required to convert frequency to period. Therefore, you will need to calculate the period you want before programming Edison and use this as the number in your program.

Ed.PlayTone()

EDPY FUNCTION

Compatible with V1 V2 V3

```
Ed.PlayTone(note, duration)
```

Parameters:

Note:

- Ed.NOTE_A_6 - Play a low A.
- Ed.NOTE_A_SHARP_6 - Play a low A sharp.
- Ed.NOTE_B_6 - Play a low B.
- Ed.NOTE_C_7 - Play a C.
- Ed.NOTE_C_SHARP_7 - Play a C sharp.
- Ed.NOTE_D_7 - Play a D.
- Ed.NOTE_D_SHARP_7 - Play a D sharp.
- Ed.NOTE_E_7 - Play an E.
- Ed.NOTE_F_7 - Play an F.
- Ed.NOTE_F_SHARP_7 - Play an F sharp.
- Ed.NOTE_G_7 - Play a G.
- Ed.NOTE_G_SHARP_7 - Play a G sharp.
- Ed.NOTE_A_7 - Play an A.
- Ed.NOTE_A_SHARP_7 - Play an A sharp.
- Ed.NOTE_B_7 - Play a B.
- Ed.NOTE_C_8 - Play a high C.
- Ed.NOTE_REST - Play a rest.
- A positive integer representing the period. The period is how long it takes an acoustic wave to complete a full cycle. Changing this number causes a change in the frequency of the sound played because when the period increases, the frequency decreases. To convert a frequency into a period, divide the number 8,000,000 by the desired frequency. For example, to play a 1 kHz (1,000 cycles per second) sound: 8,000,000/1,000 = 8,000

Duration:

- Ed.NOTE_SIXTEENTH - Play a note for 125 milliseconds.
- Ed.NOTE_EIGHTH - Play a note for 250 milliseconds.
- Ed.NOTE_QUARTER - Play a note for 500 milliseconds.
- Ed.NOTE_HALF - Play a note for 1,000 milliseconds.
- Ed.NOTE_WHOLE - Play a note for 2,000 milliseconds.
- An integer from 0 to 32,767 representing the duration in milliseconds.

Returns:

N/A

Explanation:

Sounds a single note with a given period for a given length of time.

Examples:

Play a 1 kHz note using a period value of 8,000 for two seconds (2,000 milliseconds).

```
#--------Your code below-----------
Ed.PlayTone(8000, 2000)
while Ed.ReadMusicEnd() == Ed.MUSIC_NOT_FINISHED:
    pass
```

Play an A sharp for half a second.

```
#--------Your code below-----------
Ed.PlayTone(Ed.NOTE_A_SHARP_6, Ed.NOTE_QUARTER)
while Ed.ReadMusicEnd() == Ed.MUSIC_NOT_FINISHED:
    pass
```

Watch out for:

All of Edison's sounds occur in the background. As such, Edison moves on to the next line of code as soon as the sound starts without waiting for the sound to finish. To make Edison wait for the sound to finish, use the Ed.ReadMusicEnd() function in a loop.

The largest number Edison can use is 32,767, which means Edison cannot handle the number 8,000,000, which is required to convert frequency to period. Therefore, you will need to calculate the period you want before programming Edison and use this as the number in your program.

To find the period for a musical note, divide 8,000,000 by the note's frequency. For example, the note A at 440 Hz gives 8,000,000 ÷ 440 = 18,181 (rounded down), which is the value of Ed.NOTE_A_6.

Ed.PlayTune()

EDPY FUNCTION

Compatible with V1 V2 V3

```
Ed.PlayTune(tune)
```

Parameters:

Tune:

An Edison tune represented by a Python-style string created with Ed.TuneString().

A tune string looks like this: "ndndndndndnd...ndz" where n is a note from the notes table and d is a duration from the duration table; see below.

Notes:

m - low A

M - low A sharp

n - low B

c - C

C - C sharp

d - D

D - D sharp

e - E

f - F

F - F sharp

g - G

G - G sharp

a - A

A - A sharp

b - B

o - high C

Duration:

1 - whole note

2 - half note

4 - quarter note

8 - eighth note

6 - sixteenth note

Other:

r - rest

z - end of tune

Returns:

N/A

Explanation:

Plays a tune through Edison's speaker. The function accepts a tune string created with Ed.TuneString(). Change Ed.Tempo in the setup code to adjust the speed at which the tune plays.

Examples:

Play a simple tune.

```
#--------Your code below-----------
simple = Ed.TuneString(25, "d4e4f4e4d4c4n2d4e4f4e4d1z")
Ed.PlayTune(simple)
while Ed.ReadMusicEnd() == Ed.MUSIC_NOT_FINISHED:
    pass
```

Watch out for:

Ed.PlayTune() requires a Python-style tune string created with Ed.TuneString().

All tune strings must end with the character "z".

All of Edison's sounds occur in the background. As such, Edison moves on to the next line of code as soon as the sound starts without waiting for the sound to finish. To make Edison wait for the sound to finish, use the Ed.ReadMusicEnd() function in a loop.

You can change the speed at which your tune plays by changing Ed.Tempo in the setup code.

Ed.TuneString()

EDPY FUNCTION

Compatible with V1 V2 V3

```
Ed.TuneString(size)
```

Parameters:

Size:

A positive integer that sets the number of characters in the new string.

The maximum size is 250 characters.

Returns:

An Edison tune, which is a Python-style string.

```
Ed.TuneString(size, initialTune)
```

Parameters:

Size:

A positive integer that sets the number of characters in the new string.

The maximum size is 250 characters.

initialTune:

A Python-style string, for example "a4g2z", that sets the notes to be played in the tune.

A tune string looks like this: "ndndndndndnd...ndz" where n is a note from the notes table and d is a duration from the duration table; see below.

Notes:

m - low A

M - low A sharp

n - low B

c - C

C - C sharp

d - D

D - D sharp

e - E

f - F

F - F sharp

g - G

G - G sharp

a - A

A - A sharp

b - B

o - high C

Duration:

1 - whole note

2 - half note

4 - quarter note

8 - eighth note

6 - sixteenth note

Other:

r - rest

z - end of the tune

Returns:

An Edison tune, which is a Python-style string.

Explanation:

Creates a new tune string that can be used with the Ed.PlayTune() function.

Examples:

Play a simple tune.

```
#--------Your code below-----------
simple = Ed.TuneString(25, "d4e4f4e4d4c4n2d4e4f4e4d1z")
Ed.PlayTune(simple)
while Ed.ReadMusicEnd() == Ed.MUSIC_NOT_FINISHED:
    pass
```

Watch out for:

This function creates a tune string; it does not play the tune. Use Ed.PlayTune() to play it.

The maximum tune size is 250 characters.

All tune strings must end with the character "z".

All of Edison's sounds occur in the background. As such, Edison moves on to the next line of code as soon as the sound starts without waiting for the sound to finish. To make Edison wait for the sound to finish, use the Ed.ReadMusicEnd() function in a loop.

You can change the speed at which your tune plays by changing Ed.Tempo in the setup code.

Ed.Drive()

EDPY FUNCTION

Compatible with V1 V2 V3

```
Ed.Drive(direction, speed, distance)
```

Parameters:

Direction:

- Ed.FORWARD - Edison drives forwards.
- Ed.BACKWARD - Edison drives backwards.
- Ed.FORWARD_RIGHT - Edison uses one wheel to turn forwards right (clockwise).
- Ed.BACKWARD_RIGHT - Edison uses one wheel to turn backwards right (counter-clockwise).
- Ed.FORWARD_LEFT - Edison uses one wheel to turn forwards left (counter-clockwise).
- Ed.BACKWARD_LEFT - Edison uses one wheel to turn backwards left (clockwise).
- Ed.SPIN_RIGHT - Edison spins on the spot to the right (clockwise).
- Ed.SPIN_LEFT - Edison spins on the spot to the left (counter-clockwise).
- Ed.STOP - Stops Edison immediately.

Speed:

- A positive integer from 1 to 10.
- Ed.SPEED_1 - PWM-controlled speed 1.
- Ed.SPEED_2 - PWM-controlled speed 2.
- Ed.SPEED_3 - PWM-controlled speed 3.
- Ed.SPEED_4 - PWM-controlled speed 4.
- Ed.SPEED_5 - PWM-controlled speed 5.
- Ed.SPEED_6 - PWM-controlled speed 6.
- Ed.SPEED_7 - PWM-controlled speed 7.
- Ed.SPEED_8 - PWM-controlled speed 8.
- Ed.SPEED_9 - PWM-controlled speed 9.
- Ed.SPEED_10 - PWM-controlled speed 10.
- Ed.SPEED_FULL - Full power to the motors. (Note: Edison may not drive perfectly straight with this setting.)

Distance:

- An integer representing the distance. The maximum value is 32,767.

OR

- Ed.DISTANCE_UNLIMITED - Turns on Edison's motors and moves on with the code. (Note: a stop will be needed later in the code.)

For an integer distance, the unit is set by Ed.DistanceUnits in the setup code. Change Ed.DistanceUnits in the setup code to select the unit.

- Ed.DistanceUnits = Ed.CM - distance in centimetres (default for V2 and V3).
- Ed.DistanceUnits = Ed.INCH - distance in inches.
- Ed.DistanceUnits = Ed.TIME - distance in milliseconds (default for V1).
- Ed.DistanceUnits = Ed.TICKS - distance in encoder ticks (V3 only).

Note: When Edison is turning and Ed.DistanceUnits is set to CM or INCH, distance becomes the number of degrees to turn with a maximum value of 360. When it is set to Ed.TICKS, distance stays in ticks.

Returns:

N/A

Explanation:

Controls both of Edison's motors to create movement. The function can move for a set distance (CM, INCH or TICKS) or a period of time (TIME). It completes the specified movement before continuing to the next line of code.

Examples:

Drive Edison forward for 3 cm at speed 5. (Edison V2 and V3 only)

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
Ed.Drive(Ed.FORWARD, Ed.SPEED_5, 3)
```

Drive Edison forward for 5 inches at speed 5. (Edison V2 and V3 only)

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.INCH
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
Ed.Drive(Ed.FORWARD, Ed.SPEED_5, 5)
```

Drive Edison forward for 2,000 milliseconds at speed 7.

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.TIME
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
Ed.Drive(Ed.FORWARD, Ed.SPEED_7, 2000)
```

Spin Edison left 90 degrees at speed 10.

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
Ed.Drive(Ed.SPIN_LEFT, 10, 90)
```

Set speed and distance with variables.

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
driveSpeed = 5
driveDistance = 10
Ed.Drive(Ed.FORWARD, driveSpeed, driveDistance)
```

Watch out for:

Distance units of CM and INCH are available only for Edison V2 and V3. Distance units of TICKS are available only for Edison V3. If you are using a V1 Edison, please make sure that Ed.DistanceUnits = Ed.TIME.

The Ed.TIME constant is in milliseconds. When Ed.DistanceUnits = Ed.TIME, remember that the distance input integer is milliseconds. For example, to drive for 2 seconds, enter a distance of 2,000.

When Edison is turning and Ed.DistanceUnits is set to CM or INCH, distance becomes the number of degrees to turn with a maximum value of 360. If you enter a value above 360, the code will wrap the value around and Edison will perform a shorter turn. For example, an input of 380 will cause Edison to turn 20 degrees.

When Ed.DistanceUnits is set to Ed.TICKS, distance is always measured in ticks, even when Edison is turning or spinning. For example, Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_5, 90) makes Edison spin about 180 degrees.

Ed.SPEED_FULL turns Edison's motors up to the maximum value. As such, Edison has no control over the speed and Edison V2 and V3 robots cannot use their encoders to correct for driving drift.

Setting distance to 0 or Ed.DISTANCE_UNLIMITED makes Edison turn on the motors and move on with the code. A later drive-function call is required to stop or change the movement.

When the distance input is set to anything other than 0 or Ed.DISTANCE_UNLIMITED, Edison will drive for the full distance supplied before moving on to the next line of code.

Ed.DriveLeftMotor()

EDPY FUNCTION

Compatible with V1 V2 V3

```
Ed.DriveLeftMotor(direction, speed, distance)
```

Parameters:

Direction:

- Ed.FORWARD - Edison's left motor drives forwards.
- Ed.BACKWARD - Edison's left motor drives backwards.
- Ed.STOP - Stops Edison's left motor immediately.

Speed:

- A positive integer from 1 to 10.
- Ed.SPEED_1 - PWM-controlled speed 1.
- Ed.SPEED_2 - PWM-controlled speed 2.
- Ed.SPEED_3 - PWM-controlled speed 3.
- Ed.SPEED_4 - PWM-controlled speed 4.
- Ed.SPEED_5 - PWM-controlled speed 5.
- Ed.SPEED_6 - PWM-controlled speed 6.
- Ed.SPEED_7 - PWM-controlled speed 7.
- Ed.SPEED_8 - PWM-controlled speed 8.
- Ed.SPEED_9 - PWM-controlled speed 9.
- Ed.SPEED_10 - PWM-controlled speed 10.
- Ed.SPEED_FULL - Full power to the left motor.

Distance:

- An integer representing the distance. The maximum value is 32,767.

OR

- Ed.DISTANCE_UNLIMITED - Turns on Edison's left motor and moves on with the code. (Note: a stop will be needed later in the code.)

For an integer distance, the unit is set by Ed.DistanceUnits in the setup code. Change Ed.DistanceUnits in the setup code to select the unit.

- Ed.DistanceUnits = Ed.CM - distance in centimetres (default for V2 and V3).
- Ed.DistanceUnits = Ed.INCH - distance in inches.
- Ed.DistanceUnits = Ed.TIME - distance in milliseconds (default for V1).
- Ed.DistanceUnits = Ed.TICKS - distance in encoder ticks (V3 only).

Returns:

N/A

Explanation:

Controls Edison's left motor to create movement. The function can move for a set distance (CM, INCH or TICKS) or a period of time (TIME). It completes the specified movement before continuing to the next line of code.

Examples:

Drive Edison's left motor forward for 3 cm at speed 5. (Edison V2 and V3 only)

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
Ed.DriveLeftMotor(Ed.FORWARD, Ed.SPEED_5, 3)
```

Drive Edison's left motor forward for 5 inches at speed 5. (Edison V2 and V3 only)

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.INCH
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
Ed.DriveLeftMotor(Ed.FORWARD, Ed.SPEED_5, 5)
```

Drive Edison's left motor forward for 2,000 milliseconds at speed 7.

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.TIME
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
Ed.DriveLeftMotor(Ed.FORWARD, Ed.SPEED_7, 2000)
```

Set speed and distance with variables.

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
driveSpeed = 5
driveDistance = 10
Ed.DriveLeftMotor(Ed.FORWARD, driveSpeed, driveDistance)
```

Watch out for:

Distance units of CM and INCH are available only for Edison V2 and V3. Distance units of TICKS are available only for Edison V3. If you are using a V1 Edison, please make sure that Ed.DistanceUnits = Ed.TIME.

The Ed.TIME constant is in milliseconds. When Ed.DistanceUnits = Ed.TIME, remember that the distance input integer is milliseconds. For example, to drive for 2 seconds, enter a distance of 2,000.

Ed.SPEED_FULL runs Edison's left motor at full, unregulated power. Edison has no control over the motor's speed at this setting, and Edison V2 and V3 robots cannot use the encoder to correct it.

Setting distance to 0 or Ed.DISTANCE_UNLIMITED makes Edison turn on the left motor and move on with the code. A later drive-function call is required to stop or change the movement.

When the distance input is set to anything other than 0 or Ed.DISTANCE_UNLIMITED, Edison will drive for the full distance supplied before moving on to the next line of code.

Ed.DriveRightMotor()

EDPY FUNCTION

Compatible with V1 V2 V3

```
Ed.DriveRightMotor(direction, speed, distance)
```

Parameters:

Direction:

- Ed.FORWARD - Edison's right motor drives forwards.
- Ed.BACKWARD - Edison's right motor drives backwards.
- Ed.STOP - Stops Edison's right motor immediately.

Speed:

- A positive integer from 1 to 10.
- Ed.SPEED_1 - PWM-controlled speed 1.
- Ed.SPEED_2 - PWM-controlled speed 2.
- Ed.SPEED_3 - PWM-controlled speed 3.
- Ed.SPEED_4 - PWM-controlled speed 4.
- Ed.SPEED_5 - PWM-controlled speed 5.
- Ed.SPEED_6 - PWM-controlled speed 6.
- Ed.SPEED_7 - PWM-controlled speed 7.
- Ed.SPEED_8 - PWM-controlled speed 8.
- Ed.SPEED_9 - PWM-controlled speed 9.
- Ed.SPEED_10 - PWM-controlled speed 10.
- Ed.SPEED_FULL - Full power to the right motor.

Distance:

- An integer representing the distance. The maximum value is 32,767.

OR

- Ed.DISTANCE_UNLIMITED - Turns on Edison's right motor and moves on with the code. (Note: a stop will be needed later in the code.)

For an integer distance, the unit is set by Ed.DistanceUnits in the setup code. Change Ed.DistanceUnits in the setup code to select the unit.

- Ed.DistanceUnits = Ed.CM - distance in centimetres (default for V2 and V3).
- Ed.DistanceUnits = Ed.INCH - distance in inches.
- Ed.DistanceUnits = Ed.TIME - distance in milliseconds (default for V1).
- Ed.DistanceUnits = Ed.TICKS - distance in encoder ticks (V3 only).

Returns:

N/A

Explanation:

Controls Edison's right motor to create movement. The function can move for a set distance (CM, INCH or TICKS) or a period of time (TIME). It completes the specified movement before continuing to the next line of code.

Examples:

Drive Edison's right motor forward for 3 cm at speed 5. (Edison V2 and V3 only)

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
Ed.DriveRightMotor(Ed.FORWARD, Ed.SPEED_5, 3)
```

Drive Edison's right motor forward for 5 inches at speed 5. (Edison V2 and V3 only)

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.INCH
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
Ed.DriveRightMotor(Ed.FORWARD, Ed.SPEED_5, 5)
```

Drive Edison's right motor forward for 2,000 milliseconds at speed 7.

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.TIME
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
Ed.DriveRightMotor(Ed.FORWARD, Ed.SPEED_7, 2000)
```

Set speed and distance with variables.

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
driveSpeed = 5
driveDistance = 10
Ed.DriveRightMotor(Ed.FORWARD, driveSpeed, driveDistance)
```

Watch out for:

Distance units of CM and INCH are available only for Edison V2 and V3. Distance units of TICKS are available only for Edison V3. If you are using a V1 Edison, please make sure that Ed.DistanceUnits = Ed.TIME.

The Ed.TIME constant is in milliseconds. When Ed.DistanceUnits = Ed.TIME, remember that the distance input integer is milliseconds. For example, to drive for 2 seconds, enter a distance of 2,000.

Ed.SPEED_FULL runs Edison's right motor at full, unregulated power. Edison has no control over the motor's speed at this setting, and Edison V2 and V3 robots cannot use the encoder to correct it.

Setting distance to 0 or Ed.DISTANCE_UNLIMITED makes Edison turn on the right motor and move on with the code. A later drive-function call is required to stop or change the movement.

When the distance input is set to anything other than 0 or Ed.DISTANCE_UNLIMITED, Edison will drive for the full distance supplied before moving on to the next line of code.

Ed.ReadObstacleDetection()

EDPY FUNCTION

Compatible with V1 V2 V3

Parameters:

N/A

Returns:

- Ed.OBSTACLE_NONE - Edison has not detected an obstacle.
- Ed.OBSTACLE_RIGHT - Edison has detected an obstacle on the right.
- Ed.OBSTACLE_LEFT - Edison has detected an obstacle on the left.
- Ed.OBSTACLE_AHEAD - Edison has detected an obstacle straight ahead.

Explanation:

Reads Edison's obstacle-detection state, returns its value, and then clears the obstacle-detection register. Ed.ObstacleDetectionBeam(Ed.ON) must be called before this function can return a value other than Ed.OBSTACLE_NONE.

Examples:

Play a beep when any obstacle is detected.

```
#--------Your code below-----------
Ed.ObstacleDetectionBeam(Ed.ON)
while True:
    if Ed.ReadObstacleDetection() > Ed.OBSTACLE_NONE:
        Ed.PlayBeep()
```

Drive until an obstacle is detected ahead.

```
#--------Your code below-----------
Ed.ObstacleDetectionBeam(Ed.ON)
#clear obstacle-detection register
Ed.ReadObstacleDetection()
Ed.Drive(Ed.FORWARD, 5, Ed.DISTANCE_UNLIMITED)
while Ed.ReadObstacleDetection() != Ed.OBSTACLE_AHEAD:
    pass
Ed.Drive(Ed.STOP, 0, 0)
```

Wait 3 seconds, then clear the obstacle-detection register before looking for new obstacles to be detected and signalled with a beep.

```
#--------Your code below-----------
Ed.ObstacleDetectionBeam(Ed.ON)
Ed.TimeWait(3, Ed.TIME_SECONDS)
Ed.ReadObstacleDetection()
while Ed.ReadObstacleDetection() != Ed.OBSTACLE_AHEAD:
    pass
Ed.PlayBeep()
```

Watch out for:

Ed.ObstacleDetectionBeam(Ed.ON) must be called for this function to return any value other than Ed.OBSTACLE_NONE.

When the obstacle-detection beam is set to ON, Edison is constantly updating the obstacle detection state. This function will read the state. As such, the function may return a detection recorded before the read function is called in your code.

The read function clears the obstacle detection state.

When using a read function inside a loop, include a read function outside of the loop before the loop to clear any previous data.

Ed.ReadKeypad()

EDPY FUNCTION

Compatible with V1 V2 V3

Parameters:

N/A

Returns:

- Ed.KEYPAD_NONE - None of Edison's buttons have been pressed.
- Ed.KEYPAD_TRIANGLE - Edison's triangle button has been pressed.
- Ed.KEYPAD_ROUND - Edison's round button has been pressed.

Explanation:

Reads Edison's keypad state, returns its value, and then clears Edison's keypad register. Edison's keypad state is set whenever a button is pressed, regardless of what the code is doing at the time.

Examples:

Play a beep when any button is pressed.

```
#--------Your code below-----------
while True:
    if Ed.ReadKeypad() != Ed.KEYPAD_NONE:
        Ed.PlayBeep()
```

Wait for the triangle button to be pressed, then beep.

```
#--------Your code below-----------
#clear previous presses
Ed.ReadKeypad()
while Ed.ReadKeypad() != Ed.KEYPAD_TRIANGLE:
    pass
Ed.PlayBeep()
```

Wait 3 seconds, clear the keypad state, then wait for a new button press and signal it with a beep.

```
#--------Your code below-----------
Ed.TimeWait(3, Ed.TIME_SECONDS)
Ed.ReadKeypad()
while Ed.ReadKeypad() == Ed.KEYPAD_NONE:
    pass
Ed.PlayBeep()
```

Watch out for:

Edison is constantly updating the keypad state. This function will read the state. As such, the function may read a keypad press from before the read function is called in your code.

The read function clears the keypad state.

When using a read function inside a loop, include a read function outside of the loop before the loop to clear any previous data.

Ed.ReadClapSensor()

EDPY FUNCTION

Compatible with V1 V2 V3

Parameters:

N/A

Returns:

- Ed.CLAP_NOT_DETECTED - Edison has not detected a clap.
- Ed.CLAP_DETECTED - Edison has detected a clap.

Explanation:

Reads Edison's clap detection state, returns its value, and then clears Edison's clap detection register. Edison's clap detection state is set whenever a clap is detected, regardless of what the code is doing at the time.

Examples:

Flash an LED when a clap is detected.

```
#--------Your code below-----------
while True:
    if Ed.ReadClapSensor() == Ed.CLAP_DETECTED:
        Ed.LeftLed(Ed.ON)
        Ed.TimeWait(50, Ed.TIME_MILLISECONDS)
        Ed.LeftLed(Ed.OFF)
        Ed.TimeWait(50, Ed.TIME_MILLISECONDS)
```

Beep when a clap is detected after a drive.

```
#--------Your code below-----------
Ed.Drive(Ed.FORWARD, 5, 10)
# wait a short time and clear the clap that was detected during the drive
Ed.TimeWait(350, Ed.TIME_MILLISECONDS)
Ed.ReadClapSensor()
# wait for a new clap
while Ed.ReadClapSensor() == Ed.CLAP_NOT_DETECTED:
    pass
Ed.PlayBeep()
```

Watch out for:

Edison is constantly updating the clap detection state. This function will read the state. As such, the function may read a clap from before the read function is called in your code.

The read function clears the clap detection state.

When using a read function inside a loop, include a read function outside of the loop before the loop to clear any previous data.

Edison's motors cause noise that will be detected and registered as "claps" while Edison is in motion. Therefore, driving Edison will cause claps to be detected. Make sure that you clear the clap detection state a few milliseconds after driving has finished before calling the function to detect a new clap event.

Ed.ReadLineState()

EDPY FUNCTION

Compatible with V1 V2 V3

Parameters:

N/A

Returns:

- Ed.LINE_ON_BLACK - Edison's line tracker is over a non-reflective surface.
- Ed.LINE_ON_WHITE - Edison's line tracker is over a reflective surface.

Explanation:

Reads the current line-tracker status based on the reflected light from the line-tracking sensor.

Examples:

Play a beep when a black surface is detected.

```
#--------Your code below-----------
Ed.LineTrackerLed(Ed.ON)
while True:
    if Ed.ReadLineState() == Ed.LINE_ON_BLACK:
        Ed.PlayBeep()
```

Watch out for:

Ed.LineTrackerLed(Ed.ON) must be called for this function to return any value.

Edison sets the Ed.LINE_ON_WHITE status when the line-tracking LED is turned on and determines the Ed.LINE_ON_BLACK status by looking for a sharp drop-off in reflected light. If Edison is on a black line when the line-tracking LED is turned on, Edison will mistakenly use that value as Ed.LINE_ON_WHITE and will be unable to detect a sharp drop-off to assign Ed.LINE_ON_BLACK. This will cause an error where Ed.LINE_ON_BLACK remains unset.

When the line tracker LED is ON, Edison is constantly updating the line tracker state, so this function continuously reads the current state of the line tracker.

Ed.ReadRemote()

EDPY FUNCTION

Compatible with V1 V2 V3

Ed.ReadRemote(clearRegister)

Parameters:

Edison V3 only: clearRegister (optional):

- True (default) - Clear the data in the remote-control register after reading.
- False - Leave the data in the remote-control register after reading.

Returns:

- Ed.REMOTE_CODE_NONE - Edison has not received a remote code.
- Ed.REMOTE_CODE_RELEASE - Edison has received the release button code.
- Ed.REMOTE_CODE_0 - Edison has received remote code 0.
- Ed.REMOTE_CODE_1 - Edison has received remote code 1.
- Ed.REMOTE_CODE_2 - Edison has received remote code 2.
- Ed.REMOTE_CODE_3 - Edison has received remote code 3.
- Ed.REMOTE_CODE_4 - Edison has received remote code 4.
- Ed.REMOTE_CODE_5 - Edison has received remote code 5.
- Ed.REMOTE_CODE_6 - Edison has received remote code 6.
- Ed.REMOTE_CODE_7 - Edison has received remote code 7.

Explanation:

Reads the last received remote-control code and, by default, clears the remote-control register.

Examples:

Play a beep when each remote code is received in sequence.

```
#--------Your code below-----------
codes = Ed.List(8)
codes[0] = Ed.REMOTE_CODE_0
codes[1] = Ed.REMOTE_CODE_1
codes[2] = Ed.REMOTE_CODE_2
codes[3] = Ed.REMOTE_CODE_3
codes[4] = Ed.REMOTE_CODE_4
codes[5] = Ed.REMOTE_CODE_5
codes[6] = Ed.REMOTE_CODE_6
codes[7] = Ed.REMOTE_CODE_7
#clear remote-control register
Ed.ReadRemote()
for x in range(8):
    while Ed.ReadRemote() != codes[x]:
        pass
    Ed.PlayBeep()
    Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
```

Drive forwards while remote code 1 is being received.

```
#--------Your code below-----------
while True:
    if Ed.ReadRemote() == Ed.REMOTE_CODE_1:
        Ed.Drive(Ed.FORWARD, 5, Ed.DISTANCE_UNLIMITED)
        Ed.TimeWait(300, Ed.TIME_MILLISECONDS)
    else:
        Ed.Drive(Ed.STOP, 1, 1)
```

Check for multiple remote codes without using a variable.

```
#--------Your code below-----------
while True:
    if Ed.ReadRemote(False) == Ed.REMOTE_CODE_1:
        Ed.LeftLed(Ed.ON)
    elif Ed.ReadRemote(False) == Ed.REMOTE_CODE_2:
        Ed.LeftLed(Ed.OFF)
```

Watch out for:

Edison can only react to remote-control codes that have been saved into the robot's memory using the barcodes provided. See https://meetedison.com/robot-activities/youre-a-robot-programmer/remote-control-barcodes/ for the full list.

Edison robots use the same IR receiver to receive remote-control codes and IR data from other Edison robots, and to perform obstacle detection. Therefore, Ed.ObstacleDetectionBeam must be OFF for Edison to receive a remote-control code.

Edison is constantly updating the remote-control register. The function may therefore return a remote-control code from before the read function is called in your code.

The read function clears the remote-control register by default. To read the remote-control register multiple times in quick succession, use the optional clearRegister parameter.

When using a read function inside a loop, include a read function outside of the loop before the loop to clear any previous data.

Ed.ReadIRData()

EDPY FUNCTION

Compatible with V1 V2 V3

Ed.ReadIRData(clearRegister)

Parameters:

Edison V3 only: clearRegister (optional):

- True (default) - Clear data in IR data register after reading.
- False - Leave data in IR data register after reading.

Returns:

- 1-255 - The last infrared data value received from another Edison robot.
- 0 - No new data received (or a received value of 0).

Explanation:

Reads the last received infrared data sent from another Edison robot and clears the IR data register by default. Data can be left in the IR data register by setting the optional clearRegister parameter to False.

Examples:

Play a beep when the value 10 is received from another Edison robot.

```
#--------Your code below-----------
Ed.ReadIRData()
while True:
    if Ed.ReadIRData() == 10:
        Ed.PlayBeep()
        Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
```

Check for multiple IR messages without using a variable.

```
#--------Your code below-----------
while True:
    if Ed.ReadIRData(False) == 10:
        Ed.PlayMyBeep(1000)
        Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
    elif Ed.ReadIRData(False) == 20:
        Ed.PlayMyBeep(2000)
        Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
```

Watch out for:

Edison robots use the same IR receiver to receive remote-control codes and IR data from other Edison robots, and to perform obstacle detection. Therefore, Ed.ObstacleDetectionBeam must be OFF for Edison to receive IR data from another robot.

Edison is constantly updating the data-received register. This function will read the register. As such, the function may read data from before the read function is called in your code.

The read function clears the IR data register by default. To read the IR data register multiple times in quick succession, use the optional clearRegister parameter.

When using a read function inside a loop, include a read function outside of the loop before the loop to clear any previous data.

Ed.ReadLeftLightLevel()

EDPY FUNCTION

Compatible with V1 V2 V3

Parameters:

N/A

Returns:

The current light level of the left light sensor.

Explanation:

Reads the current light level of the left light sensor.

Examples:

Play a beep whenever the left light level reading is higher than the right light level reading.

```
#--------Your code below-----------
while True:
    if Ed.ReadLeftLightLevel() > Ed.ReadRightLightLevel():
        Ed.PlayBeep()
        Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
```

Watch out for:

Edison reads the light level using an analogue-to-digital converter, so the returned value can range from 0 to 1,023.

Ed.ReadRightLightLevel()

EDPY FUNCTION

Compatible with V1 V2 V3

Parameters:

N/A

Returns:

The current light level of the right light sensor.

Explanation:

Reads the current light level of the right light sensor.

Examples:

Play a beep whenever the right light level reading is higher than the left light level reading.

```
#--------Your code below-----------
while True:
    if Ed.ReadRightLightLevel() > Ed.ReadLeftLightLevel():
        Ed.PlayBeep()
        Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
```

Watch out for:

Edison reads the light level using an analogue-to-digital converter, so the returned value can range from 0 to 1,023.

Ed.ReadLineTracker()

EDPY FUNCTION

Compatible with V1 V2 V3

Parameters:

N/A

Returns:

The current light level of the line-tracking sensor.

Explanation:

Reads the current light level of the line-tracking sensor.

Examples:

Play a beep whenever the line-tracking light level drops below a threshold buffer value.

```
#--------Your code below-----------
Ed.LineTrackerLed(Ed.ON)
white = Ed.ReadLineTracker()
buffer = 150
while True:
    if Ed.ReadLineTracker() < (white - buffer):
        Ed.PlayBeep()
        Ed.TimeWait(1, Ed.TIME_SECONDS)
```

Watch out for:

Edison reads the light level using an analogue-to-digital converter, so the returned value can range from 0 to 1,023.

This function can be used without turning the line-tracking LED on; however, the returned values will be much lower and harder to distinguish. Turning the line-tracking LED on is highly recommended before using this function.

Ed.ReadCountDown()

EDPY FUNCTION

Compatible with V1 V2 V3

Ed.ReadCountDown(units)

Parameters:

Units:

- Ed.TIME_MILLISECONDS - Read the number of milliseconds left in the countdown timer.
- Ed.TIME_SECONDS - Read the number of seconds left in the countdown timer.

Returns:

The number of seconds or milliseconds left on the countdown timer.

Explanation:

Reads the number of seconds or milliseconds left on the countdown timer.

Examples:

Flash the left LED for 3,000 milliseconds.

```
Ed.StartCountDown(3100, Ed.TIME_MILLISECONDS)
while Ed.ReadCountDown(Ed.TIME_MILLISECONDS) > 100:
    Ed.LeftLed(Ed.ON)
    Ed.TimeWait(50, Ed.TIME_MILLISECONDS)
    Ed.LeftLed(Ed.OFF)
    Ed.TimeWait(50, Ed.TIME_MILLISECONDS)
```

Watch out for:

The timer counts down only; it cannot be used as a count-up stopwatch.

The timer is not a clock and cannot tell time.

When reading the timer in seconds, the read function will return an integer value, not a floating-point value. Therefore, Ed.ReadCountDown(Ed.TIME_SECONDS) will return 0 if the time left in the countdown is less than 1 second (for example, 0.8 seconds). Consider using milliseconds if working with sub-second values.

Ed.ReadMusicEnd()

EDPY FUNCTION

Compatible with V1 V2 V3

Parameters:

N/A

Returns:

- Ed.MUSIC_FINISHED - Edison has finished playing the tune, tone or beep.
- Ed.MUSIC_NOT_FINISHED - Edison is still playing a tune, tone or beep.

Explanation:

Reads the current state of the sound being played from Edison's buzzer.

Examples:

Play a simple tune.

```
#--------Your code below-----------
simple = Ed.TuneString(25, "d4e4f4e4d4c4n2d4e4f4e4d1z")
Ed.PlayTune(simple)
while Ed.ReadMusicEnd() == Ed.MUSIC_NOT_FINISHED:
    pass
```

Watch out for:

All tune strings must end with the character "z".

Edison plays sounds in the background. As such, Edison moves on to the next line of code as soon as the sound starts. To make Edison wait for the sound to finish, use Ed.ReadMusicEnd() in a loop.

You can change the speed at which your tune plays by changing Ed.Tempo in the setup code.

Ed.ReadDriveLoad()

EDPY FUNCTION

Compatible with V1 V2 V3

Parameters:

N/A

Returns:

- Ed.DRIVE_NO_STRAIN - Edison's wheels are turning correctly.
- Ed.DRIVE_STRAINED - Edison's wheels are not turning.

Explanation:

Reads the current state of Edison's drive strain.

Examples:

Drive until Edison detects drive strain.

```
#--------Your code below-----------
Ed.Drive(Ed.FORWARD, 5, Ed.DISTANCE_UNLIMITED)
while Ed.ReadDriveLoad() == Ed.DRIVE_NO_STRAIN:
    pass
Ed.Drive(Ed.STOP, 1, 1)
```

Watch out for:

When driving with both wheels, detected strain on either wheel will trigger the Ed.DRIVE_STRAINED condition. When driving with only one wheel, only strain detected on the moving wheel will trigger the Ed.DRIVE_STRAINED condition.

Ed.ReadDistance()

EDPY FUNCTION

Compatible with V2 V3

Ed.ReadDistance(side)

Parameters:

Side:

- Ed.MOTOR_LEFT - Selects the left distance register.
- Ed.MOTOR_RIGHT - Selects the right distance register.

Returns:

The number of ticks remaining on the left or right distance register.

Explanation:

Reads the number of ticks remaining on the left or right distance register.

Examples:

Drive for 40 ticks, then beep.

```
#--------Your code below-----------
Ed.Drive(Ed.FORWARD, 5, Ed.DISTANCE_UNLIMITED)
Ed.SetDistance(Ed.MOTOR_LEFT, 40)
Ed.SetDistance(Ed.MOTOR_RIGHT, 40)
while Ed.ReadDistance(Ed.MOTOR_LEFT) > 0:
    pass
Ed.PlayBeep()
```

Watch out for:

This function is only compatible with Edison V2 and V3 robots.

The function reads values in ticks, where a tick is 1.25 mm.

This read function is used with the Ed.SetDistance() function.

Ed.ReadRound()

EDPY FUNCTION

Compatible with V3

Parameters:

N/A

Returns:

- Ed.KEYPAD_NONE - Edison's round button has not been pressed.
- Ed.KEYPAD_PRESSED - Edison's round button has been pressed.

Explanation:

Reads Edison's round button state, returns its value, and then clears Edison's round button register. Edison's round button state is set whenever the round button is pressed, regardless of what the code is doing at the time.

Examples:

Play a beep when the round button is pressed.

```
#--------Your code below-----------
while True:
    if Ed.ReadRound() != Ed.KEYPAD_NONE:
        Ed.PlayBeep()
```

Wait for the round button to be pressed, then beep.

```
#--------Your code below-----------
#clear previous presses
Ed.ReadRound()
while Ed.ReadRound() != Ed.KEYPAD_PRESSED:
    pass
Ed.PlayBeep()
```

Watch out for:

Edison is constantly updating the keypad state. This function will read the state. As such, the function may read a keypad press from before the read function is called in your code.

When using a read function inside a loop, include a read function outside of the loop before the loop to clear any previous data.

Ed.ReadTriangle()

EDPY FUNCTION

Compatible with V3

Parameters:

N/A

Returns:

- Ed.KEYPAD_NONE - Edison's triangle button has not been pressed.
- Ed.KEYPAD_PRESSED - Edison's triangle button has been pressed.

Explanation:

Reads Edison's triangle button state, returns its value, and then clears Edison's triangle button register. Edison's triangle button state is set whenever the triangle button is pressed, regardless of what the code is doing at the time.

Examples:

Play a beep when the triangle button is pressed.

```
#--------Your code below-----------
while True:
    if Ed.ReadTriangle() != Ed.KEYPAD_NONE:
        Ed.PlayBeep()
```

Wait for the triangle button to be pressed, then beep.

```
#--------Your code below-----------
#clear previous presses
Ed.ReadTriangle()
while Ed.ReadTriangle() != Ed.KEYPAD_PRESSED:
    pass
Ed.PlayBeep()
```

Watch out for:

Edison is constantly updating the keypad state. This function will read the state. As such, the function may read a keypad press from before the read function is called in your code.

When using a read function inside a loop, include a read function outside of the loop before the loop to clear any previous data.

Ed.SetDistance()

EDPY FUNCTION

Compatible with V2 V3

`Ed.SetDistance(side, ticks)`

Parameters:

Side:

- Ed.MOTOR_LEFT - Selects the left distance register.
- Ed.MOTOR_RIGHT - Selects the right distance register.

Ticks:

A positive integer that sets the number of ticks in the distance register. The maximum value is 32,767. One encoder tick represents approximately 1.25 mm of wheel travel.

Returns:

N/A

Explanation:

Sets one of Edison's distance registers, turning an unlimited drive into a distance-limited drive. This provides access to the distance registers in ticks, Edison's internal distance measurement.

Examples:

Drive for 40 ticks, then beep.

```
#--------Your code below-----------
Ed.Drive(Ed.FORWARD, 5, Ed.DISTANCE_UNLIMITED)
Ed.SetDistance(Ed.MOTOR_LEFT, 40)
Ed.SetDistance(Ed.MOTOR_RIGHT, 40)
while Ed.ReadDistance(Ed.MOTOR_LEFT) > 0:
    pass
Ed.PlayBeep()
```

Watch out for:

This function is only compatible with Edison V2 and V3.

The function sets values in ticks, where a tick is 1.25 mm.

Ed.ResetDistance()

EDPY FUNCTION

Compatible with V2 V3

Parameters:

N/A

Returns:

N/A

Explanation:

Resets the number of ticks remaining on both the left and right distance registers to zero.

Examples:

Drive for 40 ticks or until an obstacle is encountered, then beep.

```
#--------Your code below-----------
Ed.ObstacleDetectionBeam(Ed.ON)
Ed.Drive(Ed.FORWARD, 5, Ed.DISTANCE_UNLIMITED)
Ed.SetDistance(Ed.MOTOR_LEFT, 40)
Ed.SetDistance(Ed.MOTOR_RIGHT, 40)
while Ed.ReadDistance(Ed.MOTOR_LEFT) > 0:
    if Ed.ReadObstacleDetection() > Ed.OBSTACLE_NONE:
        Ed.ResetDistance()
Ed.PlayBeep()
```

Watch out for:

This function is only compatible with Edison V2 and V3 robots.

Distance-register values are measured in ticks, where a tick is 1.25 mm.

Ed.AddLogEntry()

EDPY FUNCTION

Compatible with V3

Ed.AddLogEntry(logVal1, logVal2, ..., logVal32)

Parameters:

logVal1:

An integer from -32,767 to 32,767, or text of up to 126 characters, to save into Edison V3's data log.

logVal2 to logVal32 (optional):

Up to 31 more values (integers from -32,767 to 32,767, or text of up to 126 characters) to save in the same log entry. A maximum of 32 values can be saved in one call.

Returns:

N/A

Explanation:

Saves one or more values into Edison V3's data log, together with a timestamp.

Examples:

Save a light reading every second for 10 seconds.

```
#--------Your code below-----------
#loop 10 times
for i in range(10):
    #save the light level to the data log
    Ed.AddLogEntry(Ed.ReadLeftLightLevel())
    #wait one second between readings
    Ed.TimeWait(1000, Ed.TIME_MILLISECONDS)
```

Save several values in one entry whenever an IR message is received.

```
#--------Your code below-----------
while True:
    #check for an IR message without clearing it
    if Ed.ReadIRData(False) != 0:
        #save the message, the light level and a random number
        Ed.AddLogEntry(Ed.ReadIRData(), Ed.ReadLeftLightLevel(), Ed.Random(1, 10))
```

Watch out for:

Edison V3 keeps track of the current time so it can save it with each log entry. The time is updated automatically every time a program is downloaded.

The data log is kept through program downloads, firmware updates and normal power cycles. If the log is not cleared, it can contain data from several program runs. Clear it with the "clear data log" button in the USB box.

The data log holds about 53 KB. For example, one light reading plus its timestamp per entry stores a little over 4,500 entries. When the log is full, Edison plays a fail tone and the program ends.

All values passed in one call are saved under the same timestamp. For the most useful data, use the same number of values in every Ed.AddLogEntry() call in a program.

Ed.ON

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.ON = 1

Used in:

- Ed.LeftLed()
- Ed.RightLed()
- Ed.ObstacleDetectionBeam()
- Ed.LineTrackerLed()

Ed.OFF

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.OFF = 0

Used in:

- Ed.LeftLed()
- Ed.RightLed()
- Ed.ObstacleDetectionBeam()
- Ed.LineTrackerLed()

Ed.NOTE_#

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.NOTE_A_6 = 18181
Ed.NOTE_A_SHARP_6 = 17167
Ed.NOTE_B_6 = 16202
Ed.NOTE_C_7 = 15289
Ed.NOTE_C_SHARP_7 = 14433
Ed.NOTE_D_7 = 13622
Ed.NOTE_D_SHARP_7 = 12856
Ed.NOTE_E_7 = 12135
Ed.NOTE_F_7 = 11457
Ed.NOTE_F_SHARP_7 = 10814
Ed.NOTE_G_7 = 10207
Ed.NOTE_G_SHARP_7 = 9632
Ed.NOTE_A_7 = 9090
Ed.NOTE_A_SHARP_7 = 8581
Ed.NOTE_B_7 = 8099
Ed.NOTE_C_8 = 7644
Ed.NOTE_REST = 0
Ed.NOTE_SIXTEENTH = 125
Ed.NOTE_EIGHTH = 250
Ed.NOTE_QUARTER = 500

Ed.NOTE_HALF = 1000
Ed.NOTE_WHOLE = 2000

Used with:

Ed.PlayTone()

Ed.NOTE_A_6 - Play a low A.
Ed.NOTE_A_SHARP_6 - Play a low A sharp.
Ed.NOTE_B_6 - Play a low B.
Ed.NOTE_C_7 - Play a C.
Ed.NOTE_C_SHARP_7 - Play a C sharp.
Ed.NOTE_D_7 - Play a D.
Ed.NOTE_D_SHARP_7 - Play a D sharp.
Ed.NOTE_E_7 - Play an E.
Ed.NOTE_F_7 - Play an F.
Ed.NOTE_F_SHARP_7 - Play an F sharp.
Ed.NOTE_G_7 - Play a G.
Ed.NOTE_G_SHARP_7 - Play a G sharp.
Ed.NOTE_A_7 - Play an A.
Ed.NOTE_A_SHARP_7 - Play an A sharp.
Ed.NOTE_B_7 - Play a B.
Ed.NOTE_C_8 - Play a high C.
Ed.NOTE_REST - Play a rest.
Ed.NOTE_SIXTEENTH - Play a note for 125 milliseconds.
Ed.NOTE_EIGHTH - Play a note for 250 milliseconds.
Ed.NOTE_QUARTER - Play a note for 500 milliseconds.
Ed.NOTE_HALF - Play a note for 1,000 milliseconds.
Ed.NOTE_WHOLE - Play a note for 2,000 milliseconds.

Ed.TEMPO_#

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.TEMPO_VERY_SLOW	=	1000
Ed.TEMPO_SLOW	=	500
Ed.TEMPO_MEDIUM	=	250
Ed.TEMPO_FAST	=	70
Ed.TEMPO_VERY_FAST	=	1

Used with:

- Ed.Tempo

Ed.FORWARD

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.FORWARD = 1

Used in:

- Ed.Drive()
- Ed.DriveLeftMotor()
- Ed.DriveRightMotor()

Ed.BACKWARD

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.BACKWARD = 2

Used in:

- Ed.Drive()
- Ed.DriveLeftMotor()
- Ed.DriveRightMotor()

Ed.FORWARD_RIGHT

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.FORWARD_RIGHT = 3

Used in:

- Ed.Drive()

Ed.BACKWARD_RIGHT

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.BACKWARD_RIGHT = 4

Used in:

- Ed.Drive()

Ed.FORWARD_LEFT

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.FORWARD_LEFT = 5

Used in:

- Ed.Drive()

Ed.BACKWARD_LEFT

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.BACKWARD_LEFT = 6

Used in:

- Ed.Drive()

Ed.SPIN_#

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.SPIN_RIGHT	=	7
Ed.SPIN_LEFT	=	8

Used in:

- Ed.Drive()

Ed.STOP

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.STOP = 0

Used in:

- Ed.Drive()
- Ed.DriveLeftMotor()
- Ed.DriveRightMotor()

Example:

Drive forwards at speed 6 until the IR message 10 is received from another Edison robot.

```
#--------Your code below-----------
#clear IR data register
Ed.ReadIRData()
Ed.Drive(Ed.FORWARD, Ed.SPEED_6, Ed.DISTANCE_UNLIMITED)
while Ed.ReadIRData() != 10:
    pass
Ed.Drive(Ed.STOP, 0, 0)
```

Watch out for:

An Ed.Drive() call with Ed.STOP as the direction ignores the speed and distance values. It is best practice to set them to 0 when using Ed.STOP.

Ed.SPEED_#

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.SPEED_1	=	1
Ed.SPEED_2	=	2
Ed.SPEED_3	=	3
Ed.SPEED_4	=	4
Ed.SPEED_5	=	5
Ed.SPEED_6	=	6
Ed.SPEED_7	=	7
Ed.SPEED_8	=	8
Ed.SPEED_9	=	9
Ed.SPEED_10	=	10
Ed.SPEED_FULL	=	0

Used in:

- Ed.Drive()
- Ed.DriveLeftMotor()
- Ed.DriveRightMotor()

Ed.DISTANCE_UNLIMITED

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.DISTANCE_UNLIMITED = 1073741823 (Edison V3)
Ed.DISTANCE_UNLIMITED = 0 (Edison V1 and V2)

Used in:

- Ed.Drive()
- Ed.DriveLeftMotor()
- Ed.DriveRightMotor()

Ed.MOTOR_#

CONSTANT

Compatible with V2 V3

Value:

Ed.MOTOR_LEFT	=	0
Ed.MOTOR_RIGHT	=	1

Used in:

- Ed.ReadDistance()
- Ed.SetDistance()

Ed.TIME_#

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.TIME_SECONDS	=	0
Ed.TIME_MILLISECONDS	=	1

Used in:

- Ed.StartCountDown()
- Ed.TimeWait()
- Ed.ReadCountDown()

Ed.OBSTACLE_#

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.OBSTACLE_NONE	=	0
Ed.OBSTACLE_RIGHT	=	8
Ed.OBSTACLE_LEFT	=	32
Ed.OBSTACLE_AHEAD	=	16

Used with:

- Ed.ReadObstacleDetection()

Ed.LINE_ON_#

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.LINE_ON_BLACK	=	1
Ed.LINE_ON_WHITE	=	0

Used with:

- Ed.ReadLineState()

Ed.KEYPAD_#

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.KEYPAD_NONE	=	0
Ed.KEYPAD_PRESSED	=	1 (Edison V3 only)
Ed.KEYPAD_TRIANGLE	=	1
Ed.KEYPAD_ROUND	=	4

Used with:

- Ed.ReadKeypad()

- Ed.ReadRound() (Edison V3 only)
- Ed.ReadTriangle() (Edison V3 only)

Ed.CLAP_#

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.CLAP_NOT_DETECTED	=	0
Ed.CLAP_DETECTED	=	4

Used in:

- Ed.ReadClapSensor()

Ed.DRIVE_#

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.DRIVE_STRAINED	=	1
Ed.DRIVE_NO_STRAIN	=	0

Used with:

- Ed.ReadDriveLoad()

Ed.MUSIC_#

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.MUSIC_FINISHED	=	1
Ed.MUSIC_NOT_FINISHED	=	0

Used with:

- Ed.ReadMusicEnd()

Ed.REMOTE_CODE_#

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.REMOTE_CODE_0	=	0
Ed.REMOTE_CODE_1	=	1
Ed.REMOTE_CODE_2	=	2
Ed.REMOTE_CODE_3	=	3
Ed.REMOTE_CODE_4	=	4
Ed.REMOTE_CODE_5	=	5
Ed.REMOTE_CODE_6	=	6
Ed.REMOTE_CODE_7	=	7
Ed.REMOTE_CODE_NONE	=	255
Ed.REMOTE_CODE_RELEASE	=	254

Used with:

- Ed.ReadRemote()

Ed.EVENT_#

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.EVENT_TIMER_FINISHED	=	0
Ed.EVENT_REMOTE_CODE	=	1
Ed.EVENT_IR_DATA	=	2
Ed.EVENT_CLAP_DETECTED	=	3
Ed.EVENT_OBSTACLE_ANY	=	4
Ed.EVENT_OBSTACLE_LEFT	=	5
Ed.EVENT_OBSTACLE_RIGHT	=	6
Ed.EVENT_OBSTACLE_AHEAD	=	7
Ed.EVENT_DRIVE_STRAIN	=	8
Ed.EVENT_KEYPAD_TRIANGLE	=	9
Ed.EVENT_KEYPAD_ROUND	=	10
Ed.EVENT_LINE_TRACKER_ON_WHITE	=	11
Ed.EVENT_LINE_TRACKER_ON_BLACK	=	12
Ed.EVENT_LINE_TRACKER_SURFACE_CHANGE	=	13
Ed.EVENT_TUNE_FINISHED	=	14

Used in:

- Ed.RegisterEventHandler()

Event meanings:

- Ed.EVENT_TIMER_FINISHED - Calls the function when the countdown timer finishes.

- Ed.EVENT_REMOTE_CODE - Calls the function when Edison receives a remote code.
- Ed.EVENT_IR_DATA - Calls the function when Edison receives data from another Edison.
- Ed.EVENT_CLAP_DETECTED - Calls the function when Edison detects a clap.
- Ed.EVENT_OBSTACLE_ANY - Calls the function when Edison detects any obstacle.
- Ed.EVENT_OBSTACLE_LEFT - Calls the function when Edison detects an obstacle to the left.
- Ed.EVENT_OBSTACLE_RIGHT - Calls the function when Edison detects an obstacle to the right.
- Ed.EVENT_OBSTACLE_AHEAD - Calls the function when Edison detects an obstacle straight ahead.
- Ed.EVENT_DRIVE_STRAIN - Calls the function when Edison detects drive strain.
- Ed.EVENT_KEYPAD_TRIANGLE - Calls the function when Edison detects a triangle-button press.
- Ed.EVENT_KEYPAD_ROUND - Calls the function when Edison detects a round-button press.
- Ed.EVENT_LINE_TRACKER_ON_WHITE - Calls the function when Edison detects a white surface under the line tracker.
- Ed.EVENT_LINE_TRACKER_ON_BLACK - Calls the function when Edison detects a black surface under the line tracker.
- Ed.EVENT_LINE_TRACKER_SURFACE_CHANGE - Calls the function when Edison detects a surface change under the line tracker.
- Ed.EVENT_TUNE_FINISHED - Calls the function when Edison finishes playing a tune.

Ed.CM

CONSTANT

Compatible with V2 V3

Value:

Ed.CM = 0

Used with:

- Ed.DistanceUnits
  - Sets distance units to centimetres.

Ed.INCH

CONSTANT

Compatible with V2 V3

Value:

Ed.INCH = 1

Used with:

- Ed.DistanceUnits
  - Sets distance units to inches.

Ed.TIME

CONSTANT

Compatible with V1 V2 V3

Value:

Ed.TIME = 2

Used with:

- Ed.DistanceUnits
  - Sets distance units to milliseconds.

Ed.TICKS

CONSTANT Compatible with V3

Value:

Ed.TICKS = 3

Used with:

- Ed.DistanceUnits
  - Sets drive distances to encoder ticks. Edison V3 drives approximately 1.25 mm per tick.
  - Distances stay in ticks for turns and spins too. For example, Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_5, 90) makes Edison spin about 180 degrees.

Ed.V1

CONSTANT Compatible with V1

Value:

Ed.V1 = 1

Used with:

- Ed.EdisonVersion
  - Makes the code compile for Edison V1 robots.

Ed.V2

CONSTANT Compatible with V2

Value:

Ed.V2 = 2

Used with:

- Ed.EdisonVersion
  - Makes the code compile for Edison V2 robots.

Ed.V3

CONSTANT Compatible with V3

Value:

Ed.V3 = 3

Used with:

- Ed.EdisonVersion
  - Makes the code compile for Edison V3 robots.

Ed.Tempo

EDPY VARIABLE

Compatible with V1 V2 V3

Can be set to:

- Ed.TEMPO_VERY_SLOW - Play tunes very slowly.
- Ed.TEMPO_SLOW - Play tunes slowly.
- Ed.TEMPO_MEDIUM - Play tunes normally.
- Ed.TEMPO_FAST - Play tunes fast.
- Ed.TEMPO_VERY_FAST - Play tunes very fast.

Explanation:

Changes how fast or slow Edison plays a tune.

Examples:

Play a simple tune very fast.

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_VERY_FAST

#--------Your code below-----------
simple = Ed.TuneString(25, "d4e4f4e4d4c4n2d4e4f4e4d1z")
Ed.PlayTune(simple)
while Ed.ReadMusicEnd() == Ed.MUSIC_NOT_FINISHED:
    pass
```

Watch out for:

Ed.Tempo must be set to one of the constants specified above.

Ed.Tempo can only be set once per program and must be set in the setup code.

Ed.Tempo is set to Ed.TEMPO_MEDIUM by default. This can be changed in the setup code section.

Ed.DistanceUnits

EDPY VARIABLE

Compatible with V1 V2 V3

Can be set to:

- Ed.CM - Set drive distances to centimetres.
- Ed.INCH - Set drive distances to inches.
- Ed.TIME - Set drive distances to milliseconds.
- Ed.TICKS - Set drive distances to encoder ticks (Edison V3 only).

Explanation:

Changes the units of the distance used in all drive functions in a program.

Examples:

Drive Edison forward for 3 cm at speed 5 (Edison V2 and V3 only).

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
Ed.Drive(Ed.FORWARD, Ed.SPEED_5, 3)
```

Drive Edison forward for 5 inches at speed 5 (Edison V2 and V3 only).

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.INCH
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
Ed.Drive(Ed.FORWARD, Ed.SPEED_5, 5)
```

Drive Edison forward for 2,000 milliseconds at speed 7.

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.TIME
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
Ed.Drive(Ed.FORWARD, Ed.SPEED_7, 2000)
```

Drive Edison forward for 50 ticks (approximately 62.5 mm) at speed 7 (Edison V3 only).

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.TICKS
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
Ed.Drive(Ed.FORWARD, Ed.SPEED_7, 50)
```

Watch out for:

Ed.DistanceUnits must be set to one of the constants specified above.

Ed.DistanceUnits can only be set once per program and must be set in the setup code.

For Edison V2 and V3, Ed.DistanceUnits is set to Ed.CM by default. This can be changed in the setup code.

When Ed.DistanceUnits is set to Ed.TICKS, Edison will drive approximately 1.25 mm per tick. Edison's physical movements are accurate within plus or minus 5 ticks.

When Ed.DistanceUnits is set to Ed.TICKS, distance is always measured in ticks, even when Edison is turning or spinning. For example, Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_5, 90) makes Edison spin about 180 degrees.

Edison V1 robots support only Ed.DistanceUnits = Ed.TIME.

Ed.EdisonVersion

EDPY VARIABLE

Compatible with V1 V2 V3

Can be set to:

- Ed.V1 - Edison V1 robot.
- Ed.V2 - Edison V2 robot.
- Ed.V3 - Edison V3 robot.

Explanation:

Sets the version of Edison for which the code is compiled. Edison V2 and V3 robots contain encoders, allowing them to drive for very precise distances.

Edison V2 and V3 robots can be set to drive in centimetres or inches and can read from and write to the distance registers.

Examples:

Edison V1 setup.

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V1
Ed.DistanceUnits = Ed.TIME
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
```

Edison V2 setup.

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V2
Ed.DistanceUnits = Ed.TIME
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
```

Edison V3 setup.

```
#-------------Setup----------------
import Ed
Ed.EdisonVersion = Ed.V3
Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
```

Watch out for:

Ed.EdisonVersion must be set to one of the constants specified above.

Ed.EdisonVersion can only be set once per program and must be set in the setup code.

Ed.EdisonVersion is set using the version pop-up box that opens when EdPy launches. It can also be changed in the setup code.

Edison V1 robots can only use Ed.DistanceUnits = Ed.TIME.

abs()

PYTHON LANGUAGE FEATURE

Compatible with V1 V2 V3

abs(int)

Parameters:

int:

An integer from -32,767 to 32,767.

Returns:

The mathematical absolute value of the input.

Explanation:

Returns the non-negative magnitude of the input value.

Examples:

Convert -30 to 30 and confirm the result with a beep.

```
#--------Your code below-----------
x = -30
x = abs(x)
if x == 30:
    Ed.PlayBeep()
```

Watch out for:

abs() is a native Python function. As such, it does not need the "Ed." prefix.

len()

PYTHON LANGUAGE FEATURE

Compatible with V1 V2 V3

len(list)

Parameters:

List:

An EdPy list.

Returns:

The number of elements in the EdPy list.

Explanation:

Returns the number of elements in the given list.

Examples:

Loop through a created list and beep when a 2 is found.

```
#--------Your code below-----------
exampleList = Ed.List(6, [1, 2, 3, 3, 2, 1])
listSize = len(exampleList)
for x in range(listSize):
    if exampleList[x] == 2:
        Ed.PlayBeep()
        Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
```

Watch out for:

len() returns the value you entered as the size of the list when it was created, even if you have not assigned values to all of the integers.

range()

PYTHON LANGUAGE FEATURE

Compatible with V1 V2 V3

`range(limit)`

Parameters:

Limit:

The value to count up to.

Returns:

The integer values from 0 to one less than the limit.

Explanation:

Generates a sequence of integers for use in a for loop.

Examples:

Loop through a created list and beep when a 2 is found.

```
#--------Your code below-----------
exampleList = Ed.List(6, [1, 2, 3, 3, 2, 1])
listSize = len(exampleList)
for x in range(listSize):
    if exampleList[x] == 2:
        Ed.PlayBeep()
        Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
```

Watch out for:

The limit is not included in the sequence. For example, range(5) produces 0, 1, 2, 3 and 4.

True

PYTHON LANGUAGE FEATURE

Compatible with V1 V2 V3

Value:

True = 1

Used with:

- while

- if

Explanation:

A Boolean value used when a condition is true.

Examples:

Loop forever flashing an LED.

```
#--------Your code below-----------
while True:
    Ed.RightLed(Ed.ON)
    Ed.TimeWait(50, Ed.TIME_MILLISECONDS)
    Ed.RightLed(Ed.OFF)
    Ed.TimeWait(50, Ed.TIME_MILLISECONDS)
```

False

PYTHON LANGUAGE FEATURE

Compatible with V1 V2 V3

Value:

False = 0

Used with:

- while
- if

Explanation:

A Boolean value used when a condition is false.

Examples:

Skip a loop.

```
#--------Your code below-----------
while False:
    Ed.RightLed(Ed.ON)
    Ed.TimeWait(50, Ed.TIME_MILLISECONDS)
    Ed.RightLed(Ed.OFF)
    Ed.TimeWait(50, Ed.TIME_MILLISECONDS)
```

import

PYTHON LANGUAGE FEATURE

Compatible with V1 V2 V3

Can be imported:

- Ed

Explanation:

Imports the Ed library so the program can use Edison functions, constants and variables.

def

PYTHON LANGUAGE FEATURE

Compatible with V1 V2 V3

Explanation:

Defines a new user function. A function must be defined before the main body of the code and can be used anywhere in the main body of the code.

User functions can be defined to accept input values and return values to the main code.

Examples:

A function that waits for an obstacle to be detected ahead of Edison.

```
#--------Your code below-----------
#--------Your function def here-----------
def waitObs():
    #clear obstacle-detection register
    Ed.ReadObstacleDetection()
    while Ed.ReadObstacleDetection() != Ed.OBSTACLE_AHEAD:
        Ed.TimeWait(50, Ed.TIME_MILLISECONDS)
#-------- Your main body code here -----------
Ed.ObstacleDetectionBeam(Ed.ON)
waitObs()
Ed.PlayBeep()
```

A function that accepts an input and returns a value based on the input.

```
#--------Your code below-----------
#--------Your function def here-----------
def isTwo(number):
    if number == 2:
        return True
    else:
        return False
#-------- Your main body code here -----------
for x in range(10):
    if isTwo(x):
        Ed.PlayBeep()
    else:
        Ed.LeftLed(Ed.ON)
    Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
    Ed.LeftLed(Ed.OFF)
    Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
```

Watch out for:

Functions must be defined with def before they are called. Place function definitions near the top of the program, as shown in the examples.

pass

PYTHON LANGUAGE FEATURE

Compatible with V1 V2 V3

Used with:

- if
- while
- for

Explanation:

Used to do nothing in an if statement or loop.

Examples:

Flash the right LED on every iteration except when x equals 3.

```
#--------Your code below-----------
for x in range(10):
    if x == 3:
        pass
    else:
        Ed.RightLed(Ed.ON)
        Ed.TimeWait(100, Ed.TIME_MILLISECONDS)
        Ed.RightLed(Ed.OFF)
        Ed.TimeWait(100, Ed.TIME_MILLISECONDS)
```

Keep looping until a key is pressed; that is, wait for a key press.

```
#--------Your code below-----------
#clear previous presses
Ed.ReadKeypad()
while Ed.ReadKeypad() == Ed.KEYPAD_NONE:
    pass
Ed.PlayBeep()
```

for

PYTHON LANGUAGE FEATURE

Compatible with V1 V2 V3

Explanation:

Used with the range() function to repeat indented code a set number of times.

Examples:

Flash the right LED 10 times.

```
#--------Your code below-----------
for x in range(10):
    Ed.RightLed(Ed.ON)
    Ed.TimeWait(100, Ed.TIME_MILLISECONDS)
    Ed.RightLed(Ed.OFF)
    Ed.TimeWait(100, Ed.TIME_MILLISECONDS)
```

Set up a list with all elements set to zero.

```
#--------Your code below-----------
zeros = Ed.List(5)
for x in range(5):
    zeros[x] = 0
```

while

PYTHON LANGUAGE FEATURE

Compatible with V1 V2 V3

Explanation:

Used to repeat indented code until the condition following the while evaluates to false.

Examples:

Flash the right LED 10 times.

```
#--------Your code below-----------
x = 0
while x < 10:
    Ed.RightLed(Ed.ON)
    Ed.TimeWait(100, Ed.TIME_MILLISECONDS)
    Ed.RightLed(Ed.OFF)
    Ed.TimeWait(100, Ed.TIME_MILLISECONDS)
    x = x + 1
```

Loop until an obstacle is detected in front of Edison.

```
#--------Your code below-----------
Ed.ObstacleDetectionBeam(Ed.ON)
#clear obstacle-detection register
Ed.ReadObstacleDetection()
while Ed.ReadObstacleDetection() != Ed.OBSTACLE_AHEAD:
    pass
Ed.PlayBeep()
```

if / elif / else

PYTHON LANGUAGE FEATURE

Compatible with V1 V2 V3

if

Explanation:

Used to run indented code if the condition following the if evaluates to true.

elif

Explanation:

Runs its indented block when all preceding if and elif conditions are false and its own condition evaluates to true.

else

Explanation:

Runs its indented block when all preceding if and elif conditions are false.

Examples:

Loop 10 times. Flash the LED on every iteration except the fifth, when Edison beeps instead.

```
#--------Your code below-----------
testVal = 1
while True:
    if testVal == 11:
        break
    elif testVal == 5:
        Ed.PlayBeep()
        Ed.TimeWait(1000, Ed.TIME_MILLISECONDS)
    else:
        Ed.LeftLed(Ed.ON)
        Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
        Ed.LeftLed(Ed.OFF)
        Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
    testVal = testVal + 1
```

class

PYTHON LANGUAGE FEATURE

Compatible with V1 V2 V3

Explanation:

Used to define a new type of Python object.

A Python object is a collection of variables and functions that act on those variables. Multiple objects can be created from a single class definition. Changing the values in one created object does not change the values in any other created object.

Examples:

Create a single object.

```
#--------Your code below-----------

class notes:
    def __init__(self):
        self.i = 1000
        self.b = 2000
    def changeB(self, newB):
        self.b = newB
    def changeI(self, newI):
        self.i = newI
x = notes()
Ed.PlayTone(x.i, x.b)
while Ed.ReadMusicEnd() == Ed.MUSIC_NOT_FINISHED:
    pass
```

Create two objects based on a single class and change the values in one of the objects.

```
#--------Your code below-----------

class notes:
    def __init__(self):
        self.i = 1000
        self.b = 2000
    def changeB(self, newB):
        self.b = newB
    def changeI(self, newI):
        self.i = newI
x = notes()
y = notes()
y.changeB(3000)
y.changeI(32000)
Ed.PlayTone(x.i, x.b)
while Ed.ReadMusicEnd() == Ed.MUSIC_NOT_FINISHED:
    pass
Ed.TimeWait(10, Ed.TIME_MILLISECONDS)
Ed.PlayTone(y.i, y.b)
```

Watch out for:

Functions defined in a class must use self as the first parameter. This parameter is omitted when the function is called from the main program.

Classes use a function named __init__(), referred to as the constructor, which runs when a new object is created. The constructor sets the object's initial variable values and can accept parameters that are supplied when the object is created.

```
#--------Your code below-----------

class notes:
    def __init__(self, newI, newB):
        self.i = newI
        self.b = newB
    def changeB(self, newB):
        self.b = newB
    def changeI(self, newI):
        self.i = newI
x = notes(1000, 2000)
Ed.PlayTone(x.i, x.b)
while Ed.ReadMusicEnd() == Ed.MUSIC_NOT_FINISHED:
    pass
```

return

PYTHON LANGUAGE FEATURE

Compatible with V1 V2 V3

Explanation:

Used to return a value from a function.

In most EdPy programs, a function returns a number or True/False.

Examples:

A function that accepts an input and returns a value based on the input.

```
#--------Your code below-----------

def isTwo(number):
    if number == 2:
        return True
    else:
        return False
for x in range(10):
    if isTwo(x):
        Ed.PlayBeep()
    else:
        Ed.LeftLed(Ed.ON)
    Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
    Ed.LeftLed(Ed.OFF)
    Ed.TimeWait(500, Ed.TIME_MILLISECONDS)
```