r-one Python Library Reference

The rone robot contains a number of different libraries to move the robot, to read from sensors and to help keep track of information. This document gives a short overview of the functions you may want to use in this class.

You must import a library to use the functions inside. In order to import a module, use the command **import** module, like so:

```
import module
module.function()
```

If you want to learn more about modules and how to use them, refer to the Python documentation: https://docs.python.org/2/tutorial/modules.html

1 Built-in Functions

These functions are built-in to the Python interpreter on your robot. They do not require importing any package. They are called using their name, e.g. abs(2).

- abs(n): Returns the absolute value of n
- str(n): Returns the string representation of n
- dir(): Returns a list of the defined variable names in the current namespace.
- dir(n): Returns a list of the defined variable names in the given package n.
- len(l): Returns the number of elements in list or tuple l or the number of characters in the string l.
- chr(n): Returns a one-character string containing the nth ASCII character.
- ord(s): Returns the ordinal of the one-character string s in the ASCII character set. (The inverse of chr(n))
- int(n): Returns the greatest integer less than or equal to n.
- float(n): Returns the floating point representation of n.
- pow(b, e): Returns b raised to the power of e.
- enumerate(l): Returns a list of tuples containing the index of each element in list l with the element itself.
- hex(): Returns the hexadecimal representation of n in a string.
- sum(): Returns the arithmetic sum of the elements of l.
- range(a): Returns a list of integers from 0 to a-1, inclusive.
- range(a, b): Returns a list of integers from a to b-1, inclusive.
- range(a, b, c): Returns a list of integers from a to b-1, inclusive, incrementing by c at each step.
- rand(): Returns a pseudo-random integer.
- filter(f, l): Filters the elements of iterable data structure l by function f.
- map(f, l): Maps the elements of iterable data structure l using function f.

2 r-one Library

The r-one robots come preloaded with a module named rone that includes functions for accessing the various peripherals of the robot. These include motors, encoders, accelerometer, gyroscope, LEDs, radio, infrared communications, audio playback, and buttons. This document outlines the functions and their usage.

- get_id(): Retrieves the unique ID of the robot.

2.1 Motor Functions

The motors are specified through the string keys ”l” or ”r” for the left and right motors respectively.

- motor_set_pwm(motor, dutyCycle): Sets a PWM signal on the specified motor for two seconds on a robot. Duty cycles are between -100 and 100. To stop use 0, to go in reverse use a negative value and to go forward use positive values.
- motor_brake(motor): Causes the motor specified to brake.

2.2 LED Functions

The LED color parameter can either be the colors ”red”, ”green”, or ”blue”, or the first letter of the colors for brevity, i.e. ”r”, ”g”, or ”b”.

- led_set(color, index, brightness): Sets an LED to the specified brightness. Brightness values can range from 0 to 127. 127 is extremely bright. Index is from 0 to 4 starting at the bottom and moving counterclockwise on the robot for red, green, or blue.
- led_set_group(colors, brightness): This functions sets a whole group of LEDs to the specified brightness. Color may be given as a single string (e.g. led_set_group('r', 100)) or a list of LED groups to set (e.g. led_set_group(['r', 'g'], 50)).

2.3 IR Communication Functions

The angular positions of the IR receivers is found in the dictionary _ir_comms_locations, where the angular position of receiver i can be found in radians.

- ir_comms_send_message(): Transmits the robot's ID on the infrared emitters. Each emitter sends a different ”orientation bit” that allows receivers to tell which transmitter on the transmitting robot sent the message. This method can be combined with the radio system to send data to other robots by sending a radio message tagged with the sender's ID and payload data.
- (received_id, receivers, transmitters, distance) = ir_comms_get_message(): Reads and clears the most recent IR message. Returns a tuple containing: 1. The received robot ID, 2. A bit-array of receivers the message was received on, 3. a bit-array of transmitters that sent the message, 4. The estimated distance away the message was sent from. Returns None if no message was received.

2.4 Radio Functions

These radio functions are the basic functions needed to interact with the radio. The rest of the messages are for advanced functionality that is not necessary for basic use.

The radio messages are queued on the robot in a queue of size 10. If the queue is filled and a new message comes in, the oldest message is replaced by the new message. When you read a message, it is removed from the queue.

- radio_get_message_usr(): Retrieves a radio message sent by another user. Returns None if no messages available.

- radio_send_message(s): Broadcasts a string over the radio. Radio messages must be 32 characters or fewer.

2.4.1 Extended Radio Functions

- radio_get_message_usr_oldest(): Retrieves the oldest radio message inside the queue. Functionally the same as radio_get_message_usr.
- radio_get_message_usr_newest(): Retrieves the newest radio message inside the queue. Returns None if no messages available.
- radio_put_message_usr(s): Puts a message back on to the radio queue. Functionally the same as radio_put_message_usr_newest. Radio messages must be 32 characters or fewer.
- radio_put_message_usr_oldest(s): Puts a message back onto the head of the radio queue, in the oldest data position. If the queue is full, replaces the newest message in the queue.
- radio_put_message_usr_newest(s): Puts a message back onto the tail of the radio queue, in the newest data position. If the queue is full, replaces the oldest message in the queue.
- radio_flush_usr_queue(): Flushes the contents of the radio queue, deleting all messages inside.
- radio_get_message_nbr(): Retrieves the most recent special neighbor message for use in the neighbor subsystem. Neighbor messages are identified by the first byte of the message being the character "@" by default.
- radio_set_nbr_prefix(c): Sets the prefix for the neighbor message. By default, this is "@".

2.5 Sensors

- bump_sensors_get(): Returns a list of bump sensors the robot is registering being bumped on. These numbers can be translated into angular positions using the internal dictionary _bump_sensor_locations.
- accelerometer_get_value(axis): Returns the acceleration on either the X, Y, or Z axis. Axis values can be "x", "y", or "z". The Z axis points down when the robot is resting on a table and will reflect the acceleration of gravity. If all values are zero, the robot is either broken or falling.
- gyro_get_value(axis): Returns the angular acceleration of the robot. The axis value is the same as in accelerometer_get_value.
- light_sensor_get_value(light): Reads the value of a light sensor. Returns a value between 0 and 1023. For the v11 robots, light sensors may be "fr", "fl" or "r" for front right, front left or rear. For the v12 and v15 robots, light sensors may be "fr", "fl", "rr", or "rl" for front right, front left, rear right, or rear left.
- encoder_get_ticks(motor): Returns the ticks of the motor. Tick count rolls over at 65535. Motor can be "l" or "r".
- charger_get_status(): Returns whether or not the robot is allowed to charge. This is distinct from whether or not the robot is actually charging. Look at the charge light. If it is off, the robot is not charging because the motors are running or the batt is charged, or it is not plugged into usb. If the charge light is flashing, panic and turn off the robot. Something is wrong with the battery.
- button_get_value(button): Returns a 1 if a button is being pressed, otherwise it returns 0. Button values can be "r", "g", or "b" referring to the color of the LEDs surrounding each button.

2.6 Audio Functions

These functions utilize the speaker and audio subsystem on the robot. For reference, the notes that are playable are from the set ”C”, ”Cs”, ”Df”, ”D”, ”Ds”, ”Ef”, ”E”, ”F”, ”Fs”, ”Gf”, ”G”, ”Gs”, ”Af”, ”A”, ”As”, ”Bf”, ”B”, and ”Ch”. This is the chromatic ordering of notes beginning at C. Lowercase ”s” and ”f” specify sharp and flat respectively, and ”Ch” is the high C that completes the scale. Instruments playable are from the Midi defined patch list, which ranges from 1-127. You can read more about patches here: http://www.midi.org/techspecs/gm1sound.php

- play_note(note, octave, instrument, velocity, duration): Plays a note using the speaker on the robot. The note parameter is a string specifying which note you wish to play. The octave is the octave of the keyboard you wish to play in. For reference, middle C is in the 5th octave. Octaves can range from 0 to 9. Instrument is the midi patch you want to play the note with. The velocity is how loud the note will be played, and ranges from 0 to 127. A velocity of 0 turns off the note. Duration is how long you want to play the note in milliseconds. Notes will automatically shut off after this time.
- play_song(song, instrument): Plays a song concurrently with program operation. This means the song can play while your code executes normally. However, this means that the song will only play when code is executing, so it will not play normally at the interactive prompt. The song parameter is a string that is composed of notes written like so: (note, octave, velocity, duration) i.e. (C,5,50,1000), and time delays written like so (Z, duration) i.e. (Z,1000). An example song is given below. The instrument is the midi patch the song will be played with.
- audio_off(): Turns off all currently playing notes.

Example Song:
(D,5,95,1920)(A,4,95,1920)(F,4,95,1920)(Z,1920)(C,5,95,1920)(A,4,95,1920)(G,4,95,1920)(Z,1920)
(E,5,95,1920)(C,5,95,1920)(A,4,95,1920)

3 sys Library

- sleep(n): Waits for n milliseconds.
- time(): Returns the number of milliseconds the robot has been running.

4 math Library

- sqrt(n): Returns the square root of n. Requires a non-negative number.
- sin(n): Returns the sine (measured in radians) of n.
- cos(n): Returns the cosine (measured in radians) of n.
- tan(n): Returns the tangent (measured in radians) of n.
- asin(n): Returns the inverse sine (measured in radians) of n.
- acos(n): Returns the inverse cosine (measured in radians) of n.
- atan(n): Returns the inverse tangent (measured in radians) of n.
- atan2(y, x): Returns the inverse tangent (measured in radians) of y/x. This function considers the sign of both inputs, so atan2(-1, -1) returns a negative angle.
- pow(x, y): Returns x raised to the power y.
- radians(d): Returns the radian value of d (input in degrees).
- degrees(r): Returns the degree value of r (input in radians).
- modf(n): Returns the fractional and integer parts of x. Both results carry the sign of x. The

integer part is returned as a real.
- fabs(n): Returns the floating point absolute value of n.
- hypot(x, y): Returns the hypotenuse of the right triangle with edges a and b. Requires a non-negative number.
- pi: Returns pi accurate to 6 decimal places.
- e: Returns e accurate to 6 decimal places.

5 List Type

Read more about the list type in Python from their official documentation:
https://docs.python.org/2/tutorial/datastructures.html

However, the Python system on the robot does not have every method implemented for each data type. The methods available are listed below.
- append(o): Add o to the end of the list.
- count(v): Returns the number of times v is in the list.
- extend(s): Extends the list by appending all the items in the given list, string or tuple L.
- index(o): Return the index in the list of the first item whose value is o. Errors if there is no such item.
- insert(i, o): Inserts o at index i in the list.
- pop(): Removes the object at the end of the list and returns it.
- pop(i): Removes the object at index i and returns it.
- remove(v): Removes the first instance of v from the list.
- clear(): Removes all elements from the list.
- sort(): Sorts the elements of the list in order from lowest to highest.
- sort(c): Sorts the elements of the list using the comparison method c.
- sort(c, k): Sorts the elements of the list using the comparison method c and the key extractor k.
- sort(c, k, r): Sorts the elements of the list using the comparison method c, the key extractor k, and whether or not it should be reversed using boolean r.

6 String Type

- join(l): Concatenate a list, string, or tuple of words with intervening occurrences of l.
- split(): Splits a string into a list on all whitespace.
- split(s): Splits a string into a list on s.
- count(s): Counts the occurrences of string s in the string.
- find(s): Finds the index of the first occurrence of s.
- strip(): Strips leading and trailing whitespace in the string.

7 Tuple Type

- index(n): Returns the first index of n in the tuple.

8 Dictionary Type

Read more about the dictionary type in Python from their official documentation:
https://docs.python.org/2/library/stdtypes.html#typesmapping

- clear(): Clears the contents of the dictionary.

- keys(): Returns a list of all the keys in the dictionary.
- has_key(k): Returns true if the dictionary contains key k. False otherwise.
- pop(k): Removes the value at key k and returns it.
- values(): Returns a list of all the values in the dictionary.

9 Set Type

Read more about the set type in Python from their official documentation:
https://docs.python.org/2/library/sets.html

- add(o): Adds element o to the set.
- discard(o): Removes o from the set if o is present.
- intersection(s): Returns the intersection of the set and set s.
- union(s): Returns the union of the set and set s.
- difference(s): Returns the difference of the set and set s
- remove(o): Removes o from the set.