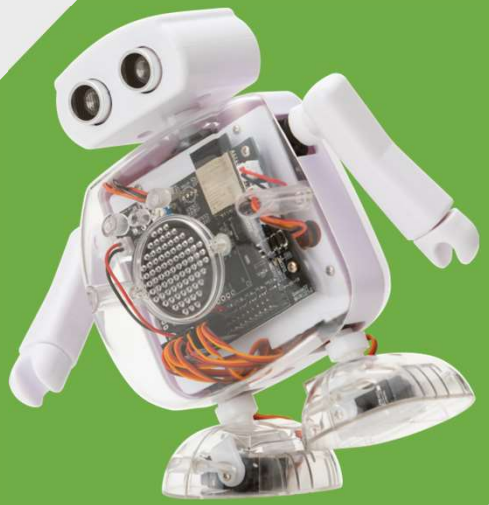


Pythonプログラミング

セッティング

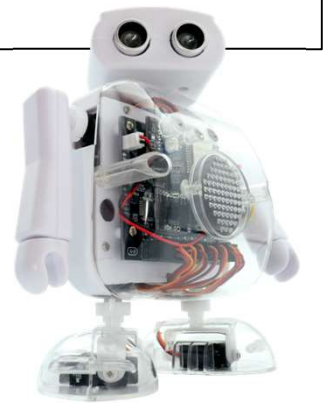
Pythonを使って

クムクムロボットを自由に動かそう！

CRETARIA.

始める前にチェックしましょう

- ✓ 1.パソコンはインターネットにつながっていますか？
- ✓ 2.パソコンにはBluetoothの機能がついていますか？
- ✓ 3.パソコンにはGoogleの「Chrome」がインストールされていますか？
- ✓ 4.ロボットには充電電池が使われていますか？
- ✓ 5.充電電池は十分に充電されていますか？



1.プログラミングの開始まで

1.プログラミング開始までの手順

1 ブラウザーを起動

1. **インターネットが接続されているパソコンで、Googleのブラウザ「Chrome」を起動してください。**
   ※ロボットを動かすために、Googleの**ChromeのWEB-BLE機能**を使用しているため**Chromeでしか動作しません。**
   ※FirefoxやEdgeやSafariなどはスクラッチが起動できても**ロボットとの通信ができません。**
2. **必ずインターネットに接続していることを確認してください。**
   ※1.fast.comなどに接続し通信速度を確認することでインターネット接続ができているか確認することができます。

2 専用Pythonエディターを起動

1. **クムクムが用意する専用サイトにChromeで接続しログインをしてスクラッチを起動してください。**
   ※クムクムは専用に開発されたPython専用エディタを使用します。
   ※標準のVisualCodeやインターネットサイトにあるPython学習用ツールなどでは動作しません。

3 ロボットの接続

1. **まずはロボットがスクラッチから動くかどうかテストしてください。**
   ※パソコンとクムクムロボットはBluetooth接続で動作するため、まずは通信がちゃんとできっているかどうかテストをします。

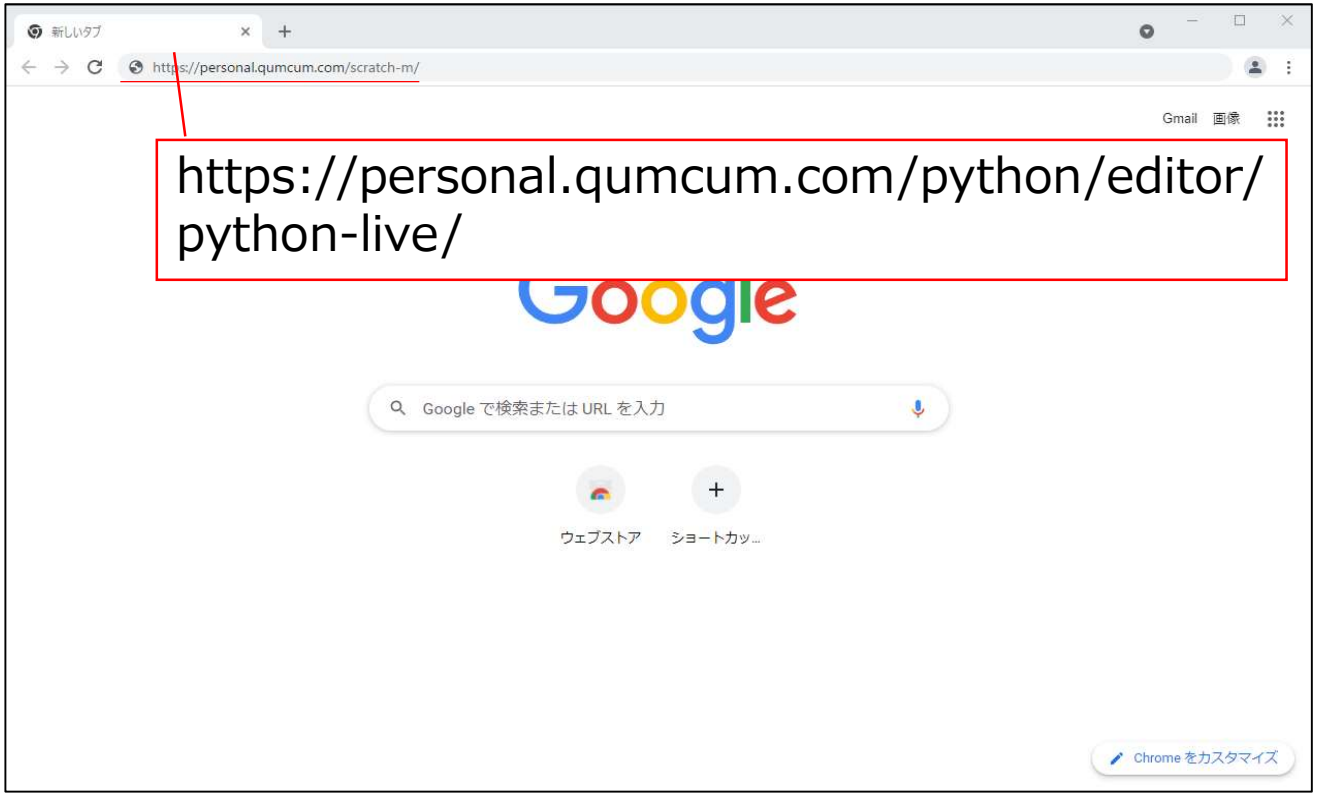
プログラミング開始

1 PythonEditorを起動

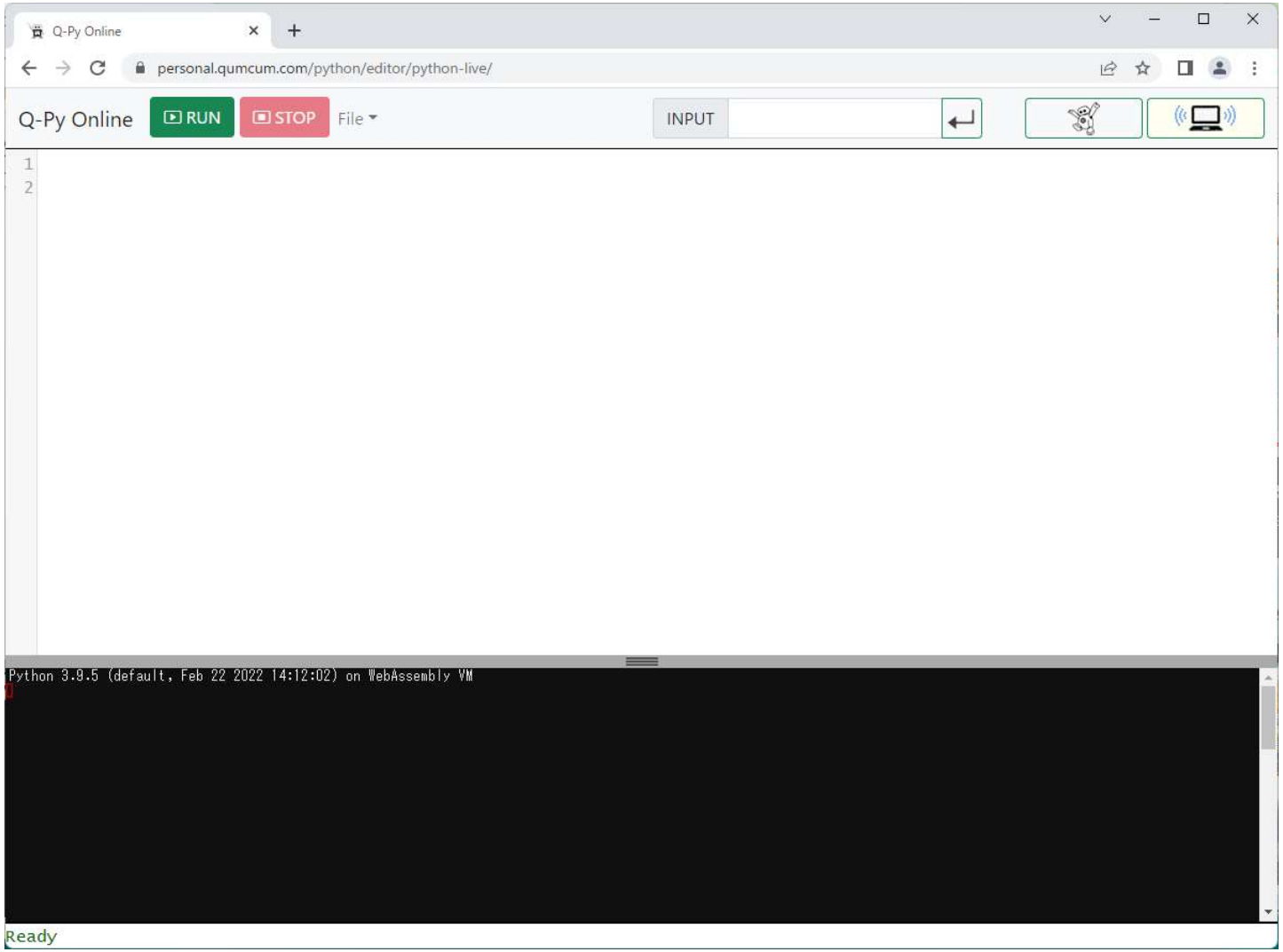
もしお使いのパソコンにChromeがインストールされていなかったら、下記のサイトよりダウンロードし、インストールをしてください。

サイトURL : https://www.google.co.jp/chrome/

1.Chromeを起動したらURL欄に下記のアドレスを入力します



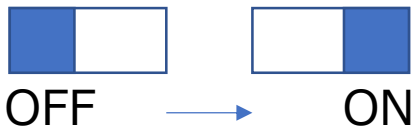
2.専用エディタが起動します



2 ロボットの接続

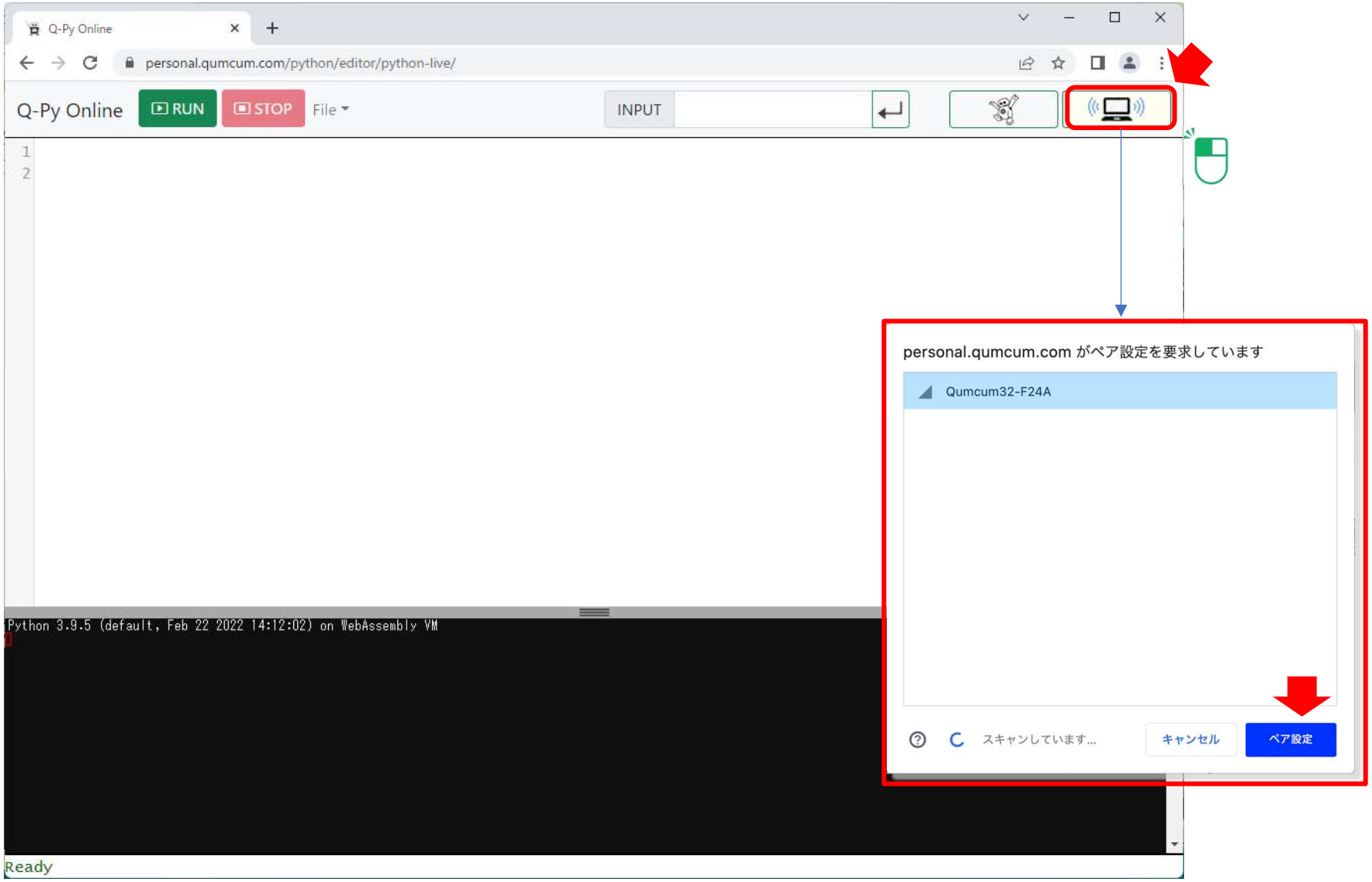
1.ロボット電源ON

クムクムロボット本体の背中面のスイッチをONにすると、クムクム正面のスピーカ上のRGB-LEDが変化し消え、その後ロボットからピっとブザー音が鳴り、ボードの右下の赤いLED光りロボット側の準備が完了します。



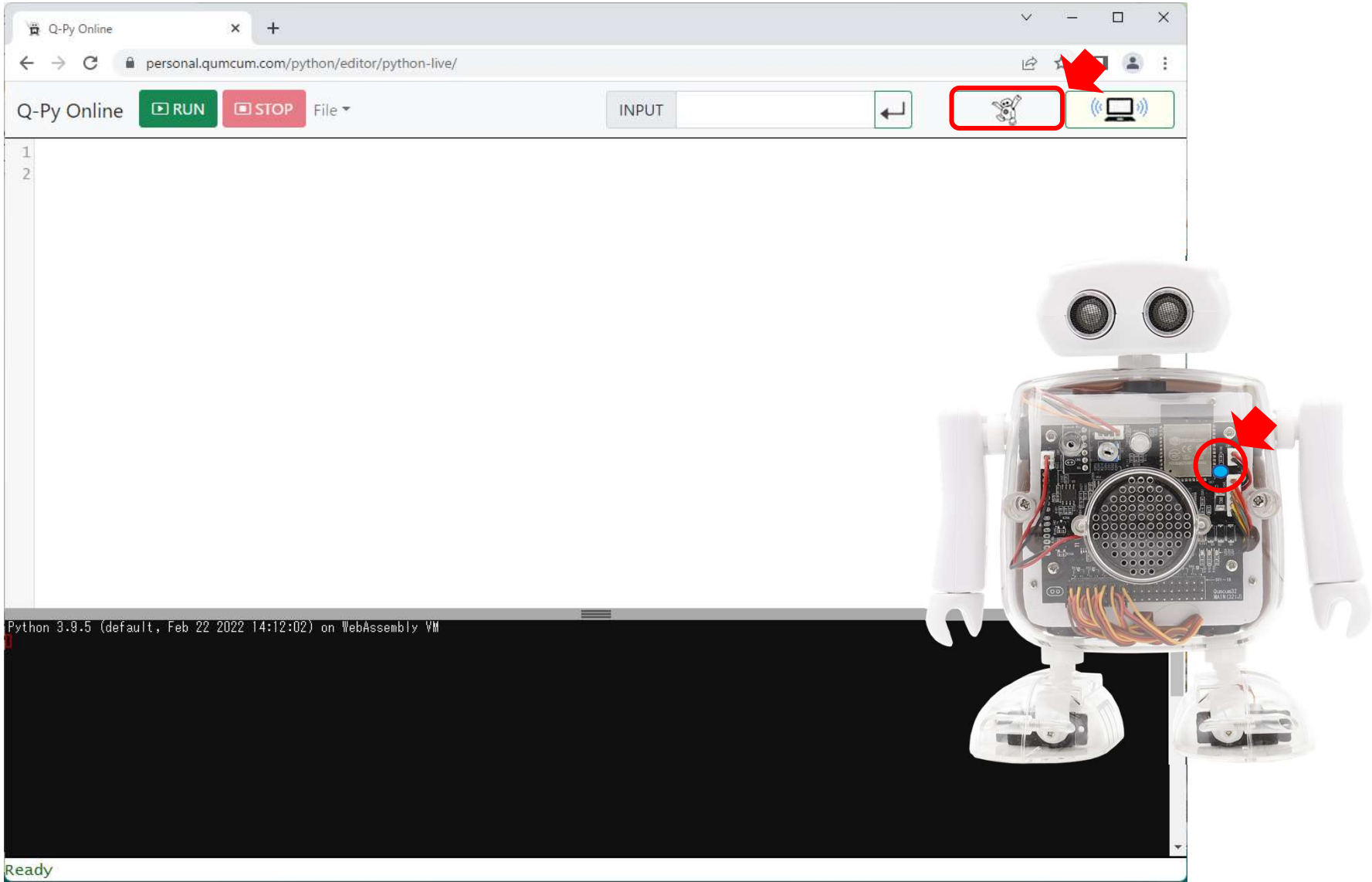
2.クムクムロボットと接続しよう

Pythonエディターの右上に表示されたパソコンの画面ボタンをクリックします。
接続ペアリングのダイアログが開くので、接続したいロボットを選択しペア設定のボタンをクリックします。

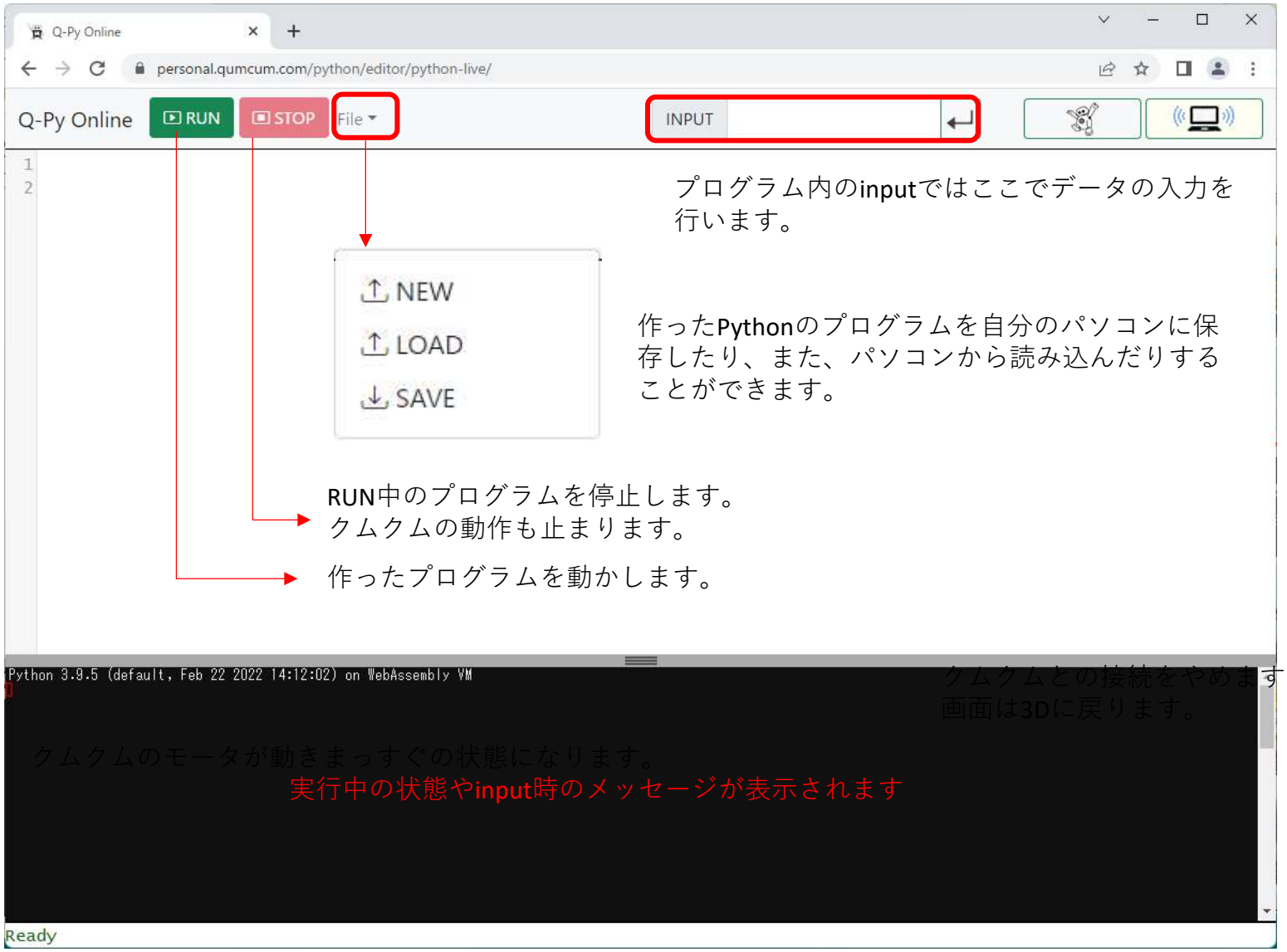


3.動作確認

接続ができるとロボット画面は「接続できたよ」としゃべり、下記のようなパネルに変わりクムクムの胸の青いLEDが点灯します。
画面右上クムクムボタン「ハロー」をクリックしクムクムロボットが動作するかを確認しましょう。
※ロボットは動き出しますので指などを挟まないように気を付けてください。



4.エディターの操作



プログラム内のinputではここでデータの入力を行います。

作ったPythonのプログラムを自分のパソコンに保存したり、また、パソコンから読み込んだりすることができます。

RUN中のプログラムを停止します。
クムクムの動作も止まります。

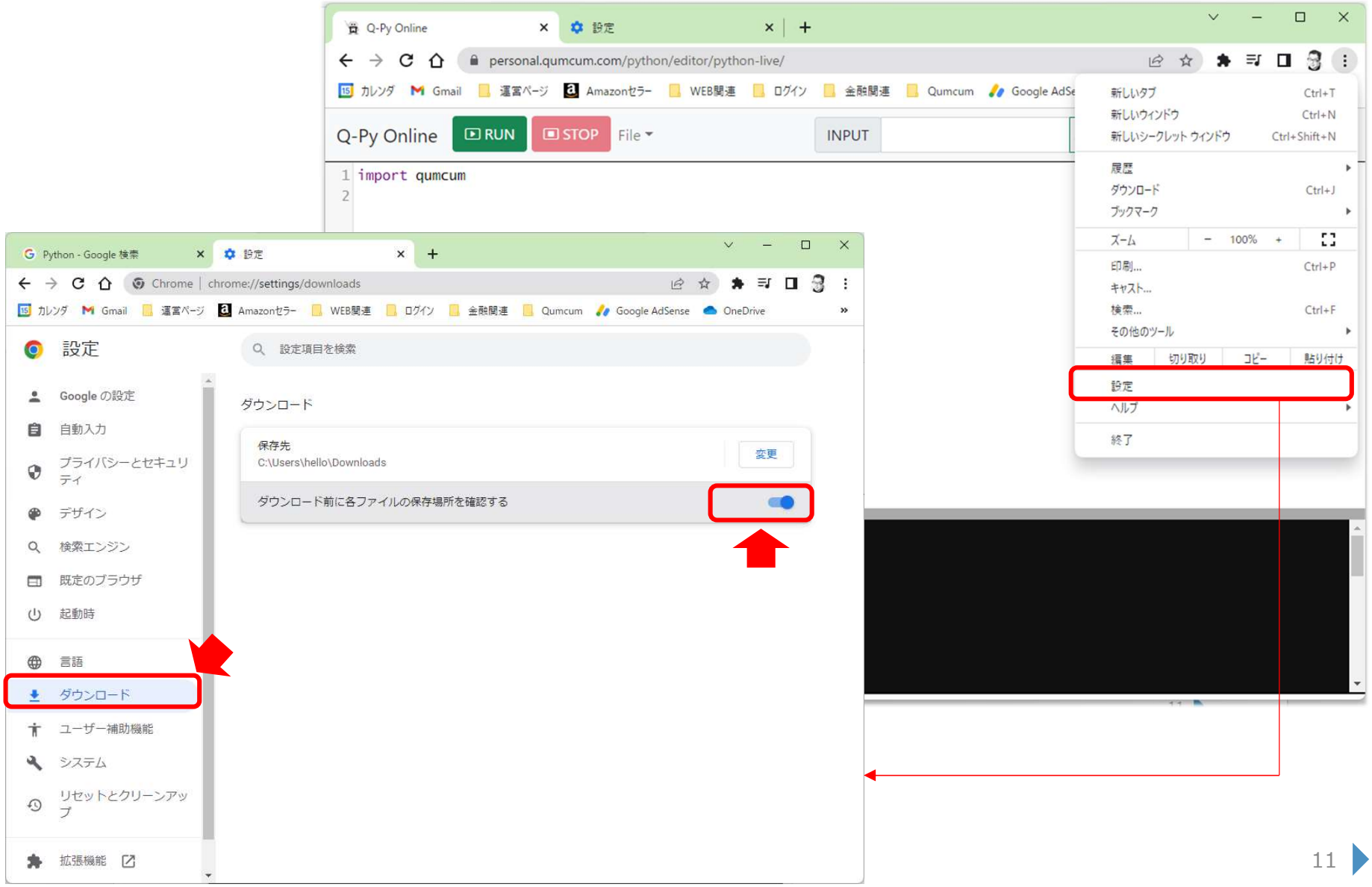
作ったプログラムを動かします。

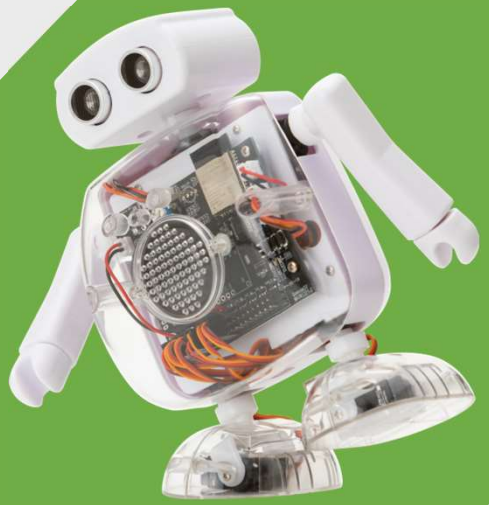
クムクムとの接続をやめます。
画面は3Dに戻ります。

クムクムのモータが動きまっすぐの状態になります。
実行中の状態やinput時のメッセージが表示されます

5.ファイル保存時に自分の好きな場所に好きなファイル名で保存したい場合

Chromeの場合「設定」-「ダウンロード」で下記のセッティングをします。





Pythonプログラミング

プログラミングリファレンス

Pythonを使って

クムクムロボットを自由に動かそう！

CRETARIA.

1 PythonEditorの動作

このエディタは、学習用ロボット「クムクム」を使ってPythonのプログラミング学習を行うためのWeb版エディタで、学習に必要なPython環境を一次的にローカルパソコンにダウンロードして動作します。

セキュリティー上インターネットを利用したプログラムを本エディタで作成することはできません。

このエディタで使用可能なライブラリー

このエディタでは、**クムクムを中心とした Python の基礎**を習得することを目的としています。
そのため、下記のライブラリーのみ使用できます。 下記以外の beatiful soup などインターネットに接続するライブラリなどを使用したプログラムを作成したい場合は、ご自分のローカルに構築した Python 環境でお使いください。

ライブラリ名	機能
sys	Pythonのインタプリタや実行環境に関する情報を扱うためのライブラリ
os	雑多なオペレーティングシステムインターフェース
platform	実行中プラットフォームの固有情報を参照する
time	時刻データへのアクセスと変換
qumcum	クムクムロボットを制御するためのライブラリ

※上記のライブラリー、また math や randomなどPython標準ライブラリーはimportして使用することができます
※本エディタにPIP等のコマンドで新しくライブラリをインストールすることはできません。
※ご自分のPCでのPython環境でクムクムを操作するライブラリは弊社ホームページよりダウンロードできます。

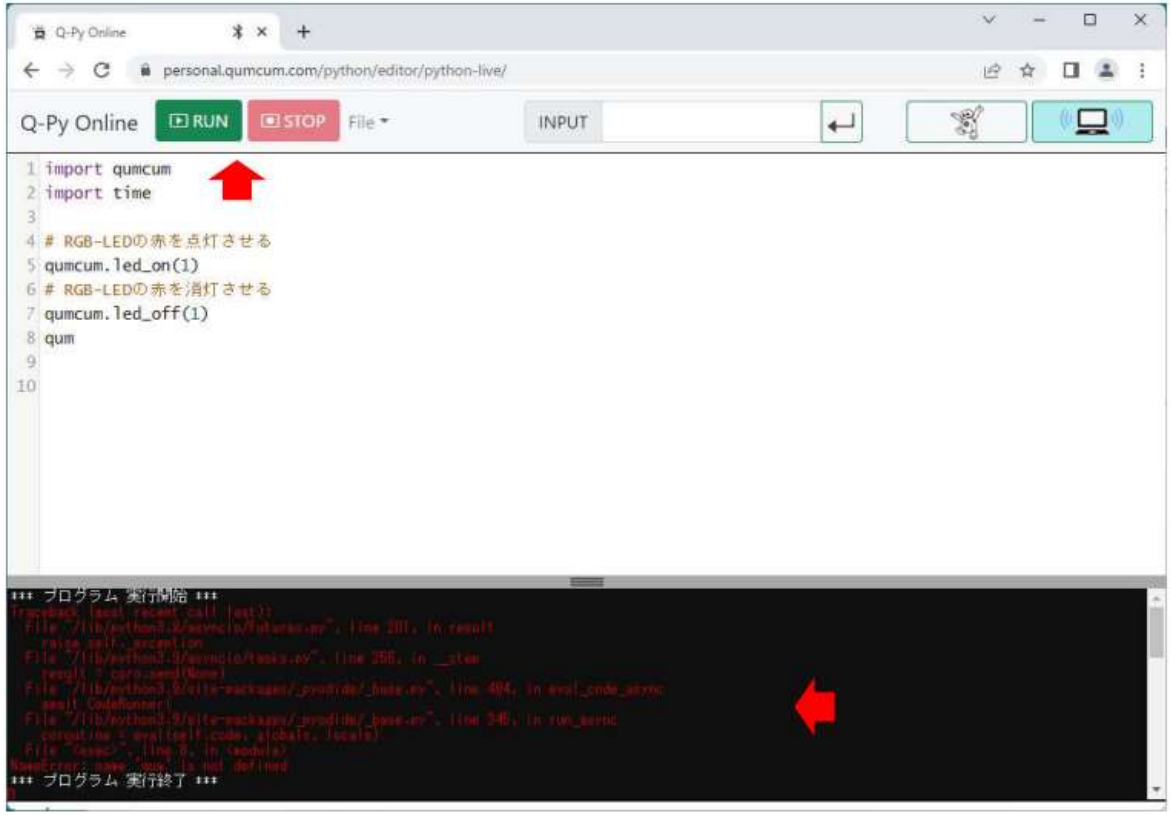
2 Pythonプログラミングと実行

プログラムを実行したい場合には[RUN]をクリックします。

Pythonエディターは記述したプログラムにエラーやミスがないかをチェックし、プログラムに問題がなければ画面下の黒いコンソール部分には、**** プログラム実行開始 ***と表示しプログラムを順次実行していきます。プログラムの動作が終わると ***プログラム 実行終了 *** と表示し動作は完了します。

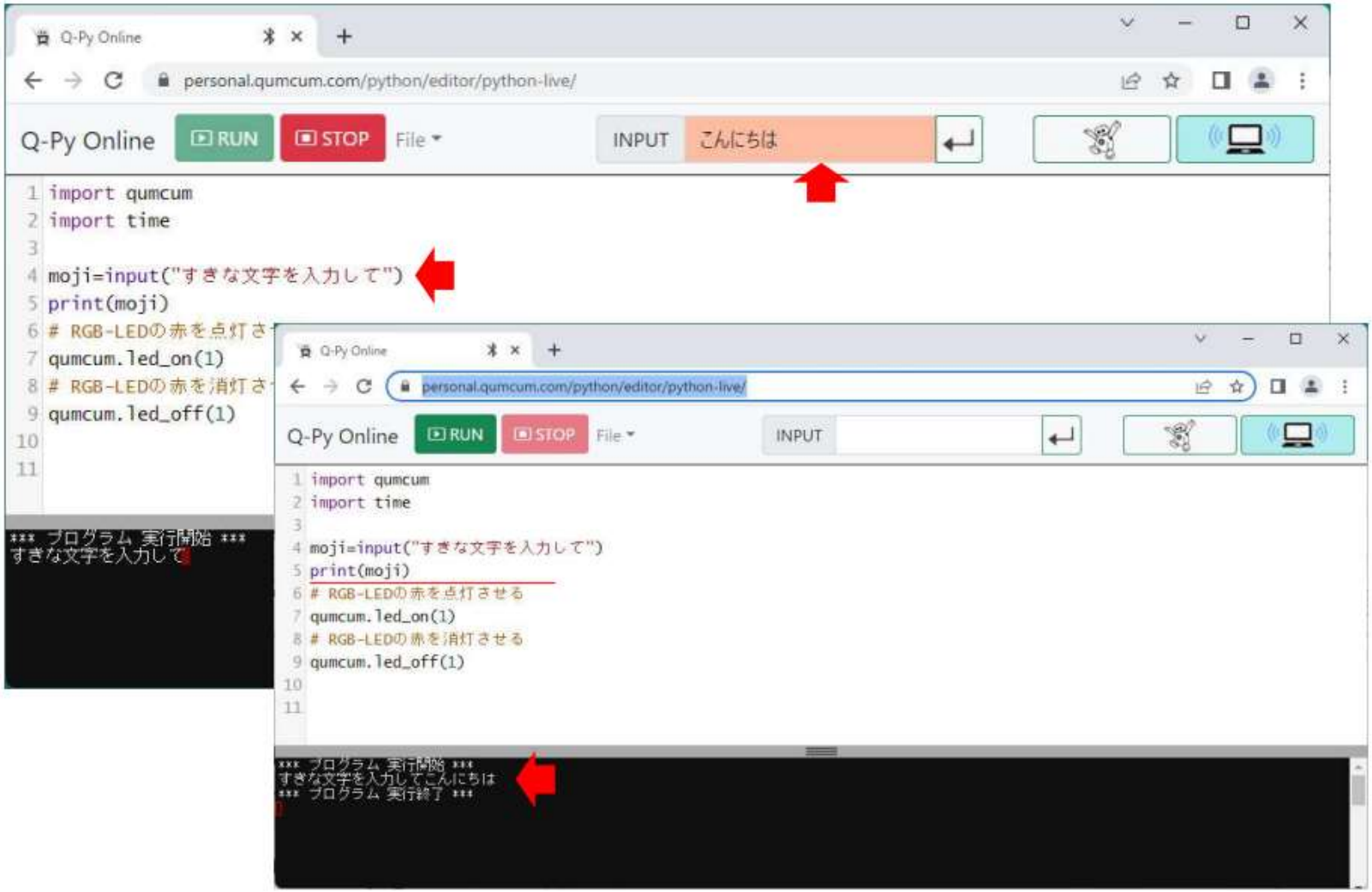
記述したプログラムにエラーやミスが見つかると画面下の黒いコンソール部分に赤い文字でエラーメッセージが表示されプログラムは実行されません。

動作中のプログラムを強制的に止めたい場合は[STOP]をクリックします。[STOP]をクリックすると画面下の黒いコンソール部分に赤い文字で強制ストップしたというメッセージが表示されます。



3 inputとprintが使えます

入力関数 input　をコールすると、画面下の黒いコンソール部分に input の引数で指定したメッセージが表示され、画面上の INPUT 部分で入力を待ちます。表示関数printを実行すると画面下の黒いコンソール部分printの引数で指定したデータなどを表示することができます。



WEB版・API版共通

コマンドリファレンス

パーツ	関数（コマンド）	機能	頁
LED	led_on	RGB-LEDの点灯	7
	led_off	RGB-LEDの消灯	7
BEEP	sound	ブザー音を鳴らす	8
VOICE	voice_speed	喋る速度を指定する	9
	voice	日本語を喋る	10
	voice_num	数字をそのまま喋る	11
	voice_numalpha	アルファベットを含む数字を喋る	11
	voice_math	日本の桁読みを付けて数字を喋る	11
	voice_fun	分を付けて数字を読む（いっぷん、にふん、さんぷん…）	12
	voice_pon	本を付けて数字を読む（いっぽん、にほん、さんぼん…）	12
MOTOR	motor_power_on	モーター電源を供給し動作完了までのデフォルト時間を決める	14
	motor_set_pos	指定した1つのモータ番号の回転角度を決める	15
	motor_pos_all	7個のモータすべての回転角度を同時に決める	16
	motor_angle_time	指定した1つのモータの回転角度と動作完了時間を個別に決める	17
	motor_angle_multi_time	7個のモータすべての回転角度と動作時間を同時に決める	18
	motor_start	モーターを回転させる（すべてのモータが目的角度になるまで次の処理に進みません）	19
	foot_up	左か右か、指定した足を上にあげ片足立ちの動作をする（歩くための簡易コマンド）	20
	foot_forward	左か右か、指定した足を前に出す （歩くための簡易コマンド）	21
	foot_down	足を下におろす（歩くための簡易コマンド）	22
SENSOR	get_sensor_value	距離センサーからのデータを取得する	23
MIC	get_mic_value	マイクからのデータを取得する	24
ROBOT	get_battery_value	バッテリーの残量を％で取得する	25
	get_lib_ver	ライブラリーのバージョンを取得する	26
	wait	処理を止めて指定時間まつ	26

関数	**led_on(color)**
機能	RGB-LEDの指定した色を点灯させます
引数	color・・・点灯させるLEDの色の番号（1＝赤　2＝青 3=緑　）
戻り値	なし
説明	指定した色のLEDがすでに点灯している場合は変化しません
使用例	qumcum.led_on(1) # 赤のLEDを点灯します qumcum.wait(1)　# <-- 赤のLEDが点灯してから1秒待ちます qumcum.led_off(1) # 赤のLEDを消灯します

関数	**led_off(color)**
機能	RGB-LEDの指定した色を消灯させます
引数	color・・・点灯させるLEDの色の番号（1＝赤　2＝青 3=緑　）
戻り値	なし
説明	指定した色のLEDがすでに消灯している場合は変化しません
使用例	qumcum.led_on(1) # 赤のLEDを点灯します qumcum.wait(1)　# <-- 赤のLEDが点灯してから1秒待ちます qumcum.led_off(1) # 赤のLEDを消灯します

関数	**sound(freq, time_ms)**
機能	指定した時間の間、指定した周波数の音を発生させます
引数	freq・・・発生させる周波数(Hz)（0～99999Hz） time_ms・・・音を出力する時間(msec)（0～9,999msec）
説明	使用しているスピーカー・アンプ機能により約30Hz～18,000Hzあたりまで再生が可能です。（ただし可聴範囲は個人差があります） 連続で sound で音を鳴らす場合、指定した時間鳴らし続けるには 1音ずつwaitもしくは time.sleepなどので鳴り終わりまで待つ処理が必要です。
使用例	qumcum.sound(880,1000) # 880Hzの音を1秒間出します qumcum.wait(1)　# <-- 1秒待ちます qumcum.sound(440,1000) # 440Hzの音を1秒間出します qumcum.wait(1)　# <-- 1秒待ちます qumcum.sound(440,0.5) # 220Hzの音を1秒間出します qumcum.wait(1)　# <-- 1秒待ちます

周波数と音階

ド	ド＃	レ	レ＃	ミ	ファ	ファ＃	ソ	ソ＃	ラ	ラ＃	シ
64	68	72	78	82	86	92	98	104	110	116	122
130	138	146	156	164	174	184	196	208	220	222	246
260	246	292	310	328	348	370	392	414	440	466	490
522	554	586	622	654	698	740	784	830	880	932	986
1044	1108	1174	1244	1318	1396	1480	1568	1660	1760	1864	1976
292	2216	2378	2490	2636	2794	2958	3136	3322	3520	3730	3950

関数	**voice_speed(speed)**
機能	voiceに用意されている各関数によって発話する速度を指定します
引数	speed・・・発話速度　（30[遅い]～300[速い]）
説明	デフォルト速度は100で指定されているため、特に速度を指定しない状態で喋らせることも可能です。
使用例	qumcum.voice_speed(30)　# 発話の速度を30(とてもゆっくり)に指定する qumcum.voice('konnitiwa') # 「こんにちわ」と発話します qumcum.wait(1)　　# 発話が終わるまで待ちます qumcum.voice_speed(300)　# 発話の速度を300(とてもはやく)に指定する qumcum.voice('konnitiwa') # 「こんにちわ」と発話します

アクセント記号

記号	動(き	記号	動き
.	長い間があく	;	下がって上がる
?	最後があがる	/	下がる
スペース	間が空く	+	あいまいな音
,	間が空く	‘	

関数	**voice(words)**
機能	ローマ字で指定した日本語を発話します。アクセント記号も使用できます。
引数	words・・・発話させたい言葉を“”でくくって指定します。（記号含めて40文字まで）
説明	※スペルミスがあった、使用できない記号などが含まれると何も発話しません。 喋らせる言葉は耳に聞こえるように指定します。（こんにちは konnnitiha → こんにちわ konnitiwa） 伸ばしたい場合は-を使います。（カレーライス kare-raisu　kare-------raisuとすると長く伸びる） **その他詳細はAquesTalkのマニュアルをご覧ください。** ※音声記号列仕様書(かな)：https://www.a-quest.com/archive/manual/siyo_onseikigou.pdf ※音声記号列仕様書(ローマ字)：https://www.a-quest.com/archive/manual/roman_onseikigou.pdf
使用例	qumcum.voice('konnitiwa') #「こんにちわ」と発話します qumcum.wait(1)　　# 発話が終わるまで待ちます qumcum.voice('<NUM VAL=123>') #「いちにさん」と発話します qumcum.wait(1)　　# 発話が終わるまで待ちます qumcum.voice('<NUMK VAL=123>en') #「ひゃくにじゅうさんえん」と発話します qumcum.wait(1)　　# 発話が終わるまで待ちます qumcum.voice('<ALPHA VAL=C001>') #「しーぜろぜろいち」と発話します qumcum.wait(1)　　# 発話が終わるまで待ちます

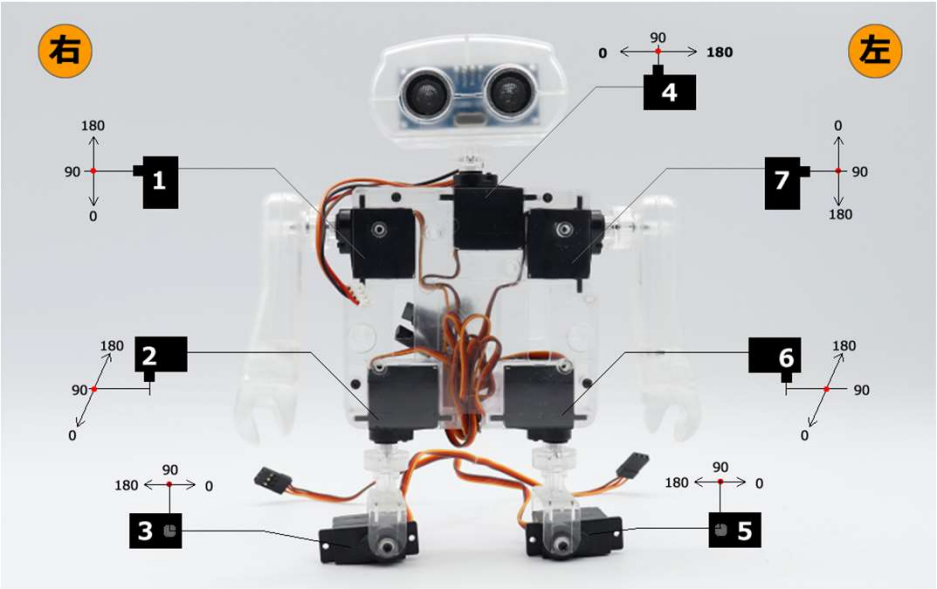
関数	voice_num(number)
機能	数字を日本語で1つずつそのまま読み上げます。1234 =「いち、に、さん、よん」
引数	number・・・40桁までの発話させる数字（“”でくくった文字列ではなく数値そのままを指定します）
説明	1つめに0がある場合（例えば電話番号 09011111111など）は発話できません。この場合は voice_numalpha 関数を使い文字列で数値を指定します。
使用例	qumcum.voice_num(123) # 「いちにさん」と発話します

関数	voice_numalpha(mum_str)
機能	英数字を1つづつそのまま読み上げます。（090-1111)「ぜろ,きゅう,ぜろ,ハイフン,いちいちいちいち」
引数	num_str・・・28文字までの英数字を表現する文字列
説明	！＃＄％＆/*-+<;:などの記号も発話します。
使用例	qumcum. voice_numalpha('A123') # 「えーいちにさん」と発話します

関数	voice_math(number)
機能	16桁（1000兆）までの数字を桁文字を付加して読みあげます
引数	number・・・桁読みする数字
説明	16桁以上を指定した場合、上位の桁は読まれません。
使用例	qumcum. voice_math(123) # 「ひゃくにじゅうさん」と発話します

関数	**voice_fun(number)**
機能	16桁（1000兆）までの指定した数字に分（ふん）をつけて読み上げます。 日本語特有の数字による分の読み方の違い（いっぷん、にふん、さんぷん）など、数字を指定するだけで読み上げます。
引数	number・・・16桁（1000兆）までの数値
説明	16桁以上を指定した場合、上位の桁は読まれません）
使用例	qumcum. voice_fun(123) #「ひゃくにじゅうさんぷん」と発話します

関数	**voice_pon(number)**
機能	16桁（1000兆）までの指定した数字に本（ほん）をつけて読み上げます。 日本語特有の数字による分の読み方の違い（いっぽん、にほん、さんぼん）など、数字を指定するだけで読み上げます
宣言	def voice_pon(number):
引数	number・・・16桁（1000兆）までの数値
説明	引数numberで指定した数値を本数としてクムクムの音声合成で発話します
使用例	qumcum. voice_pon (123) #「ひゃくにじゅうさんぼん」と発話します



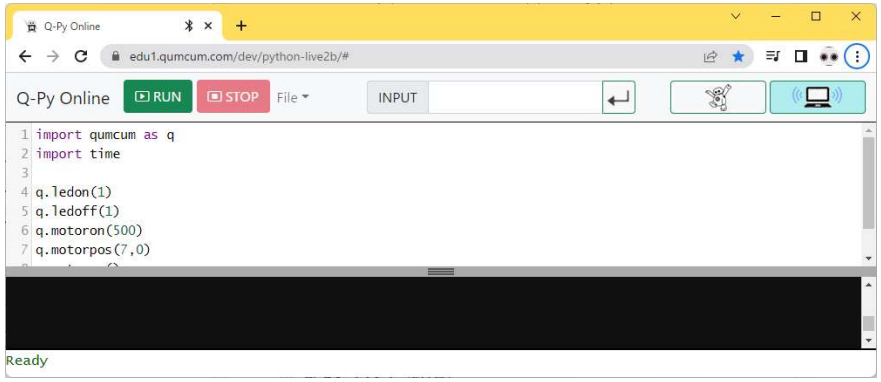
No	場所名	0度	90度	180度
1	右腕	下	前	上
2	右ふともも	内側に向く	まっすぐ	外側に開く
3	右足	内側が上がる	まっすぐ	外側が上がる
4	顔	右に向く	まっすぐ	左に向く
5	左足	外側が上がる	まっすぐ	内側が上がる
6	左ふともも	外側に向く	まっすぐ	内側に向く
7	左腕	上	前	下

モータプログラミングルール

RUNの前にはまっすぐにする

目的の動きができたかを知るには、まっすぐな姿勢から始める必要があります。そのためプログラムをRUNさせる前には下記の操作を行いロボットをまっすぐにしましょう。

エディターのどこかをクリックしエディターを選択する



CCtrl+Qでモーターを90度にする

関数	**motor_power_on(time_ms)**
機能	モーター電源をONにしモータ回転スタートから完了までにかかる時間を指定します。 動作に時間指定のない関数に有効です。 複数のモーターを同時に動かす場合、各モーターは回転する角度によって回転速度が内部で調整され、すべてのモータが同時に始まり同時に完了します。 ※動作時間を指定できる関数については、その関数で指定した動作時間が優先されます。
引数	time_ms・・・モータの動作時間(500msec～9999msec)
説明	クムクムのモーターを動かす場合、最初に本関数で必ず電源をONにしてください。 プログラム内で何度もコールする必要はありません。また電源をOFFにする関数はありません。
使用例	qumcum.motor_power_on(500) # 動作時間500msecでモータ電源をONにします qumcum.motor_set_pos(4, 0) # 4番(頭)のモータを0度(右)の位置へ動くよう指定します qumcum.motor_start() # モータを動かします qumcum.wait(1) # 動作が終わるまで待ちます qumcum.motor_power_off() # モータ電源をOFFにします

関数	**motor_set_pos(no, pos)**
機能	指定したモーターの回転角度を指定します。回転にかかる速度はmotor_power_onで指定した時間です。
引数	no・・・モータの番号（1～7）8～18までのモーターは本ツール・APIではコントロールはできません。 pos・・・モータの角度（モーター 1,4,7 ... 0～180度 モーター2,3,5,6....60～120度） 1:右手、2:右足、3:右足首、4:頭、5:左足首、6:左足、7:左手
説明	この関数をコールするだけではモーターは回転動作を行いません。 実際に回転させるタイミングで motor_start関数をコールしてください。 ※モーター2,3,5,6は安全のためロボット内プログラムで60～120度までの制限がかかっています。これ以上これ以下の角度を指定した場合最小値（60）最大値（120）に丸め込んで動作します。
使用例	qumcum.motor_power_on(500) # 動作時間500msecでモータ電源をONにします qumcum.motor_set_pos(4, 0) # 4番(頭)のモータを0度(右)の位置へ動くよう指定します qumcum.motor_start() # モータを動かします qumcum.wait(1) # 動作が終わるまで待ちます qumcum.motor_power_off() # モータ電源をOFFにします

関数	**motor_pos_all(pos1, pos2, pos3, pos4, pos5, pos6, pos7)**
機能	7個のモーター角度を同時に指定します。
引数	pos1～7・・・各モータの角度（モーター 1,4,7 ... 0～180度 モーター2,3,5,6....60～120度） pos1:右手、pos2:右足、pos3:右足首、pos4:頭、pos5:左足首、pos6:左足、pos7:左手
説明	この関数をコールするだけではモーターは回転動作を行いません。 実際に回転させるタイミングで motor_start関数をコールしてください。 ※モーター2,3,5,6は安全のためロボット内プログラムで60～120度までの制限がかかっています。これ以上これ以下の角度を指定した場合最小値（60）最大値（120）に丸め込んで動作します。
使用例	qumcum.motor_power_on(500) # 動作時間500msecでモータ電源をONにします qumcum.motor_pos_all(80, 80, 80, 80, 80, 80, 80) # 7つのモータを80度の位置へ動くよう指定します qumcum.motor_start() # モータを動かします qumcum.motor_pos_all(100, 100, 100, 100, 100, 100, 100) # 7つのモータを100度の位置へ動くよう指定します qumcum.motor_start() # モータを動かします qumcum.motor_pos_all(0, 90, 90, 90, 90, 90, 180) # まっすぐ起立したときの位置へ動くよう指定します qumcum.motor_start() # モータを動かします

関数	**motor_angle_time(no, pos, motor_time)**
機能	指定したモーターの回転角度と動作時間を指定します。 本関数で指定したモーターの速度は、motor_power_on 関数で指定した動作時間より優先されます。 ※あるモーターだけ早く動かしたい時などに使用します。
引数	no・・・モータの番号（1～7）8～18までのモーターは本ツール・APIではコントロールはできません。 pos・・・モータの角度（モーター 1,4,7 ... 0～180度 モーター2,3,5,6....60～120度） 1:右手、2:右足、3:右足首、4:頭、5:左足首、6:左足、7:左手 motor_time・・・動作時間（500msec～9999msec）
説明	この関数をコールするだけではモーターは回転動作を行いません。 実際に回転させるタイミングで motor_start関数をコールしてください。 ※モーター2,3,5,6は安全のためロボット内プログラムで60～120度までの制限がかかっています。これ以上これ以下の角度を指定した場合最小値（60）最大値（120）に丸め込んで動作します。 ※注意：この関数をコールすると、指定したモーターのデフォルト速度が書き換わり、motor_power_onで指定した速度がこのモータにだけ適用されなくなります。もう一度すべてのモータ速度を一括で指定したい場合は　motor_power_on で再度速度指定を行います。
使用例	qumcum.motor_power_on(500) # 動作時間500msecでモーター電源をONにします qumcum.motor_pos_all(100, 100, 100, 100, 100, 100, 100) #全モーターを100度の位置へ（500msecで動きます） qumcum.motor_angle_time(4, 0, 2000)　# 4番(頭)のモーターだけを2000msecで0度(右)の位置へ動かします qumcum.motor_start()　　# モータを動かします

関数	**motor_angle_multi_time(pos1, pos2, pos3, pos4, pos5, pos6, pos7, motor_time)**
機能	本関数で指定したモーターの速度は、motor_power_on 関数で指定した動作時間より優先されます。 ※あるタイミングだけ早く動かしたい時などに使用します。
引数	pos・・・モータの角度（モーター 1,4,7 ... 0～180度 モーター2,3,5,6....60～120度） 1:右手、2:右足、3:右足首、4:頭、5:左足首、6:左足、7:左手 motor_time・・・動作時間（500msec～9999msec）
説明	この関数をコールするだけではモーターは回転動作を行いません。 実際に回転させるタイミングで motor_start関数をコールしてください。 ※モーター2,3,5,6は安全のためロボット内プログラムで60～120度までの制限がかかっています。これ以上これ以下の角度を指定した場合最小値（60）最大値（120）に丸め込んで動作します。 ※注意：この関数をコールすると、全モーターのデフォルト速度が書き換わり、motor_power_onで指定した速度が適用されなくなります。もう一度すべてのモータ速度を一括で指定したい場合は　motor_power_on で再度速度指定を行います。
使用例	qumcum.motor_power_on(500) # 動作時間500msecでモータ電源をONにします qumcum. motor_angle_multi_time(80, 80, 80, 80, 80, 80, 80,1200) # 7つのモータを80度の位置へ1.2秒で動くようにします qumcum.motor_start()　　# モータを動かします qumcum. motor_angle_multi_time(0, 90, 90, 90, 90, 90, 180, 1000) # まっすぐ起立したときの位置へ1.0秒で動くようにします qumcum.motor_start()　　# モータを動かします

関数	**motor_start(no_wait=False)**
機能	モーターを回転させます。
引数	no_wait・・・(True)... モーター回転終了を待たずに次のプログラム行を実行します。 (Falseもしくは引数なし）...モーター回転終了まで次のプログラム行は実行されません。
説明	no_waitをTrueにした場合、連続でモーター回転動作プログラムを組むと思った通りの動きになりません。必ずmotor_start関数の後に waitやsleep関数などで回転完了までのタイミング取ります。ただし、プログラムでタイミングを作ることができるため、wait=Falseで動作させるより自由にモータをコントロールすることができ高速歩行なども可能になります。
使用例	qumcum.motor_power_on(500) # 動作時間500msecでモータ電源をONにします qumcum.motor_set_pos(4, 0) # 4番のモータを0度の位置へ動くよう指定します qumcum.motor_start() # モータを動かします qumcum.motor_set_pos(4, 90) # 4番のモータを90度の位置へ動くよう指定します qumcum.motor_start() # モータを動かします qumcum.motor_power_off() # モータ電源をOFFにします

関数	**foot_up(side)**
機能	指定した足を上にあげもう一方の足で片足立ちをする動作を行います。
引数	side・・・上げる足(1:右足、2:左足)
説明	歩かせたりする際の片足立ちの動きは、4個のモータの角度と速度を微妙に調整する必要があるため非常に難しいため、簡単に歩かせてみたい場合に利用できるサービス関数です。 足の4個のモータをAPI内部でコントロールし確実に片足で立つ動作を提供します。 動作速度の時間は motor_power_on で指定した速度とは異なり独自の速度で動作します。 実際に動かすタイミングで motor_start関数をコールしてください。 motor_start前に本関数と腕や顔のモータの角度指定を行っておくことで同時にアクションをさせることができます。
使用例	qumcum.motor_power_on(500) # 動作時間500msecでモータ電源をONにします qumcum.foot_up(1) # 右足を上に上げる qumcum.motor_set_pos(4, 0) # 4番(頭)のモータを0度(右)の位置へ指定します qumcum.motor_start() # モータを動かします qumcum.motor_power_off() # モータ電源をOFFにします

関数	**foot_forward(side)**
機能	指定した足を前に出動作をします。
引数	side・・・前に出す足(1:右足、2:左足)
戻り値	なし(0のみ)
説明	足の4個のモータの角度と速度をAPI内部でコントロールし足をまっすぐ前に出す動作を提供します。 動作速度の時間は motor_power_on で指定した速度とは異なり独自の速度で動作します。 実際に動かすタイミングで motor_start関数をコールしてください。 motor_start前に本関数と腕や顔のモータの角度指定を行っておくことで同時にアクションをさせることができます。
使用例	qumcum.motor_power_on(500) # 動作時間500msecでモータ電源をONにします qumcum.foot_up(1) # 右足を上に上げる qumcum.motor_start() # モータを動かします qumcum.foot_forward(1) # 右足を前に出す qumcum.motor_start() # モータを動かします qumcum.motor_power_off() # モータ電源をOFFにします

関数	**foot_down()**
機能	foot_up関数で上げた足を下ろす動作をします。
引数	なし
説明	両足角度を90°に戻すことで上げた足を下す動作となります。 動作速度の時間は motor_power_on で指定した速度とは異なり独自の速度で動作します。 実際に動かすタイミングで motor_start関数をコールしてください。 motor_start前に本関数と腕や顔のモータの角度指定を行っておくことで同時にアクションをさせることができます。
使用例	qumcum.motor_power_on(500) # 動作時間500msecでモータ電源をONにします qumcum.foot_up(1) # 右足を上に上げる qumcum.motor_start() # モータを動かします qumcum.foot_forward(1) # 右足を前に出す qumcum.motor_start() # モータを動かします qumcum.foot_down() # 足をおろす qumcum.motor_start() # モータを動かします qumcum.motor_power_off() # モータ電源をOFFにします

	val=get_sensor_value( )
機能	超音波センサから目前にある障害物までの距離を取得します。
引数	なし
戻り値	val…計測した距離数値 （2～29cmまでの計測を行います。計測物がないか、30cm以上離れているときは999を戻します）
説明	周囲の温度等によって距離は変化します。 超音波が反射できる障害物を用意します。 ロボットの電源が入ったままの状態では、前回計測した値がボード上に残っている可能性があります。 その場合は、999が返るまでから読みをする処理を追加します。 計測は瞬時のため、繰り返しなどポーリング等で計測する処理が必要です。

使用例	#前回の計測値がなくなるまでのから読み
	sens_val = get_sensor_value()
	while sens_val !=999 :
	　sens_val = get_sensor_value()
	#なにか計測するまでループ
	while sens_val == 999 :
	　sens_val = get_sensor_value()
	print(sens_val)　　# 取得した超音波センサの計測値を出力します

関数	**val=get_mic_value()**
機能	マイクで音の大きさを取得します。
引数	なし
戻り値	取得した音の大きさを0~100までの数値で返します。
説明	クムクムで計測したマイクの計測値を取得します。 計測は瞬時のため、繰り返しなどポーリング等で計測する処理が必要です。
使用例	mic_val = get_mic_value() # マイクの計測を行い、計測値を取得します print(mic_val) # 取得したマイクの計測値を出力します

関数	**val=get_battery_value( )**
機能	ロボットのバッテリ残量のレベルを取得します。
引数	なし
戻り値	val=0（2.8V）～100（5.0V）
説明	ニッケル水素充電電池は購入時の満充電で1本約1.36Vです。 クムクムでは4本使用するため、1.36×4=約5.44Vが新品時の満充電電圧となります。 クムクムでは、電池と使用上の誤差を考慮し5.0V以上を100として値を返します。 モーターは2.8Vになると動作が不安定となるためクムクムでは2.8Vを0としています。
使用例	battery_val = get_battery_value() # バッテリ残量の計測を行い、計測した目安の値を取得します print(battery_val) # 取得したバッテリ残量の目安の値を出力します

関数	**lib=get_lib_ver()**
機能	現在使用しているクムクム用Pythonライブラリのバージョンを取得
引数	なし
戻り値	ライブラリバージョンの文字列(“0.1.0”など)
説明	当ライブラリのバージョンを取得します
使用例	lib_ver = qumcum.get_lib_ver()　# lib_verにバージョン番号の文字列が格納されます print(lib_ver)　　# ライブラリバージョンを表示します

関数	**wait(time_sec)**
機能	指定した時間プログラムの動きを止めます。（Pythonの time.sleep 関数と同じ）
引数	time_sec・・・停止時間(単位：秒)
戻り値	なし
説明	関数内部ではPythonの time.sleepをコールしています。 プログラミング上 qumcum で表記を統一したい場合に使用すると便利です。
使用例	qumcum.led_on(1) # 赤のLEDを点灯します qumcum.wait(1)　# <-- 赤のLEDが点灯してから1秒待ちます qumcum.led_off(1) # 赤のLEDを消灯します

Qumcum

qumcum.com

CRETARIA.