クムクム Python エディタ

ユーザーマニュアル

1.3

目次

始めに

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。

このエディタで使用可能なライブラリ

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しか使用することができません。
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 環境でクムクムを捜査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。

エディタの起動

1.ブラウザ(Google Chrome、もしくは Microsoft Edge Chrominum)で下記の URL のページに接続し起動します。

https://personal.qumcum.com/python/editor/python-live/

2.接続すると下記のスプラッシュ画面が行事されます。

3.しばらくするとエディタ画面が表示されます。

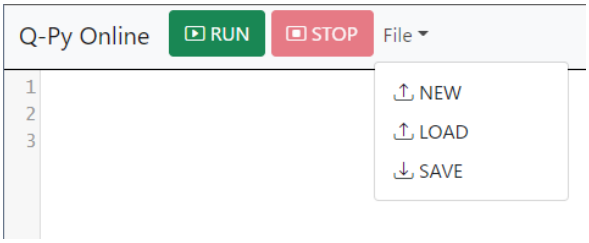
エディタ画面について

エディタの画面は、機能ごとに下記の通り①～③の構成になっています。

①操作部

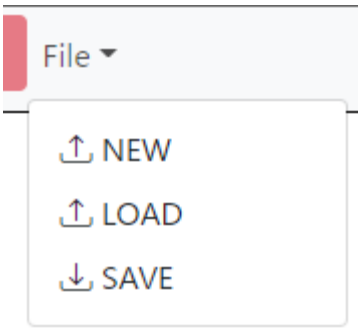
ロボットやプログラムに関する操作を行います。

(A)左部の機能



名称	機能
[RUN]ボタン	プログラムを実行します。ロボットへの命令はロボットに随時送信されます。
[STOP]ボタン	実行中のプログラムを中断します。ロボットの動きも停止します。
[File]	作成したプログラムのファイルへの保存やファイルからの読み出しなどを行います。

[File]メニュー



名称	機能
[NEW]	エディタ部のコードをすべて消して、新たにプログラムが作ることができるようにします
[LOAD]	PC にあるクムクム用の python プログラムファイル(*.py)を読み込みエディタ部に表示します
[SAVE]	作成したプログラムを PC 内に python プログラムファイル(*.py)として保存します

[SAVE]を行うときに、ファイル名をつける画面が表示されず’save.py’というファイル名で保存されることがあります。

そのときは、次ページの「[SAVE](プログラムの保存)について」を見て、ブラウザ設定を変更してから[SAVE]を行うようにしてください。

[SAVE](プログラムの保存)について

[SAVE]でプログラムを保存するときに、ブラウザの設定で既定のファイル名(save.py)でそのまま保存されることがあります。

その場合は、ブラウザの設定を変更することで保存するファイルに名前を付けてから保存ができるようになります。

【Google Chrome の場合】

① Chrome のウィンドウの右上角付近にある”点が縦に三つ並んだボタン“からメニューを開き、「設定」を開きます。

② 左側ペインの「詳細設定」をクリックし、「ダウンロード」をクリックします。

③ 右ペインに表示された「ダウンロード前に各ファイルの保存場所を確認する」を ON にします。

[SAVE]ボタンでファイルを保存するときにファイル名を入力してから保存することができるようになります。

【Microsoft Edge の場合】

① Microsoft Edge の右上角付近にある”点が横に三つ並んだボタン”からメニューを開き、「設定」を開きます。

② 左側ペインの「ダウンロード」をクリックし、「ダウンロード時の動作を毎回確認する」を ON にします

[SAVE]ボタンでファイルを保存するときにファイル名を入力してから保存することができるようになります

[SAVE]ボタンでファイルを保存するときに「ダウンロード」のダイアログが表示されます。

「ダウンロード」のダイアログで[保存]ボタンをクリックしたときに「名前を付けて...」をクリックするとファイル名を入力してから保存することができるようになります。

(B)操作部(右側)

名称	機能
[インプット]ボックス	INPUT 命令を実行するときの文字列を入力するためのボックスです。
[ハロー]ボタン	クムクムの動作チェックを行います。
[接続/切断]ボタン	本エディタとクムクムを Bluetooth で接続します。接続用のダイアログが表示されます。

②エディタ部

プログラムを記述する部分です。

③結果表示部

プログラムを実行時の実行結果が表示されます。プログラムのエラー発生時のメッセージなども表示されます

プログラミング方法

1.ロボットとの接続

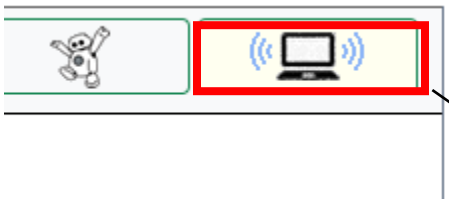
1.ロボットの電源を入れます

ロボットの電源スイッチ

右にスライド(→)すると電源が入り、左にスライド(←)すると電源が切れます

2.ロボットとエディタを Bluetooth で接続します

1.接続ボタンをクリックします



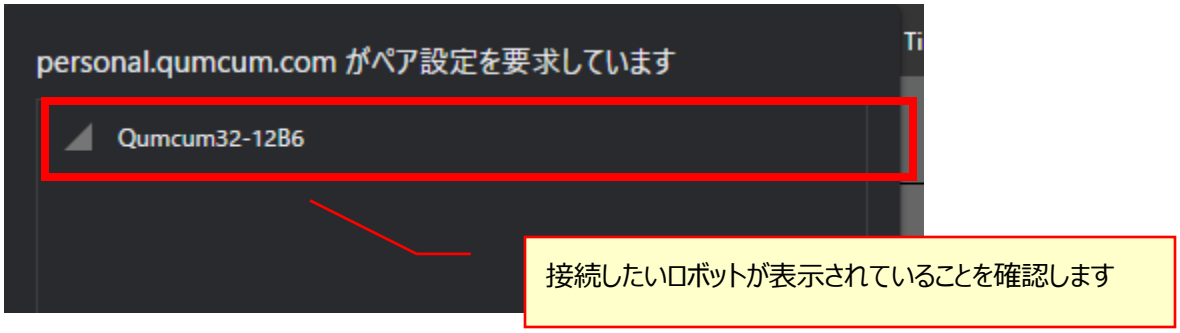
[接続/切断]ボタン

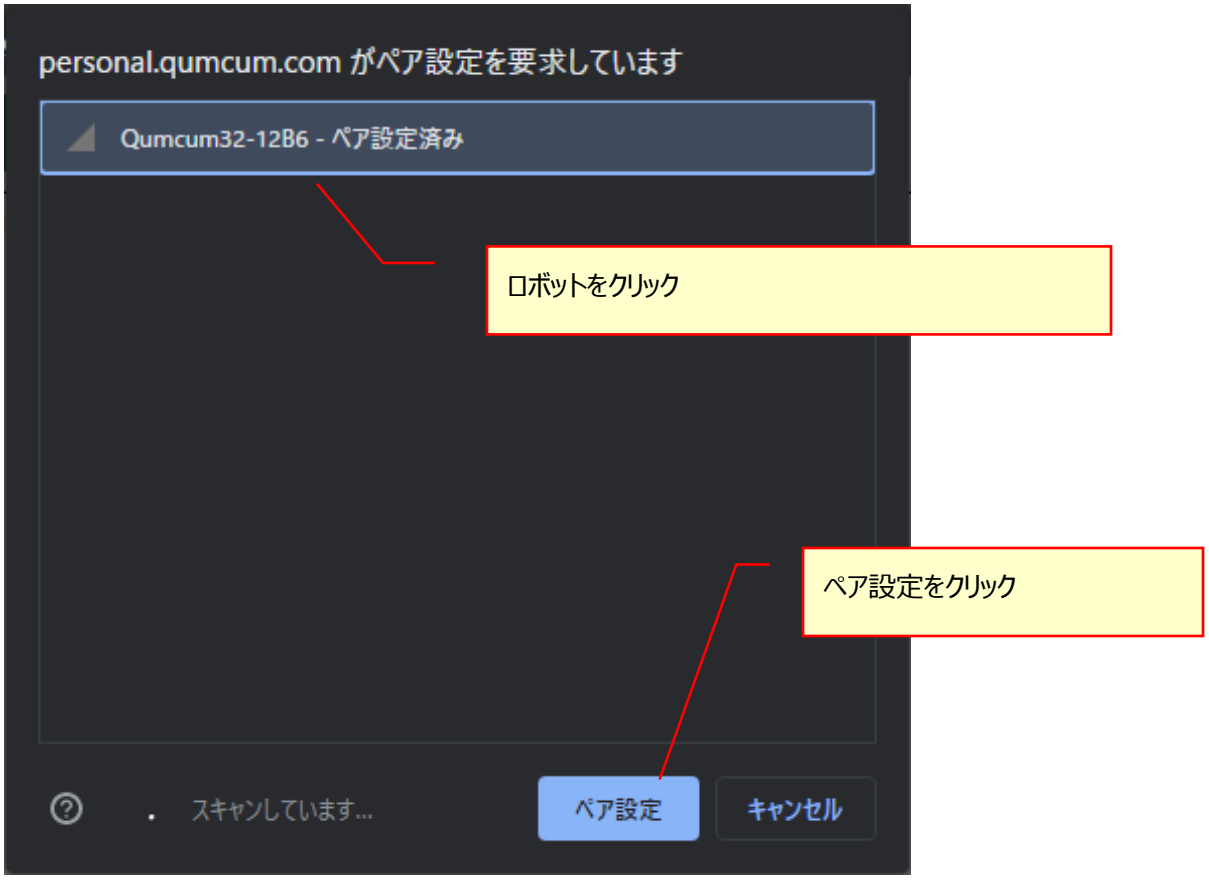
2.接続先のロボットを決定します

ロボット一覧ダイアログボックスが表示され、PC 周辺に存在するロボットのリストが表示されます。

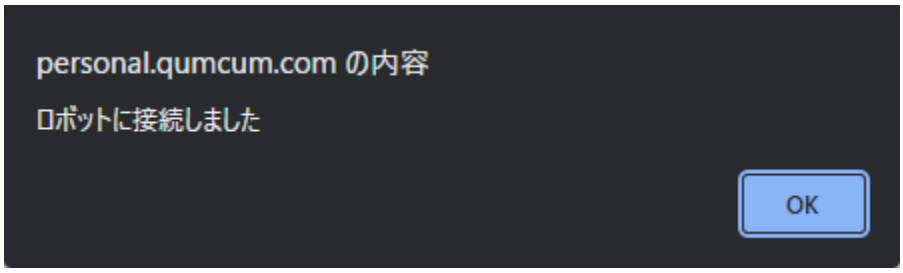
ご購入いただいたときに同封された用紙に記述されている、ロボットの ID の下 4 桁と一致っするロボットを探し出し選択します。

※パソコンに Bluetooth ない場合や、ロボットの電源が入っていないとき一覧に表示されません。

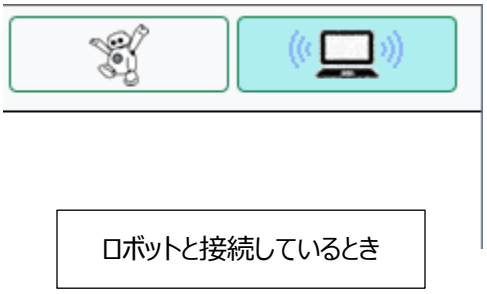




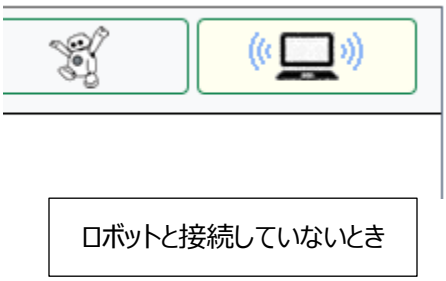
正しく接続できると下記のメッセージダイアログが表示されますので OK ボタンをクリックし接続操作を完了します。



※備考：接続ボタンの表示により接続中かどうかが判断することができます。



ロボットと接続しているとき



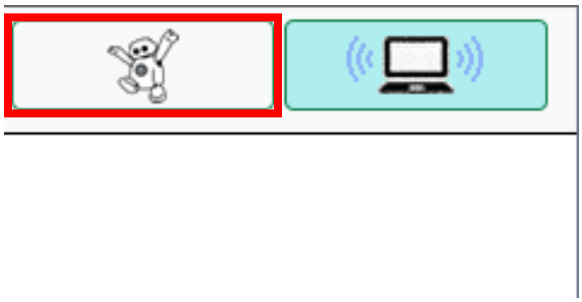
ロボットと接続していないとき

2 ロボットの機能確認(ファンクションチェック)

ロボットの機能が正しく動作するかを確認するために、まずロボットのファンクションチェックを行います。

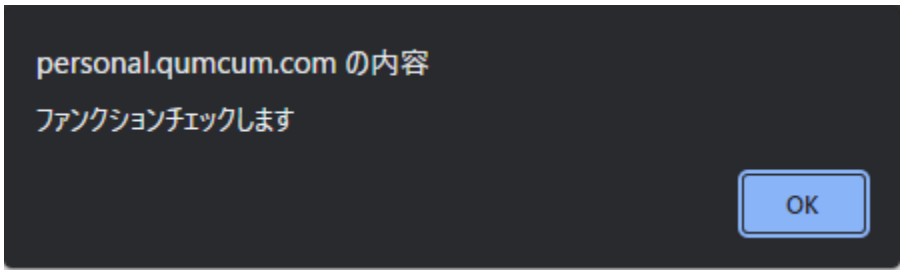
ファンクションチェックを行うことにより、パソコンとロボットが通信できているか、ロボットの各パーツが正しく動作するかを確認できます。正しく動作しない場合、電池残量や Bluetooth 接続などを再度チェックしてください。

1.ハローボタンをクリック



2.開始の確認の[OK]をクリックしロボットを動作

ロボットが動き始めるのでご注意ください。



3.下記のように動作するかを目視確認

順番	チェック対象	動作
1	RGB-LED	赤点灯→赤消灯→緑点灯→緑消灯→青点灯→青消灯
2	BEEP	440Hz→880Hz の音が連続で 2 回鳴ります
3	VOICE	「はろーくむくむ」としゃべります
4	モータ(頭)	90 度[正面]→0 度→180 度→90 度[正面]
5	モータ(両手)	右手を上げる[0 度→180 度]→左手を上げる[180 度→0 度]→両手を下げる[右手 0 度、左手 180 度へ]
6	モータ(両足)	左足を上げる→右足を上げる→両足を下げる

プログラミング

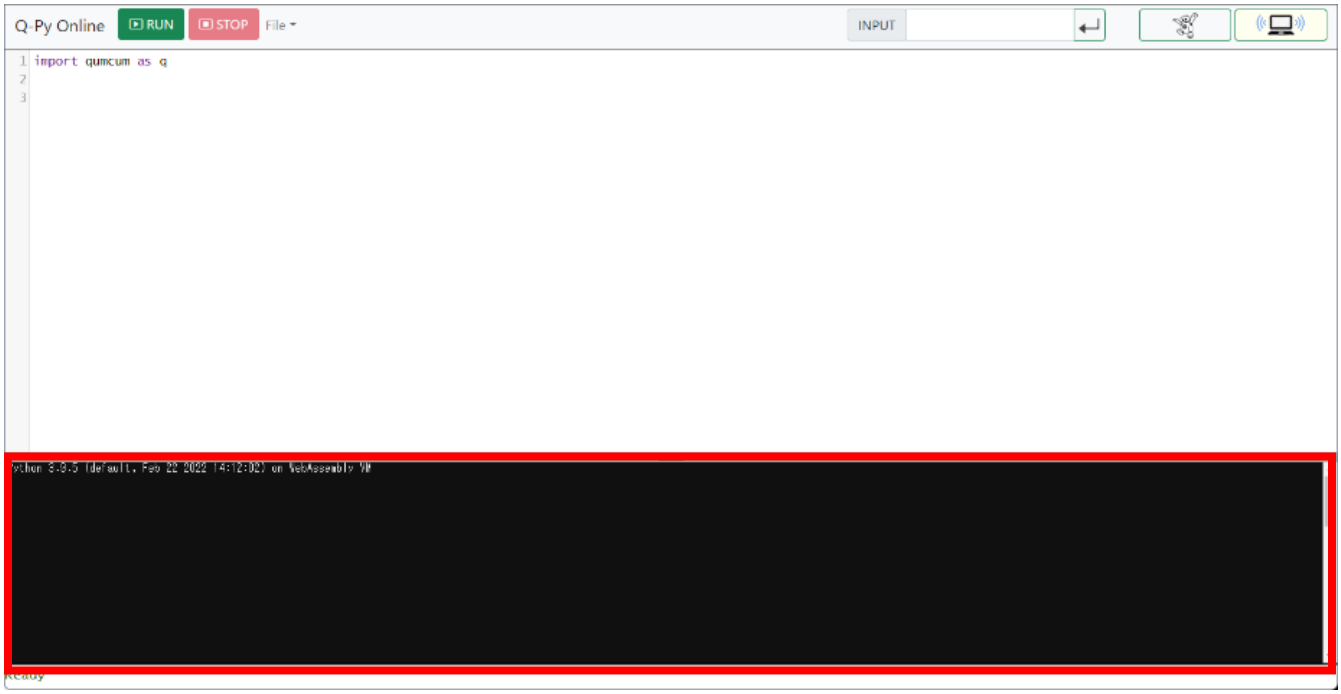
エディタ部に Python のプログラムを記述し実際にロボットを動かします。クムクムを動かすための命令などについては、「API 仕様」を参照してください。

1.プログラムを動かす

[RUN]ボタンをクリックすると Python プログラムは順次実行され、ロボットに命令を送り出し作ったプログラムの動きを確認することができます。

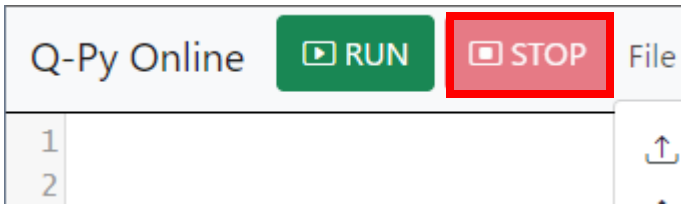
2.実行結果を確認する

プログラムの実行結果や PRINT 分などのメッセージは結果表示部にメッセージされます。



3.実行中のプログラムを停止させる

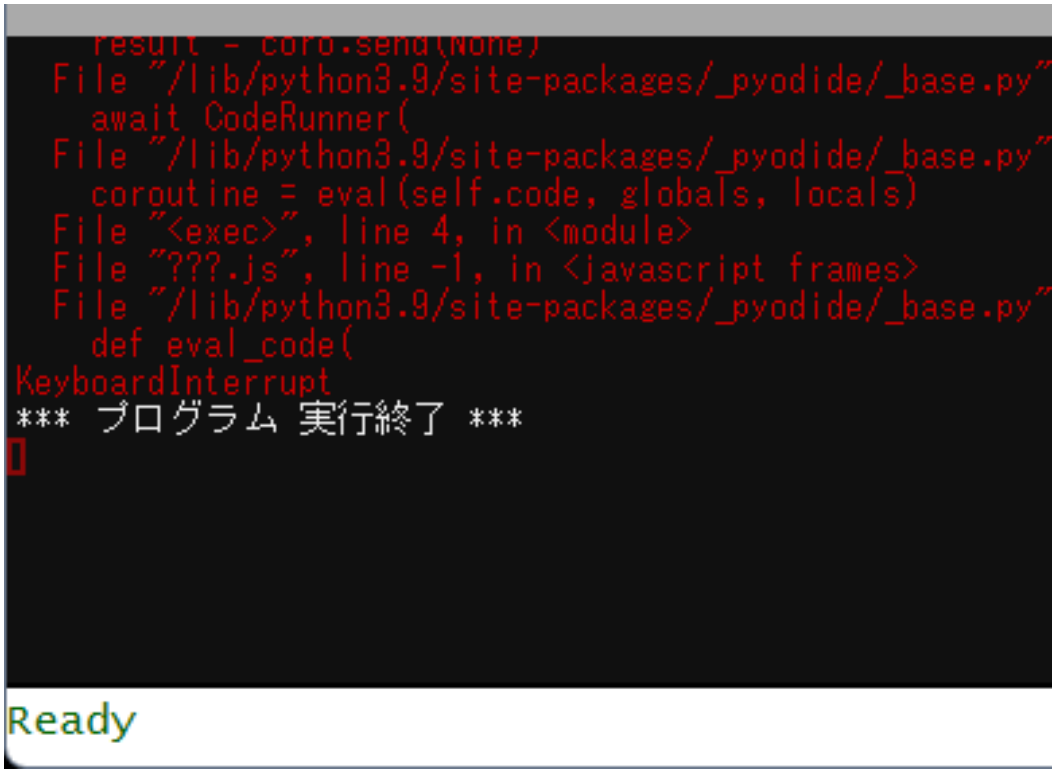
実行したプログラムが、何かのエラーや無限ループなどで終了しない場合などに、[STOP]ボタンをクリックしてプログラムを強制的に止めることができます。本ボタンをクリックした場合、ロボットへの命令転送も中止されるため、ロボットの動きも途中で止まります。そのため思わぬ状態でロボットが停止しバランスを崩して倒れることもあります。ロボットの破損にもつながりますので、十分にご注意ください。



※備考

[STOP]をクリックすると、実行中のプログラムは即座に停止され、結果表示部に下記のような赤字のメッセージが表示されますがエディタの動作などには問題ありません。

※この表示は弊社のシステムによってキーボード割り込みによりプログラムを終了させていることによっておこる内容です。



4.INPUT 処理

プログラミング中に記述した INPUT に対してキーボード入力を行う場合は、画面上の INPUT ボックスにキーボード入力します。INPUT ボタンの横の白いボックスをクリックし答えをキーボードから入力し右横のエンターボタンをクリックすることでプログラムは次のプロセスに移ります。

※入力した文字は結果表示部に表示されます。

API 仕様

1.コマンド一覧

API 名	機能
get_sensor_value	超音波センサ計測値を取得します
get_mic_value	マイク計測値を取得します
get_battery_value	電池残量の計測値を取得します
get_info	ロボットの基本情報を取得します
led_on	RGB-LED の指定した色を点灯させます
led_off	RGB-LED の指定した色を消灯させます
sound	指定した時間(msec)だけ指定した周波数の音を発生させます
voice_speed	発話する速度を指定します
voice	発話します
motor_power_on	モーター電源を ON にします
motor_power_off	モーター電源を OFF にします
motor_set_pos	モーターの角度(絶対位置)を指定します
motor_start	モーターを指定した角度(絶対位置)に移動させます
motor_pos_all	全モーター(7 つ)の角度をセットします
motor_angle_time	モーターの角度(絶対位置)、動作時間を指定します
motor_angle_multi_time	全モーター(7 つ)の角度をセットします
get_lib_ver	ライブラリのバージョンを取得します
wait	一定時間待ちます

2.コマンドレスポンス

クムクムに用意されている API コマンドは、実行後必ずレスポンスを文字列で返します。
そのレスポンス文字列は結果表示部に表示されますので、ロボットの動作状態を API レベルでも確認することができます。

レスポンス文字列のフォーマット

‘@’, <内部管理用番号> , <バッテリ残量の目安> , <超音波距離センサの計測値> , <マイクの計測値>

例）@,2,38,999,500

内容

名前	意味
‘@’	@固定
<内部管理用番号>	内部で使用されている管理用の番号です
<バッテリ残量の目安>	クムクムのバッテリ残量の目安(0～100：100 が満充電状態)
<超音波距離センサの計測値>	クムクムで最後に計測した超音波距離センサの計測値
<マイクの計測値>	クムクムで最後に計測したマイクの計測値

get_sensor_value

機能	超音波センサ計測値を取得します
宣言	def get_sensor_value():
引数	なし
戻り値	タプル(要素数 2) int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時は超音波センサの計測値、0 以外(エラー)の時は'NG'が格納されます
説明	クムクムで計測した超音波センサの計測値を取得します
使用例	sens_val = get_sensor_value()　# 超音波センサの計測を行い、計測値を取得します print(sens_val)　# 取得した超音波センサの計測値を出力します

get_mic_value

機能	マイク計測値を取得します
宣言	def get_mic_value():
引数	なし
戻り値	タプル(要素数 2) int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はマイクの計測値、0 以外(エラー)の時は'NG'が格納されます
説明	クムクムで計測したマイクの計測値を取得します
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

get_battery_value

機能	バッテリ残量の目安を取得します
宣言	def get_battery_value():
引数	なし
戻り値	タプル(要素数 2) int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はバッテリ残量の目安を表す値(0-100、100 が満充電状態)、0 以外(エラー)の時は'NG'が格納されます
説明	クムクムで計測したバッテリ残量の目安を取得します
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

get_info

機能	ロボットの基本情報を取得します
宣言	def get_info():
引数	なし
戻り値	タプル(要素数 2) int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムの基本情報を表す一連の文字列、0 以外(エラー)の時は'NG'が格納されます
説明	現在接続しているクムクムの基本情報を取得します クムクムの基本情報は下記のカンマ区切り形式で文字列として取得されます INF,Qumcum,CRETARIA,<ファームウェアバージョン>,<ロボットの MAC アドレス>
使用例	info_str = qumcum. get_info()　　# 現在接続しているクムクムの基本情報を取得します print(info_str)　　# 取得した基本情報を出力します

led_on

機能	RGB-LED の指定した色を点灯させます
宣言	def led_on(color):
引数	color・・・点灯させる LED の色の番号
戻り値	タプル(要素数 2) int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'が格納されます
説明	引数 color には 1~3 までの数値で色を指定します 1・・・赤(Red) 2・・・青(Blue) 3・・・緑(Green) 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

led_off

機能	RGB-LED の指定した色を消灯させます
宣言	def led_off(color):
引数	color･･･消灯させる LED の色の番号
戻り値	タプル(要素数 2) int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'が格納されます
説明	引数 color には 1~3 までの数値で色を指定します 1･･･赤(Red) 2･･･青(Blue) 3･･･緑(Green) 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(3)　# 緑の LED を点灯します qumcum.wait(1)　　# <-- 緑の LED が点灯してから 1 秒待ちます qumcum.led_off(3)　# 緑の LED を消灯します

sound

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
宣言	def sound(freq, time_ms):
引数	freq･･･発生させる周波数(Hz) time_ms･･･音を出力する時間(msec)
戻り値	タプル(要素数 2) int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'が格納されます
説明	引数 freq(周波数)には 30~18,000(Hz)くらい(人の耳で聞こえる)までの音を出すことが可能です time_ms は一回の呼び出しで 9,999(msec)(9.999 秒)まで音を出すことが可能です
使用例	qumcum.sound(880,9999)　# 880Hz の音を 9.999 秒間出します qumcum.wait(10)　　# <-- 10 秒待ちます

voice_speed

機能	発話する速度を指定します
宣言	def voice_speed(speed):
引数	speed･･･発話速度
戻り値	タプル(要素数 2) int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'が格納されます
説明	voice()で発話を行うときの発話の速さの設定を行います 引数 speed には 30（遅い）から 300（速い）までの数値を指定することができます（デフォルトは 100 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

voice

機能	発話します
宣言	def voice(words):
引数	words･･･発話させたい文章など
戻り値	タプル(要素数 2) int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'が格納されます
説明	クムクムの音声合成使い日本語の発話を行います 引数 words には発話させたい文章などをローマ字で指定します 数字やアクセントは専用のタグをつけて指定します ※タグの使用方法は 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

motor_power_on

機能	モーター電源を ON にします
宣言	def motor_power_on(motor_time_ms):
引数	motor_time_ms･･･モータの動作時間(msec)
戻り値	タプル(要素数 2) int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'が格納されます
説明	クムクムのモータの電源を ON にします モータを動かす場合必ずこの関数をよんでください
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

motor_set_pos

機能	モーターの角度(絶対位置)を指定します
宣言	def motor_set_pos(no, pos):
引数	no･･･モータの番号 pos･･･モータの角度
戻り値	タプル(要素数 2) int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'が格納されます
説明	モータを指定した角度に動くように指示をします 引数 no のモータの番号は、1~7 までの数字で割り当ては下記の通りです。 1:右手、2:右足、3:右足首、4:頭、5:左足首、6:左足、7:左手 引数 pos のモータの角度は、0~180 までの数字で指定します (足[右足、右足首、左足首、左足]については、ロボットの構造上 90±30 度[60~120]までしか動かせません) この関数を呼び出しただけではモータは動きません この関数で角度を設定た後、motor_start()関数を呼び出しモータを実際に動作させます
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

motor_start

機能	モーターを実際に動作させます
宣言	def motor_start(no_wait=False):
引数	no_wait(省略可)･･･モータの動作完了を待たないか(True:動作完了を待たない、False:動作完了を待つ)
戻り値	タプル(要素数 2) int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'が格納されます
説明	motor_set_pos 関数などで指定したモータの角度への動作を開始します 引数を True にすると、指定角度に回転し終わるまで次のプログラムは実行されません 引数に False にすると、回転角度かに関係なく本関数が動作した瞬間だけモータが回転します このば場合、sleep などで待ち時間を細かく決めることで最小限の待ち時間でモータを回転させることができるようになり、ロボットの動作速度を細かくコントロールできるようになります
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

motor_pos_all

機能	全モーター(7 つ)の角度を同時にセットする
宣言	def motor_pos_all(pos1, pos2, pos3, pos4, pos5, pos6, pos7):
引数	pos1･･･モーター1(右手)の角度(0~180) pos2･･･モーター2(右足)の角度(60~120) pos3･･･モーター3(右足首)の角度(60~120) pos4･･･モーター4(頭)の角度(0~180) pos5･･･モーター5(左足首)の角度(60~120) pos6･･･モーター6(左足)の角度(60~120) pos7･･･モーター7(左手)の角度(0~180)
戻り値	タプル(要素数 2) int:　処理結果(0:成功、0 以外:エラー) str:　処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'が格納されます
説明	クムクムの標準で搭載されているモータ 7 つすべての角度を指定します (ロボットの構造上足のモータは 90±30 度[60~120]までしか動かせません) この関数を呼び出したタイミングではモータは動きません この関数で角度を設定た後、motor_start()関数を呼び出すとモータが動きます この関数を呼び出した後に motor_start()関数を呼び出すと 7 つのモータが同時に動き出しますのでモータの角度を十分確認してから動かすようにしてください
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 qumcum.motor_power_off()　# モータ電源を OFF にします

motor_angle_time

機能	指定した 1 つのモータの回転角度(絶対位置)と動作時間を指定します
宣言	def motor_angle_time(no, pos, motor_time):
引数	no･･･モーターの番号 pos･･･モーターの角度 motor_time･･･モーターの動作時間
戻り値	タプル(要素数 2) int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'が格納されます
説明	引数 no のモータの番号は、1~7 までの数字で割り当ては下記のとおりです 1:右手、2:右足、3:右足首、4:頭、5:左足首、6:左足、7:左手 引数 pos のモータの角度は、0~180 までの数字です(足[右足、右足首、左足首、左足]については、ロボットの構造上 90±30 度[60~120]までしか動かせません) 引数 motor_time はミリ秒単位で 100~9999(msec)まで指定できます この関数を呼び出したタイミングではモータは動きませんこの関数で角度を設定た後、motor_start()関数を呼び出すとモータが動きます
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.motor_angle_time(4, 0, 500) # 4 番(頭)のモータを 500msec 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 qumcum.motor_angle_time(4, 180, 2000) # 4 番(頭)のモータを 2000msec で 180 度(左)の位置へ動かします qumcum.motor_start() # モータを動かします qumcum.motor_angle_time(4, 0, 100) # 4 番(頭)のモータを 100msec 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 qumcum.motor_angle_time(4, 90, 500) # 4 番(頭)のモータを 500msec で 90 度(正面)の位置へ動かします 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

motor_angle_multi_time

API 名	motor_angle_multi_time
機能	全てのモータ(7 つ)の角度と動作時間を同時にセットします
宣言	def motor_angle_multi_time(pos1, pos2, pos3, pos4, pos5, pos6, pos7, motor_time):
引数	pos1･･･モーター1(右手)の角度(0~180) pos2･･･モーター2(右足)の角度(60~120) pos3･･･モーター3(右足首)の角度(60~120) pos4･･･モーター4(頭)の角度(0~180) pos5･･･モーター5(左足首)の角度(60~120) pos6･･･モーター6(左足)の角度(60~120) pos7･･･モーター7(左手)の角度(0~180) motor_time･･･モーターの動作時間
戻り値	タプル(要素数 2) int: 処理結果(0:成功、0 以外:エラー) str: 処理結果が 0(成功)の時はクムクムからのレスポンス、0 以外(エラー)の時は'NG'が格納されます
説明	クムクムの標準で搭載されているモータ 7 つすべての角度とモータの動作時間を指定します (ロボットの構造上足のモータは 90±30 度[60~120]までしか動かせません) この関数を呼び出したタイミングではモータは動きません この関数で角度を設定た後、motor_start()関数を呼び出すとモータが動きます この関数を呼び出した後に motor_start()関数を呼び出すと 7 つのモータが同時に動き出しますのでモータの角度を十分確認してから動かすようにしてください
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(180, 80, 80, 80, 80, 80, 0, 500)　# 両手を上げて他のモータを 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.wait(1) qumcum.motor_angle_multi_time(0, 100, 100, 100, 100, 100, 180, 100)　# 両手を下げて他のモータを 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.wait(1) qumcum.motor_angle_multi_time(180, 90, 90, 90, 90, 90, 0, 1000)　# ばんざいをする位置へ 1000msec(1 秒)で動くよう指定します qumcum.motor_start()　　# モータを動かします qumcum.motor_angle_multi_time(0, 90, 90, 90, 90, 90, 180, 500)　# 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 qumcum.motor_power_off()　# モータ電源を OFF にします

get_lib_ver

機能	ライブラリのバージョンを取得
宣言	def get_lib_ver():
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	ライブラリのバージョンを表現する文字列を返します
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

wait

機能	一定時間待ちます
宣言	def wait(time_sec):
引数	time_sec･･･待つ時間(単位：秒)
戻り値	なし(0 のみ)
説明	引数 time_sec に 1 を指定した場合は 1 秒待ちます
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