Qumcum

パイソンではじめよう

Python

CRETARIA.

◆はじめに

このテキストでは、Pythonでのプログラミングの基本部分を説明します。
本格的に詳しい文法やテクニックをマスターするためには、市販のPythonの学技術専門書等を1冊お手元にご用意ください。

参考：独習Python 単行本　　翔泳社 (2020/6/22)　（著者）山田祥寛

Pythonとは…

◆誰がいつ創ったの？

Python（パイソン）はオランダ出身・アメリカ在住プログラマーであるグイド・ヴァン・ロッサムによって創り出されたプログラミング言語です。1991年にはPythonの設計哲学をリリースし、2000年にはPython2.0を2008年にはPython3.0をリリースし、2021年6月にはC言語についで2位のランキングを獲得している、今最も注目を浴びているプログラミング言語です。

◆なんで今そんなにPythonなの？

Pythonがこんなに人気のあるのは「学びやすく便利な上将来性がある」ことが大きな理由で、具体的には以下5つの点が人気の理由として挙げられます。

①簡単でわかりやすい

Pythonは、プログラムの記述ルールが明確に定義されています。C言語のように勝手気ままにプログラマーの癖で書くことは許されません。ですから、学習もしやすく、また誰が作っても読みやすく理解しやすくメンテナンス性にも優れていて、改造や修正もやりやすいプログラミング言語です。

②どんなパソコンでも動く

専門的に言うと、プラットフォームを限定しないといいます。つまり、WindowsやMac、Linuxなど同じプログラムがそのまま他のパソコンで動きます。Pythonは、C言語のように機械語に翻訳されて動く言語ではないので、Windows用やMac用などそれぞれ専用にプログラムを作る必要がありません。

③コントロールの手段が豊富

多くのプログラマーが、いろいろな処理やいろいろな装置に使えるライブラリーをどんどん開発し、ネット上に公開しています。自由に使える優秀なライブラリーがたくさんあります。
しかも、Pythonのライブラリーは無料のものが多いため、手軽にいろいろなプログラムを作っていくことが可能です。

④動かしながら作れる

Pythonはインタプリタという形式で動くため、プログラムは作るとすぐに実行して瞬時に試すことができます。C言語のように機械語に変換してから動くまで待つといったわずらわしさがありません。作ってすぐに結果を見ることができるので、気軽にトライ＆エラーを繰り返すことができ、初心者の習得にも向きます。

⑤これからの時代に必要な処理が簡単にできる

Pythonにはインターネットを利用したライブラリーや、AI・ビッグデータの処理や統計分析などこれからの時代に必要な処理を実現できるライブラリーがたくさんあります。大量のデーターベースを利用し、AIで判定して装置を動かしたりすることはこれからの時代には絶対に必要な処理で
この先、間違えなくPythonでのプログラミングは必要で伸びていく技術です。

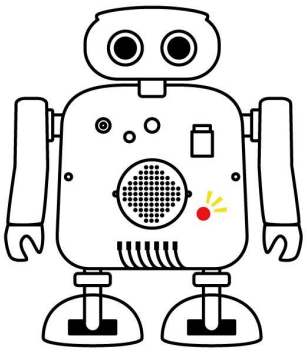
クムクムとPython…

◆まずは？

クムクムのホームページに、ブラウザー上Pythonもクムクムも動くエディターを用意しました。WEBブラウザ上で動くので、Pythonのインストールも設定も必要ありません。最新技術を使ってブラウザーからBluetooth経由でクムクムのすべての機能をPythonプログラミングで楽しめます。Pythonの基本文法はすべて使えるので、気軽に簡単にクムクムとPythonを使った未来ロボットのプログラミングを始めることができます。さあ、今すぐ、画面から飛び出したロボットプログラミングに挑戦しましょう！

◆そのあとは？

WEBブラウザーでPythonやクムクムなどの知識を付けた後はいよいよインターネットやAIサービスなどを使った本格的なPythonプログラミングに挑戦です。AIやインターネットサービスなど、世界にある沢山のライブラリーを探しだしてクムクムと繋ぎあわせて夢の先端プログラミングに挑戦してください。しゃべりかけた声を分析して世界各国のことばに変えてくれるクムクムや、好きなラジオにつないで音楽を鳴らしてくれるクムクムなどアイディアいっぱい夢いっぱいです。



なぜクムクムを使うのか？

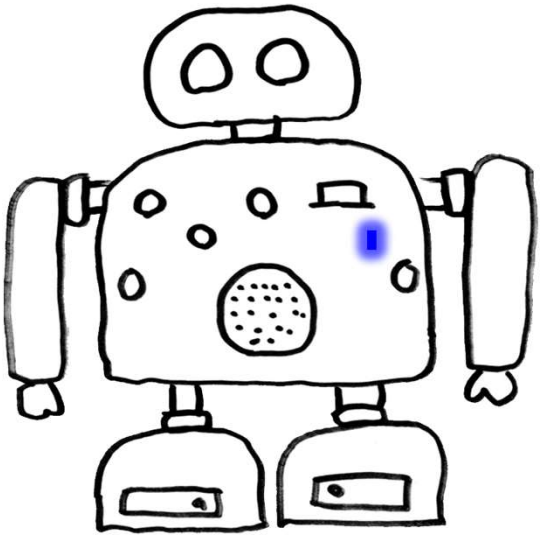
これまでの時代は、「ハードはハード」「ソフトはソフト」「データはデータ」と切り分けられており、技術者の知識も腕もそれぞれ切り分けられていました。しかしこれからの時代における装置（ドローンや自動運転の車、ロケット、空飛ぶ車、そして外科手術ロボットなど…）は自分自身以外とどんどん通信を行い、日々増え続ける新たな知識データベースを利用して、どんどん新しい動きができるように変化成長をする装置になります。これらの技術に対応できるエンジニア・プログラマーに求められるのは、「切り分けられた技術」ではなく「総合的に理解し利用できる技術」です。

クムクムでは、ロボットやPythonそしてインターネットサービスにおけるデータなをどを総合的に学ぶことでこれから必要な「先端IT技術」をマスターすることができます。

文部科学省の科学技術・学術政策研究所は、2035年には「量子化コンピューター」の実用化と「空飛ぶ車」という未来の科学技術の予測をまとめました。

2019年4月23日に経済産業省が発表した「IT人材需給に関する調査」では2030年に「従来型IT人材」が10万人余ると発表していますが、 4月24日同報告書を紹介した日本経済新聞の記事には「先端人材55万人不足　経産省試算　30年、AIやIoT」という見出しを付けています。

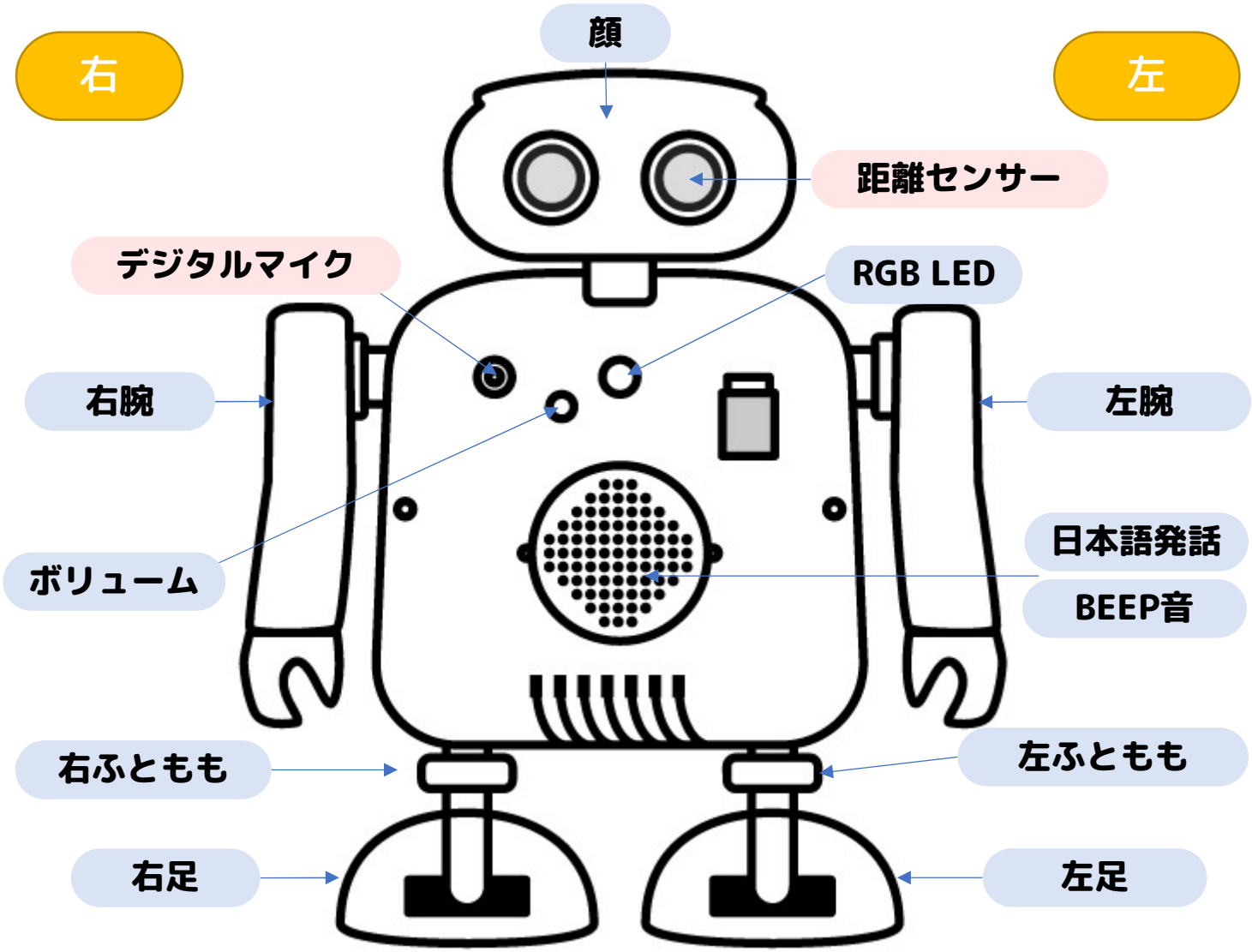
10万人余る「従来型IT人材」とは受託開発や保守運用を担うエンジニアで、 55万人不足する「先端IT人材」とは「AIやビッグデータ、IoT等、第4次産業革命に対応した新しいビジネスの担い手として、付加価値の創出や革新的な効率化等により生産性向上等に寄与できるIT人材」と定義しています。



Ready

クムクムのパーツ

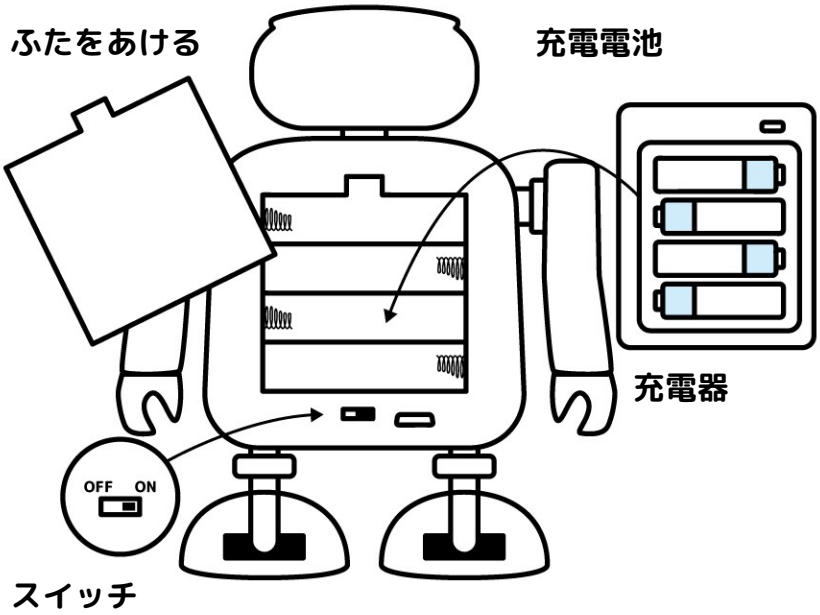
クムクムには入力・出力が学べるパーツを用意しています。
動作に必要なモーター、センサー、マイク、音・光など装置に付随する技術を一通り学ぶことができます。



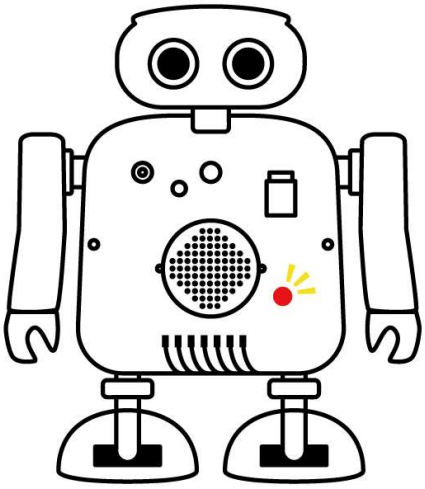
パーツ	動作	機能
RGB-LED	出力	RGB3色のLEDの点灯・消灯を制御できます
日本語発話	出力	ローマ字等で指定した任意の日本語を喋らせることができます
BEEP音	出力	任意の高さ（周波数）と長さ（秒）のブザー音を鳴らすことができます
顔	出力	顔のモーターを左右自由な角度に回転させることができます
左腕・右腕	出力	左右任意の腕のモータを上下自由な角度に回転させることができます
左・右ふとももも	出力	足のつけねのモータを左右自由な角度に回転させることができます
左・右足	出力	左右任意の足のモータを上下自由な角度に回転させることができます
距離センサー	入力	目前にある障害物までの距離を計測することができます
デジタルマイク	入力	周囲の音の大きさを計測することができます

クムクムの準備

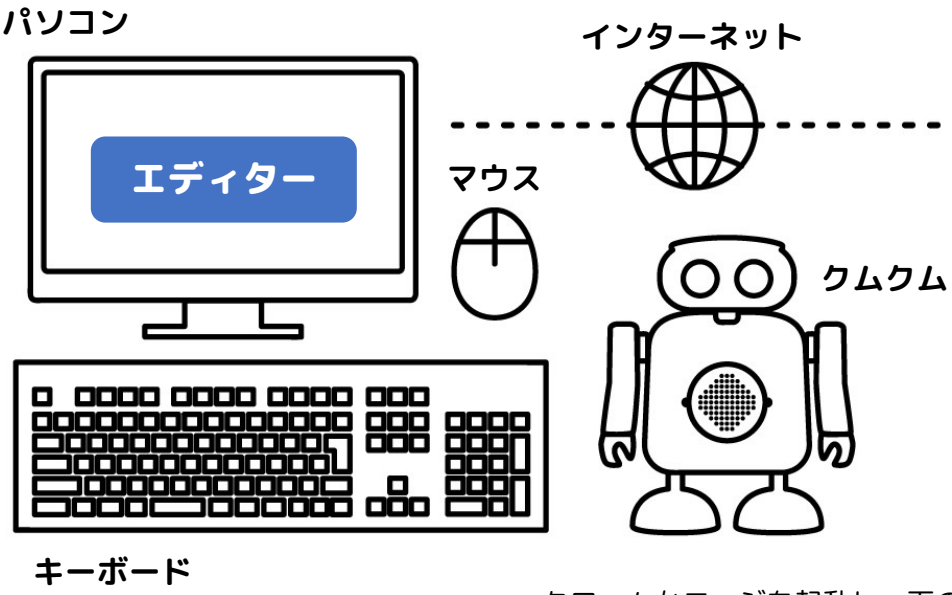
クムクムに充電電池をいれてスイッチをONにします。



クムクムの電源をONにすると赤いLEDが光ります。



Pythonエディターを起動



クロームかエッジを起動し、下のURLアドレスに接続し、クムクム専用のPythonを起動します。

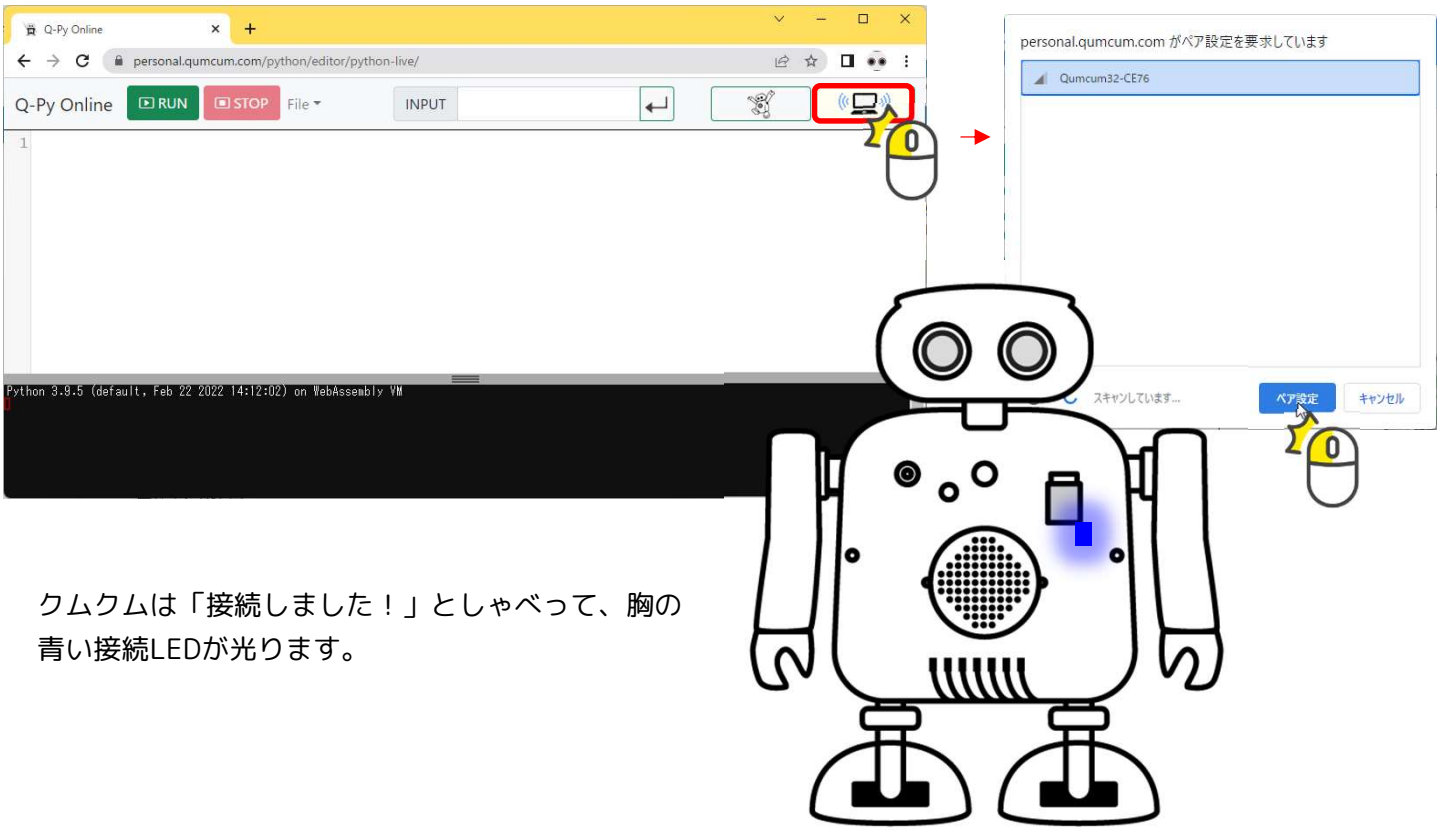
https://personal.qumcum.com/python/editor/python-live/

エディターとクムクムを接続

◆Bluetoothでワイヤレス接続

右端の接続ボタンをクリック

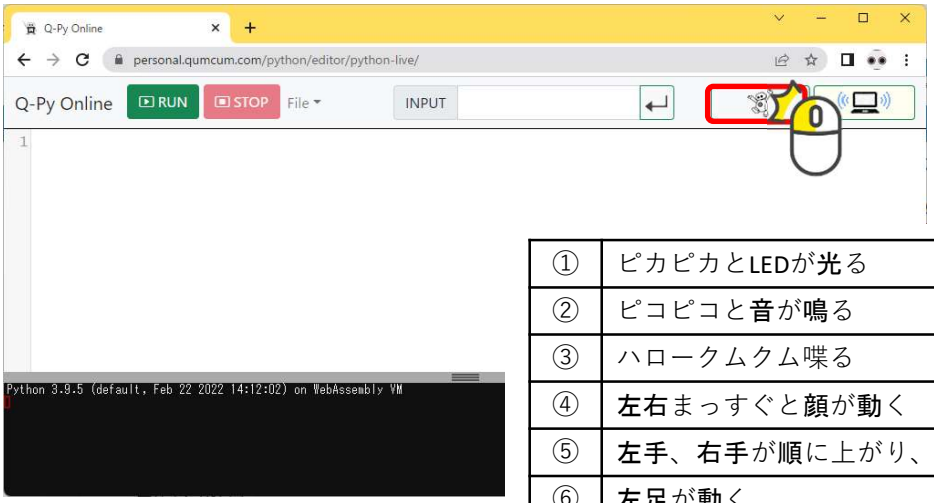
ロボットとペアリング



クムクムは「接続しました！」としゃべって、胸の青い接続LEDが光ります。

◆クムクムの動作確認

接続ボタンの左隣のハローボタンをクリックします。



①	ピカピカとLEDが光る
②	ピコピコと音が鳴る
③	ハロークムクム喋る
④	左右まっすぐと顔が動く
⑤	左手、右手が順に上がり、バンザイのあとまっすぐ戻る
⑥	左足が動く
⑦	右足が動く
⑧	まっすぐ戻る

この動きがちゃんとできなかった場合の原因は、ほとんどが電池不足です。また、通信状態が悪いと、途中で動きがおかしくなることもあるので、おかしい動きの時は、何度か試してみましょう。

プログラミングの準備

◆マニュアルの準備

本紙以外に、セッティングマニュアルとプログラミングマニュアルをホームページからあらかじめダウンロードをしてお手元にご用意ください。細かい操作方法や、クムクムをコントロールするためのコマンドなど、ここでは説明しきれなかった部分を載せています。
特にプログラミングマニュアルの関数リファレンス（7ページ以降）にはクムクムをコントロールするためのコマンドを細かく説明しています。ここでのサンプルを試しながらどんどん新しいコマンドにも挑戦してください。

セッティングマニュアル

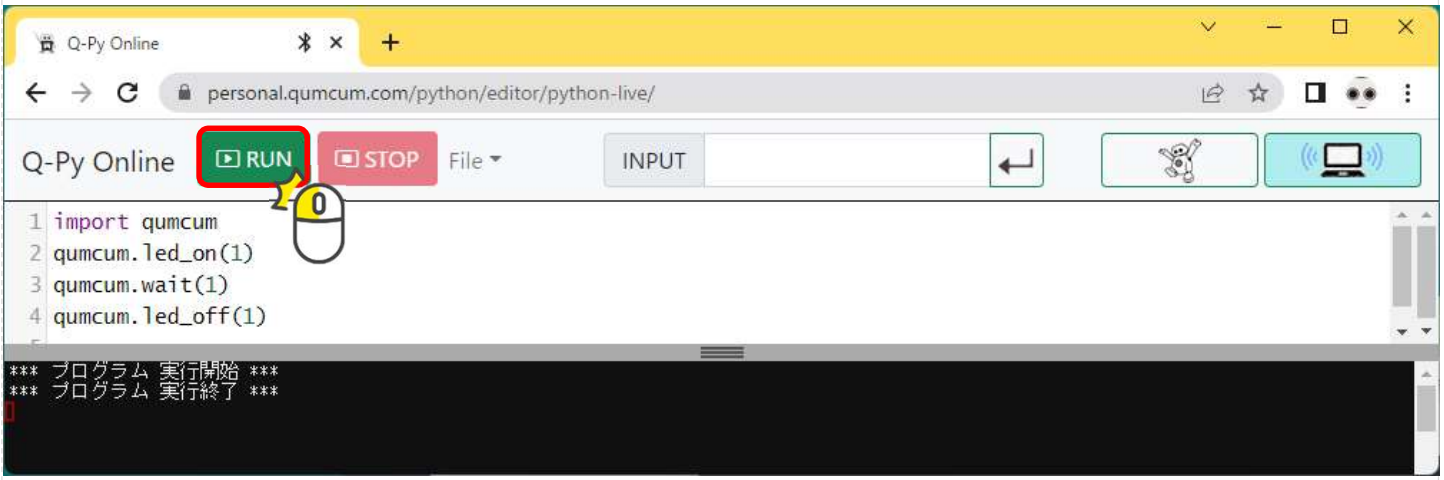
https://qumcum.com/wp-content/uploads/2022/08/python_manual_settingx.pdf

プログラミングマニュアル

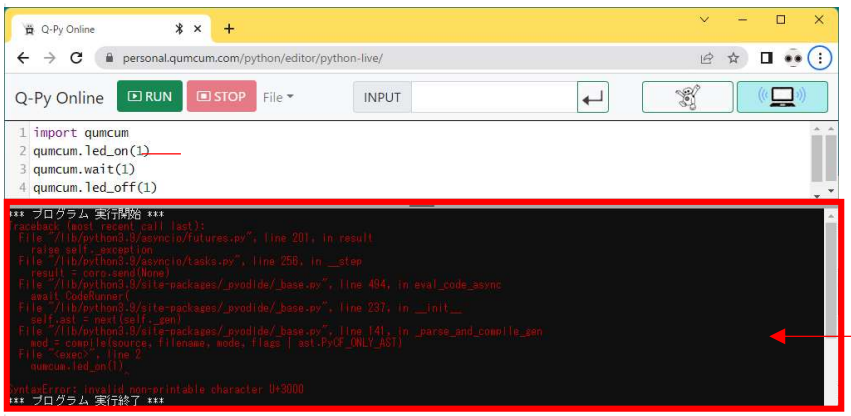
https://qumcum.com/wp-content/uploads/2022/08/python_manual_programming.pdf

◆プログラミングと動作

プログラム入力後は[RUN]ボタンをクリックしプログラムを実行します。

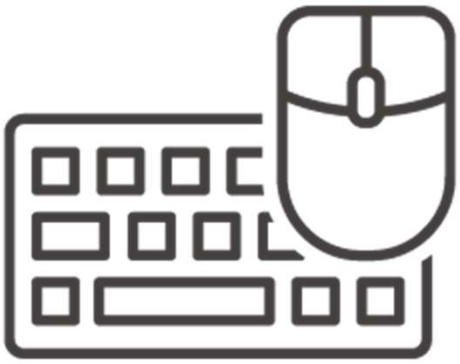


◆どこかにミスがある時



何もミスがないように見えるが、実は赤い部分に全角のスペースが入力されているのでエラーとなりました。

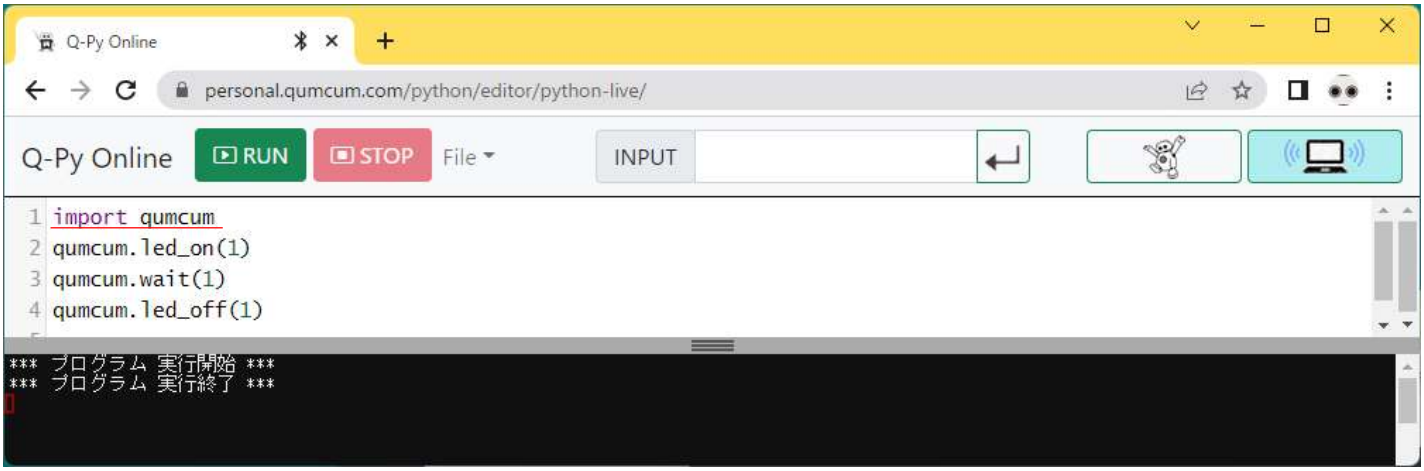
エラーがあると赤い文字で表示



Start

1.ライブラリーの読み込み

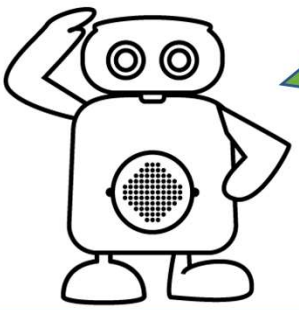
Pythonでは、標準に用意されている文法以外を使用する場合、import という命令を使って使いたいライブラリーを読み込む記述をします。インターネットに接続していろいろな処理をするプログラムを作りたいときや、特殊な装置を動かしたい場合など必要な時に必要な機能を読み込んで使います。
クムクムをコントロールするコマンドも、標準のPythonにはない特別な命令なので、プログラムの最初で import qumcum と記述しクムクムのライブラリーを読み込みます。



クムクムエディターでimportできるライブラリー

クムクムエディターはPythonの基礎をクムクムを使って学ぶためだけに用意されているので、マニュアルの2ページに記載されたライブラリーしかimportすることができません。
インターネットやAIなどのライブラリーはWEB上のクムクムプログラムからはセキュリティー上使用することができません

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



普通は、自分のパソコンにPythonやエディターをインストールしてプログラミングを行います。難しくて面倒なのですが、そのかわり、世界にあるたくさんのライブラリーをどんどんダウンロードしimportして使えるようになります。ー
基礎をマスターしたら、インターネットに接続したり、データーを分析したり、AIを動かしたり沢山のライブラリーを使えるように挑戦しましょう。

2.変数（へんすう）とデータ

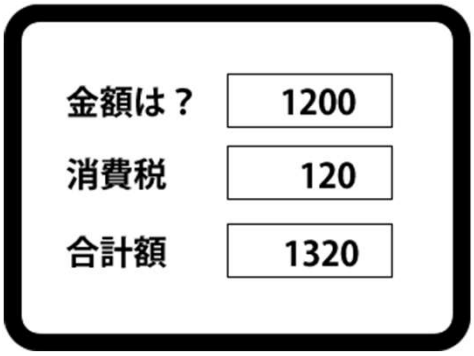
LED

マニュアル7P

プログラミング言語に必ず用意されている「変数」という機能を使いクムクムのLEDをコントロールします。
変数は、プログラミング中にどうしても一時的に記憶させたいメモリーのことを示し、好きな名前（予約語以外）を付けて好きな文字や数字をそこに記憶させます。この変数はプログラムが動いている間だけ使えるもので、プログラムが終了するとメモリーの中は全部消えてなくなります。また、変数は、いつでも好きな数字や文字に変化させることができます。

◆変数を使う例

たとえば、入力した金額から消費税を計算して表示するプログラムを作る時、まずキーボードから入力された金額を一時的に記憶するメモリーが必要です。そして、金額から計算した消費税額を記憶するためのメモリーや金額と消費税を足して合計額を記憶するメモリーなど一時的な記憶場所として変数を利用します。



プログラムでは答えを左に計算式を右に書く

```
kingaku=input("金額は？")
```
⇨ 金額を聞いて入力された数字をkingakuという変数に入れる（代入）

```
zei=kingaku * 0.1
```
⇨ kingakuの中に入った数字に0.1（10%）を掛けてzeiという変数に代入

```
goukei=kingaku + zei
```
⇨ kingakuとzeiのの中に入った数字を足してgoukeiという変数に代入

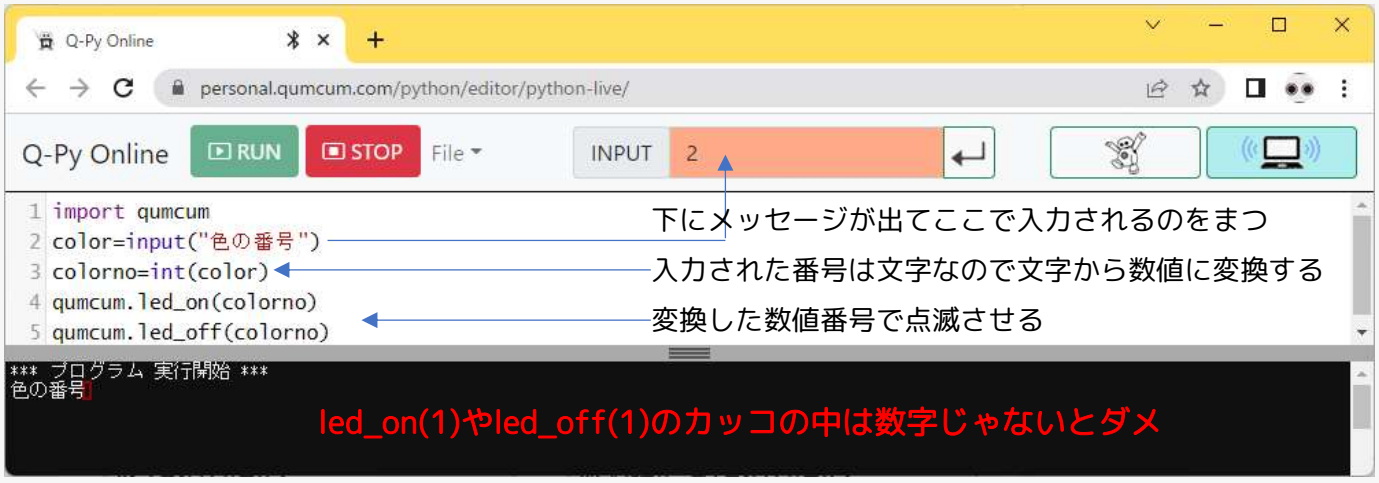
01.赤を1回点滅させる

```
1 import qumcum
2 qumcum.led_on(1)
3 qumcum.led_off(1)
```

02.変数を使って赤を1回点滅させる

```
1 import qumcum
2 color=1
3 qumcum.led_on(color)
4 qumcum.led_off(color)
```

03.キーボードで入力した色を光らせる



プログラミング言語では同じ1や2でも、数値と文字を厳密に違うものとして考えます。
Pythonの入力命令 input ではキーボードから入力されたデータはすべて文字として＝の左側の変数に代入されるため文字として入力された数字を数値に変換する処理が必要です。

変数と繰り返し

プログラミングでは、同じことを何回か繰り返す場合「繰り返し」という方法を使います。
繰り返しは、回数を数えるための変数を用意して繰り返したり、単純にある繰り返しの法則で変数の数値を変化させてその数値を使ってプログラムを面白く操ったりすることができます。

01.単純に数を数えて繰り返す

```
1 import qumcum
2 for kai in range(5):
3     qumcum.led_on(1)
4     qumcum.led_off(1)
```

コロンを忘れない

4つまたは2つのスペースかTabキーを1回おして段落をかえる
※インデントをつける

Pythonでは繰り返しを行う場合、
これから何回繰り返すよ～と指示をした後、繰り返す命令の範囲を段落を分けて（インデントを付けて）ブロック分けします。
for kai in range(5)　は、「これから5回繰り返すよ～その回数はkaiにメモリーするよ」という指示で、kaiには 回数ごとに0,1,2,3,4という数値が代入されます。

02.数値を変化させて操る

```
1 import qumcum
2 for color in range(1,4):
3     qumcum.led_on(color)
4     qumcum.led_off(color)
```

for color in range(1,4)　は、「これからcolorという名前の変数の中身を1以上4未満（1,2,3)と変化させてね！」という指示を出します。
そして、そのcolorの数値でLEDを光らせたり消したりしています。

03.繰り返しの組み合わせ

```
1 import qumcum
2 for kai in range(10):
3     qumcum.led_on(1)
4     qumcum.led_off(1)
5 for color in range(1,4):
6     qumcum.led_on(color)
7     qumcum.led_off(color)
```

01.02のプログラムを続けてみます。
赤点滅を10回繰り返した後全色の点滅を1回ずつ行います

04.繰り返しの中の繰り返し

```
1 import qumcum
2 for kai in range(10):
3     for color in range(1,4):
4         qumcum.led_on(color)
5         qumcum.led_off(color)
```

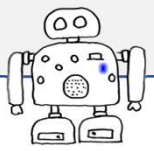
4つまたは2つのスペースかTabキーを1回おして段落をかえる
※2重にインデントをつける

全色の点滅を1回ずつを10回行います。

05.逆方向に数字を変化させる

```
1 import qumcum
2 for color in range(3,0,-1):
3     qumcum.led_on(color)
4     qumcum.led_off(color)
```

for color in range(3,0,-1)　は、「これからcolorという名前の変数の中身を3から0より大きい数値（1）まで-1ずつ変化させてね！」という指示を出します。



2.配列

BEEP

マニュアル8P

変数と繰り返しと配列を使ってクムクムのBEEPをコントロールしてみます。
変数は1つの名前のメモリーに1つだけのデータを記憶していましたが、配列という方法では1つの名前のメモリーに沢山のデータをグループとして記憶させることができます。
記憶させたデータを順序良く取り出したり並び替えたりすることもPythonではできます。

01.500Hz～2000Hz未満まで50Hzずつ飛ばしで鳴らす

```
1 import qumcum
2 for freq in range(500,2000,50):
3     qumcum.sound(freq,100)
```

for freq in range(500,2000,50)　は、「これからfreqrという名前の変数の中身を500以上2000未満で50ずつと変化させてね！」という指示です。そして、その高さで100ミリ秒音を鳴らします。

02.逆回転追加

```
1 import qumcum
2 for freq in range(500,2000,50):
3     qumcum.sound(freq,100)
4 for freq in range(2000,500,-50):
5     qumcum.sound(freq,100)
```

03.確実に1秒鳴らす

```
1 import qumcum
2 for freq in range(500,2000,100):
3     qumcum.sound(freq,1000)
4     qumcum.wait(1)
```

3行目：1000ミリ秒（1秒）ならしたあと
4行目：1秒まつ

04.配列で鳴らすドレミ

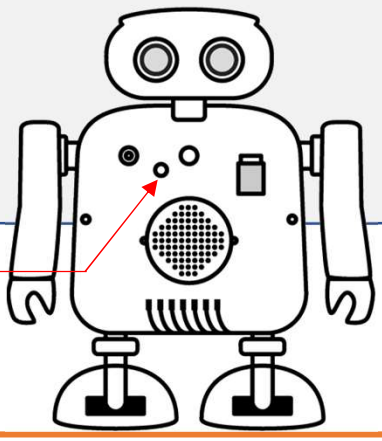
```
1 import qumcum
2 onkai=[522,586,654,698,784,880,986,1044]
3 for freq in onkai:
4     qumcum.sound(freq,100)
5     qumcum.wait(0.1)
```

ドレミファソラシドをonkaiという配列に用意して鳴らす
回数はonkaiの中のデーターの数で勝手にPythonが決めてくれる。

05.配列をひっくり返すテクニック

```
1 import qumcum
2 onkai=[522,586,654,698,784,880,986,1044]
3 for kai in range(2):
4     for freq in onkai:
5         qumcum.sound(freq,100)
6         qumcum.wait(0.1)
7     onkai.reverse()
```

2回目を繰り返す前に配列の中をひっくり返す。



音が小さい時や**大きすぎる**時は、ここのボリュームを細い+ドライバーで右に優しく回して音量を調整してください。

文字列と数値

VOICE

マニュアル9P

文字と数値がプログラミング言語では違うものだということをクムクムのVOICE機能を使ってマスターします。

01.単純に喋らせる（こんにちは）

```
import qumcum
qumcum.voice_speed(30)
qumcum.voice("konnitiwawa")
```

喋らせる言葉はvoice命令の中に“ “で囲ったローマ字で指定します。言葉は耳に聞こえる音で書きます。こんにちは　は　“konnitiwa”と書きます。

02.アクセントをつけてみる

```
import qumcum
qumcum.voice_speed(100)
qumcum.voice("konnitiwawa")
qumcum.voice("ko'nnitiwawa")
qumcum.voice("ko;nnitiwawa")
qumcum.voice("ko/nnitiwawa")
qumcum.voice("konnitiwawa?")
qumcum.voice("kon,nitiwawa")
```

母音のあとに‘ ; / , 語尾に? などの記号を付けるとアクセントをつけたしゃべり方ができます。

03.数値を単純に喋らせる

```
import qumcum
qumcum.voice("watasinoheyawa")
qumcum.voice_num(1105)
```

1105を数値として指定するので“”で囲みません。
「私の部屋は、いちいちぜろごう」
※最初に0は使えません

04.数字とローマ字をまぜて喋らせる

```
import qumcum
qumcum.voice("dennwawa")
qumcum.voice_numalpha("09012345678")
qumcum.voice_numalpha("090-1234-5678")
```

最初に0がついた数字を喋らせる場合は、numalphaというコマンドを使い　“ ” で数字を文字として指定します。

05.桁を付けて数字を読み上げる

```
import qumcum
qumcum.voice("bokunotyokinwa")
qumcum.voice_math(136587424850)
qumcum.voice("endesu")
```

05.分・本をつけて数字を読み上げる

```
import qumcum
for num in range(10):
    qumcum.voice_fun(num)
    qumcum.voice_pon(num)
```

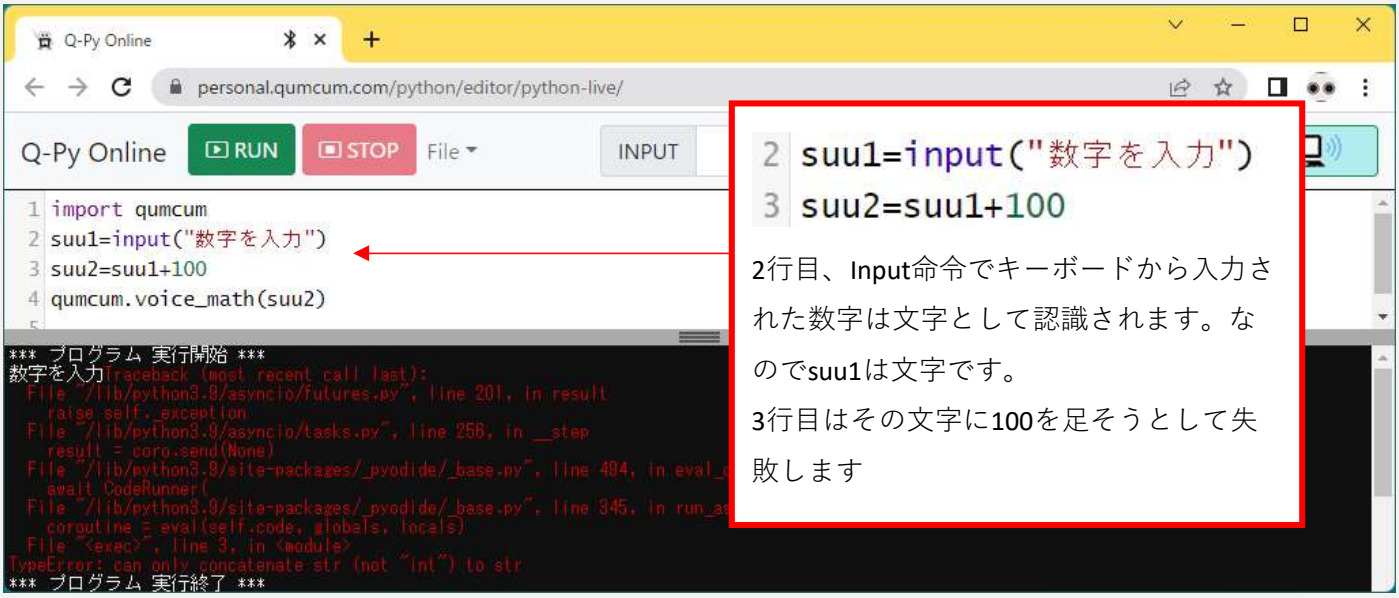
数値と計算

VOICE

マニュアル9P

コンピューターが最も得意とする数値の計算をクムクムのお喋りでマスターします。

01.入力した数字に100を足した数字を喋らせる（エラーがでます）




```
2 suu1=input("数字を入力")
3 suu2=suu1+100
```

2行目、Input命令でキーボードから入力された数字は文字として認識されます。なのでsuu1は文字です。

3行目はその文字に100を足そうとして失敗します

02.入力した数字に100を足した数字を喋らせる（正解）

```
1 import qumcum
2 suu1=input("数字を入力")
3 suu=int(suu1)
4 suu2=suu+100
5 qumcum.voice_math(suu2)
```

2行目で入力された文字のsuu1を、3行目のint(suu1)で数値に変えてsuuに代入。

4行目は数値に変換されたsuu1→suuに100を足してsuu2に代入し5行目で喋らせています。

03.2つの数字の計算

```
1 import qumcum
2 suu1=150
3 suu2=100
4 qumcum.voice_math(suu1)
5 qumcum.voice("kakeru")
6 qumcum.voice_math(suu2)
7 qumcum.voice("wa")
8 qumcum.voice_math(suu1*suu2)
```

変えて実験してみよう

足し算	tasu	suu1+suu2	+
引き算	hiku	suu1-suu2	-
掛け算	kakeru	suu1*suu2	*
割り算	waru	suu1/suu2	/
あまり	amari	suu1%suu2	%

ロボットアクション

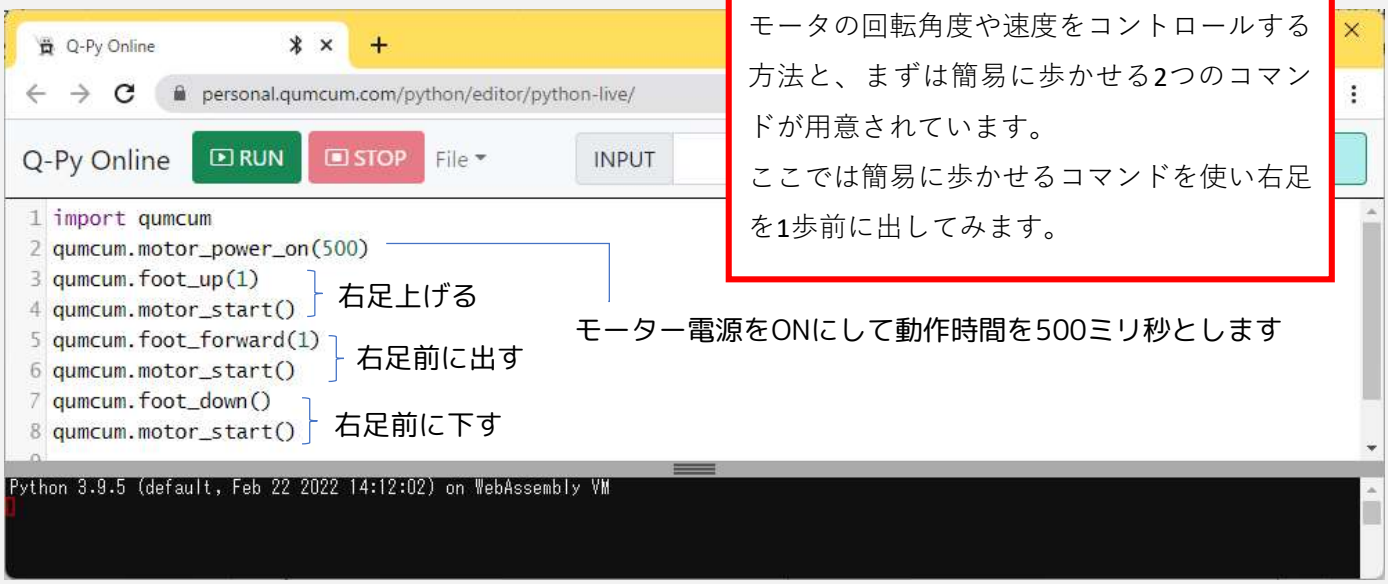
MOTOR

マニュアル14P

クムクムのモータをコントロールしていろいろ動かしてみます。

01.右足を1歩前に出す

モーターを回転させるコマンドは、1個1個のモータの回転角度や速度をコントロールする方法と、まずは簡易に歩かせる2つのコマンドが用意されています。
ここでは簡易に歩かせるコマンドを使い右足を1歩前に出してみます。



モーター電源をONにして動作時間を500ミリ秒とします

右足上げる

右足前に出す

右足前に下す

このプログラムを動かす前に、エディタ画面のどこかをクリックして、エディタをアクティブ状態にし
次にPCのキーボードで [Ctrl]+[Q]を押すとロボットはまっすぐの状態にします。
モータプログラムを行う場合には必ずまっすぐ動作を行うようにしましょう。

02.手と足の動きを追加する

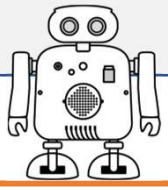
```
1  import qumcum
2  qumcum.motor_power_on(500)
3  qumcum.motor_set_pos(1,180)   右手
4  qumcum.motor_set_pos(7,0)     左手
5  qumcum.motor_set_pos(4,0)     顔
6  qumcum.foot_up(1)
7  qumcum.motor_start()
8  qumcum.foot_forward(1)
9  qumcum.motor_start()
10 qumcum.motor_set_pos(1,0)     右手
11 qumcum.motor_set_pos(7,90)    左手
12 qumcum.motor_set_pos(4,180)   顔
13 qumcum.foot_down()
14 qumcum.motor_start()
```

足を上げる時と下げる時に、同時に両腕と顔の動きを付けました。

03.歩く方向を変えるためにその場で回転

```
1  import qumcum
2  qumcum.motor_power_on(500)
3  for kai in range(5):
4      qumcum.foot_up(1)
5      qumcum.motor_start()
6      qumcum.foot_forward(1)
7      qumcum.motor_start()
8      qumcum.foot_down()
9      qumcum.motor_start()
10     qumcum.motor_set_pos(6,90)
11     qumcum.motor_start()
```

左足を1歩前に出した後、右足をまっすぐに戻すと、その場でちょっとだけ回転する動きになります。その動きを5回繰り返すことで歩く方向を変えていきます。



演算とアクション

MOTOR

04.右足一歩下がる

```
import qumcum
qumcum.motor_power_on(500)
qumcum.foot_up(1)
qumcum.motor_start()
qumcum.motor_set_pos(6,60)
qumcum.motor_set_pos(3,60)
qumcum.motor_start()
qumcum.foot_down()
qumcum.motor_start()
```

04.右後ろ回転

```
import qumcum
qumcum.motor_power_on(500)
qumcum.foot_up(1)
qumcum.motor_start()
qumcum.motor_set_pos(6,60)
qumcum.motor_start()
qumcum.foot_down()
qumcum.motor_start()
qumcum.motor_set_pos(6,90)
qumcum.motor_start()
```

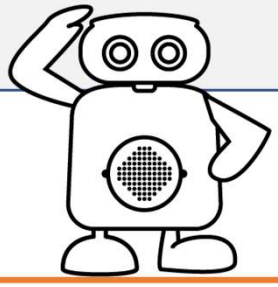
05.腕と顔が5回ランダムな位置に動く

```
import qumcum
import random
kotoba=["arayo","korayo","nanntoma","dokkoisyo","maamaa"]
qumcum.motor_power_on(500)
qumcum.motor_pos_all(90,90,90,90,90,90,90)
qumcum.motor_start()
for i in range(5):
    lhand=random.randint(0,180)
    rhand=random.randint(0,180)
    face=random.randint(0,180)
    qumcum.motor_pos_all(lhand,90,90,face,90,90,rhand)
    qumcum.voice(kotoba[i])
    qumcum.motor_start()
    qumcum.wait(0.5)
```

ほとんどのプログラミング言語は、乱数といって、自分が思いつくランダムな数字を出力してくれる機能があります。2行目の import文では random のライブラリーをインポートしています。

8～10行目では繰り返されるたびに0～180までの何かの数値がそれぞれ lhand,rhand,face の各変数に代入され、その数値を角度として使い11行目でクムクムのそれぞれのモータの回転角度を決定します。

※足のモータはめちゃくちゃな値にするとロボットが転げたり思わぬ動きをして故障の原因にもなるので、ここでは90度のまま固定にしています。



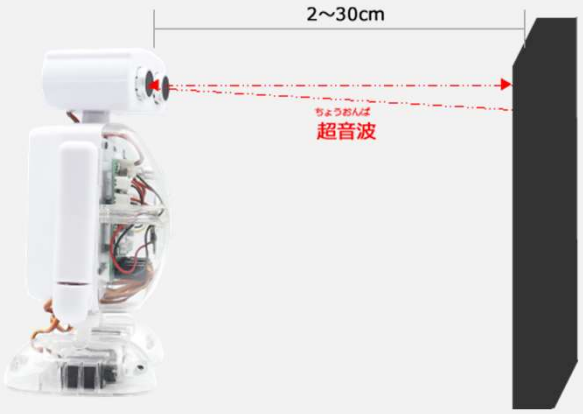
無限ループ

マニュアル22P

回数を指定した繰り返しのほかに、ずっと終わらない「無限ループ」という繰り返し方法がプログラミング言語にはあります。たとえば距離センサーやマイクなど外部から入力されるものは、いつ入力されるかわからないため、この無限ループを使ってずっと監視しておくプログラムを作ったりします。その方法をポーリングと言います。
ポーリングは、常に同じところをくるくる回っているため、回っている間ほかの処理ができなくなります。
ちょっと昔の、単機能で単純な処理しか行わないような装置ではポーリングはよく使われていましたが、最近は同時にいくつもの入力や処理が動くため、ポーリングはあまり使われず、代わりにスレッドという処理に変わっています。
まずはクムクムでは、センサーやマイクの入力を学ぶために、ポーリング処理で動かしてみます。

01.無限ループで計測して距離を表示

```
1 import qumcum
2 while True:
3     kyori=qumcum.get_sensor_value()
4     print(kyori)
5     qumcum.wait(1)
```

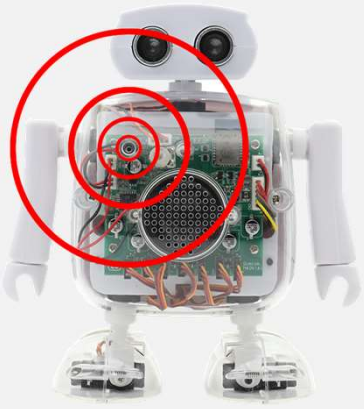


1秒ごとに計測してその距離を表示 print します

プログラムを動かしたら、本など音波を反射するものでクムクムの目を遮ってください

02.無限ループで音を検知して表示

```
1 import qumcum
2 while True:
3     oto=qumcum.get_mic_value()
4     print(oto)
5     qumcum.wait(1)
```



音は0~100までのレベル数値

1秒おきのチェックなので瞬間的な音の大きさが取得できません。
プログラムを動かしたら、マイクの前で、「おおおお~」と大きな声で長い間声を出してください。

```
1 import qumcum
2 while True:
3     kyori=qumcum.get_sensor_value()
4     print(kyori)
5     qumcum.wait(1)
```

Pythonの無限ループの書き方（Tだけが大文字）

インデントのブロックが繰り返される

Forの繰り返しと同じように2個または4個のスペースでインデント

判断

全てのプログラミング言語に用意されている判断という処理を使い、計測した距離や音の大きさを判断していろいろな処理を行ってみます。

03.なにか障害物があったらLED点滅

```
import qumcum
while True:
    ky=qumcum.get_sensor_value()
    if ky<999:
        qumcum.led_on(1)
        qumcum.led_off(1)
        qumcum.wait(0.5)
```

計測した距離の値が999より小さければ何か障害物があったことになります。ここでは4行目の if文を使って999より小さければと判断しています。

04.なにか音がなったらLED点滅

```
import qumcum
while True:
    ot=qumcum.get_mic_value()
    if ot>70:
        qumcum.led_on(1)
        qumcum.led_off(1)
        qumcum.wait(0.5)
```

計測した距離の値が0より大きければ何か音がなったことになります。ここでは4行目の if文を使って70より大きければと判断しています。

05.なにか障害物があったらLED点滅

```
import qumcum
while True:
    ky=qumcum.get_sensor_value()
    if ky<999:
        qumcum.voice("kiken")
    else:
        qumcum.voice("anzen")
    qumcum.wait(1)
```

障害物があったら「キケン」なかったら「アンゼン」としゃべります。

06.3段階の判定

```
import qumcum
while True:
    ky=qumcum.get_sensor_value()
    print(ky)
    if ky<10:
        qumcum.voice("kowai")
    elif ky<20:
        qumcum.voice("yabai")
    elif ky<30:
        qumcum.voice("dame")
    qumcum.wait(0.5)
```

計測した距離の値が10より小さければ “こわい” 10～19なら（20より小さい）“やばい” 20～29までなら（30より小さい）“だめ” としゃべる。

07.測った距離を喋る

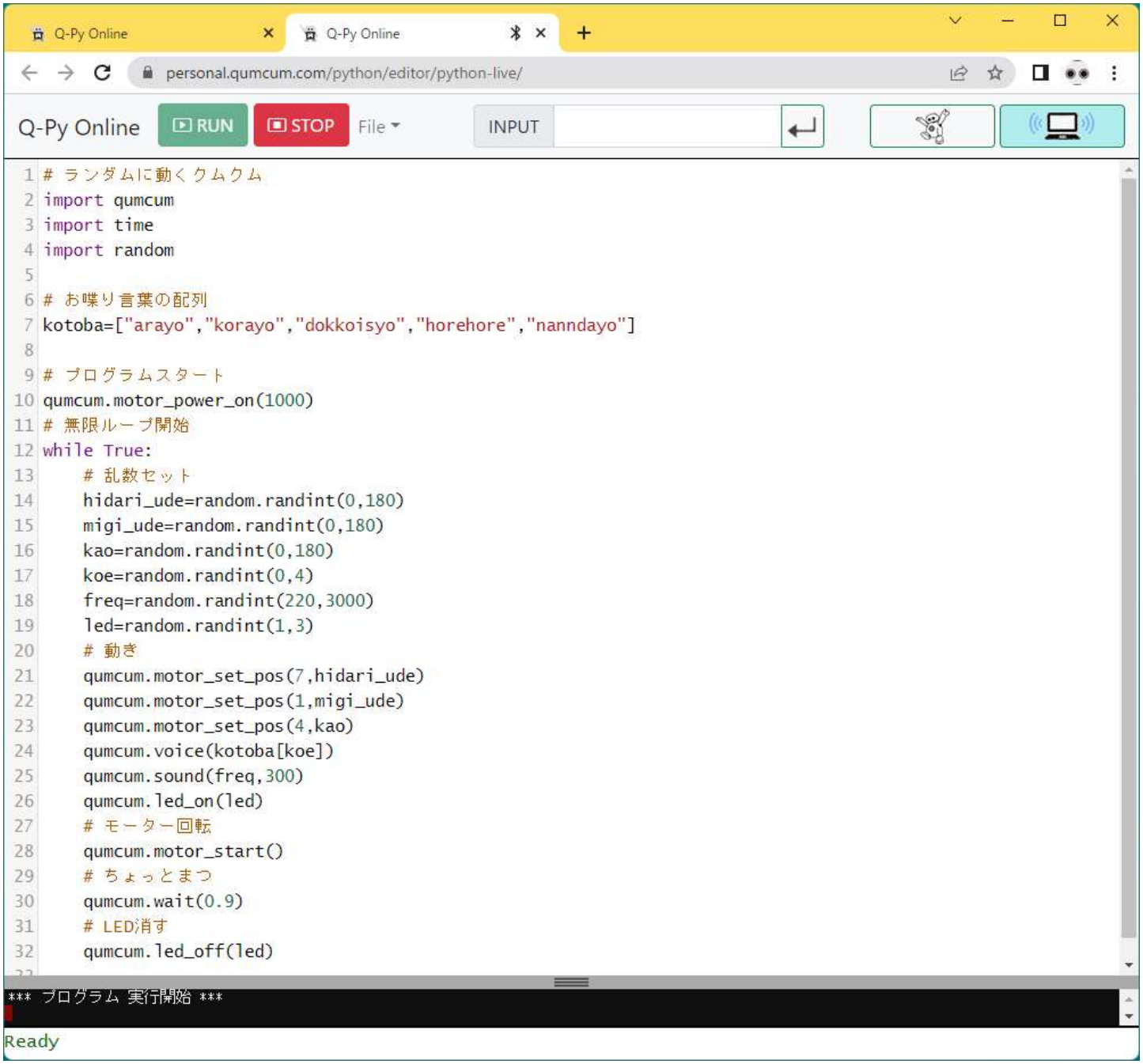
```
import qumcum
while True:
    ky=qumcum.get_sensor_value()
    if ky!=999:
        qumcum.voice("kyo'ri")
        qumcum.voice_math(ky)
        qumcum.voice("se'nti")
        qumcum.wait(1)
```

判定式の書き方

If ky<=30	kyが30以下なら
If ky<30	kyが30未満なら
If ky==30	kyが30なら
If ky!=30	kyが30以外なら
If ky>30	kyが30より大きければ
ifky>=30	kyが30以上なら

テクニック1

乱数と配列を使ってモーター・LED・BEEP・VOICEを使って永遠にランダムな動きをするクムクムを作ってみます。



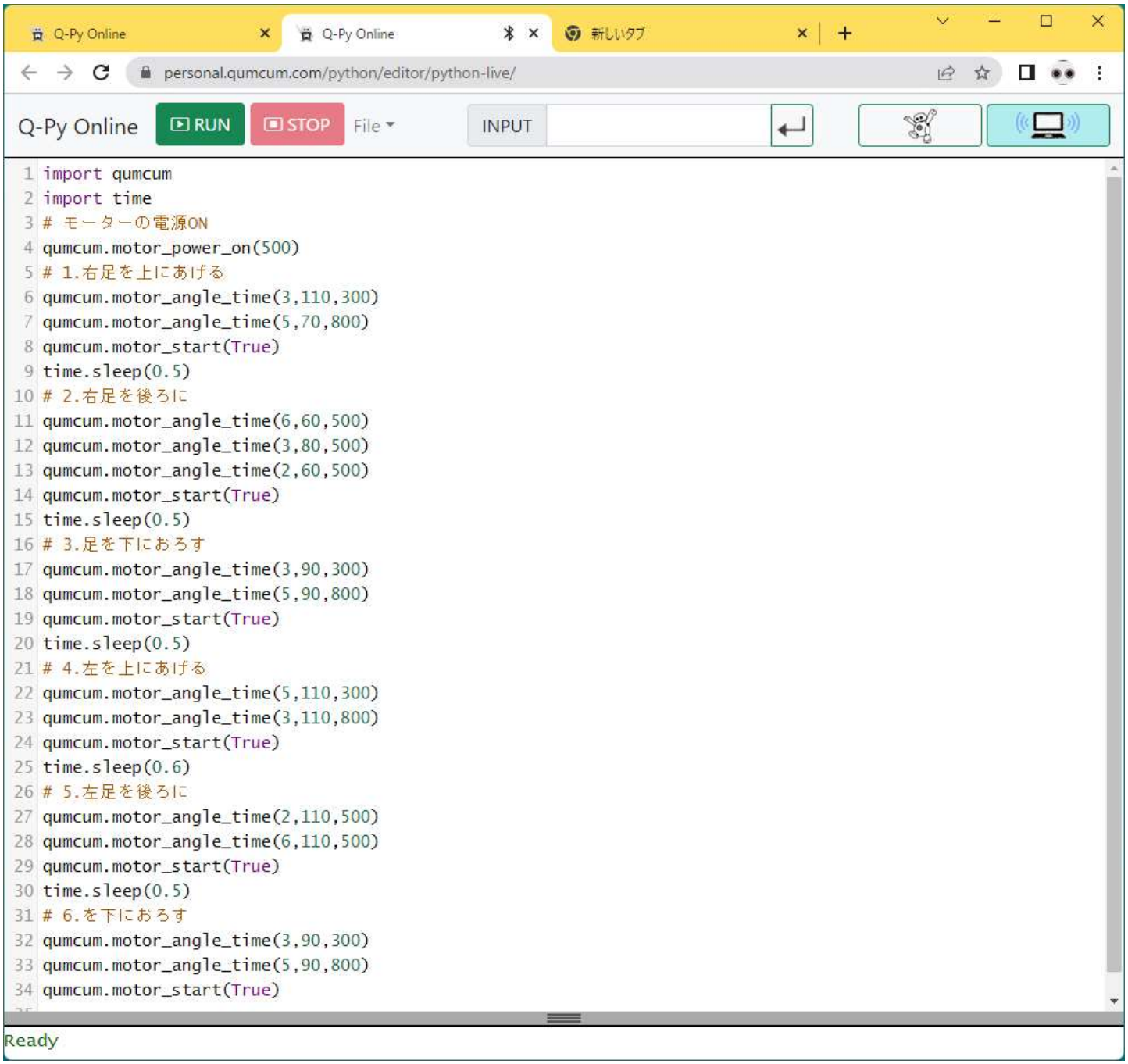

```python
# ランダムに動くクムクム
import qumcum
import time
import random

# お喋り言葉の配列
kotoba=["arayo","korayo","dokkoisyo","horehore","nanndayo"]

# プログラムスタート
qumcum.motor_power_on(1000)
# 無限ループ開始
while True:
    # 乱数セット
    hidari_ude=random.randint(0,180)
    migi_ude=random.randint(0,180)
    kao=random.randint(0,180)
    koe=random.randint(0,4)
    freq=random.randint(220,3000)
    led=random.randint(1,3)
    # 動き
    qumcum.motor_set_pos(7,hidari_ude)
    qumcum.motor_set_pos(1,migi_ude)
    qumcum.motor_set_pos(4,kao)
    qumcum.voice(kotoba[koe])
    qumcum.sound(freq,300)
    qumcum.led_on(led)
    # モーター回転
    qumcum.motor_start()
    # ちょっとまつ
    qumcum.wait(0.9)
    # LED消す
    qumcum.led_off(led)
```

#が頭についている日本語文字は“コメント”と呼び、プログラムの動きには影響しないメモです。

#のあとに一つだけ半角スペースを入れているのもPythonのお作法です。

テクニック2

後ろ歩きをするクムクムを作ってみます。

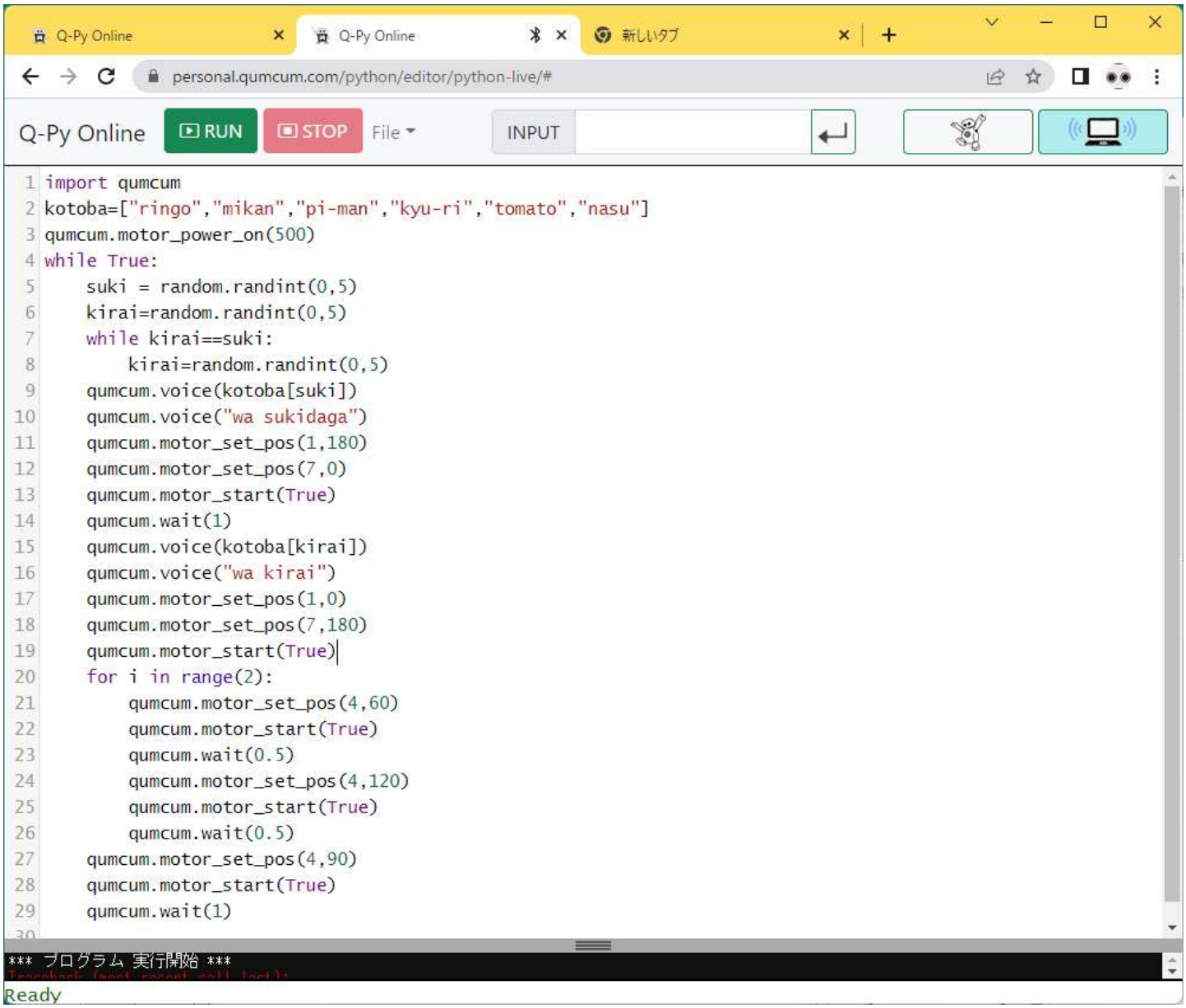



```
import qumcum
import time
# モーターの電源ON
qumcum.motor_power_on(500)
# 1.右足を上にあげる
qumcum.motor_angle_time(3,110,300)
qumcum.motor_angle_time(5,70,800)
qumcum.motor_start(True)
time.sleep(0.5)
# 2.右足を後ろに
qumcum.motor_angle_time(6,60,500)
qumcum.motor_angle_time(3,80,500)
qumcum.motor_angle_time(2,60,500)
qumcum.motor_start(True)
time.sleep(0.5)
# 3.足を下におろす
qumcum.motor_angle_time(3,90,300)
qumcum.motor_angle_time(5,90,800)
qumcum.motor_start(True)
time.sleep(0.5)
# 4.左を上にあげる
qumcum.motor_angle_time(5,110,300)
qumcum.motor_angle_time(3,110,800)
qumcum.motor_start(True)
time.sleep(0.6)
# 5.左足を後ろに
qumcum.motor_angle_time(2,110,500)
qumcum.motor_angle_time(6,110,500)
qumcum.motor_start(True)
time.sleep(0.5)
# 6.を下におろす
qumcum.motor_angle_time(3,90,300)
qumcum.motor_angle_time(5,90,800)
qumcum.motor_start(True)
```

ひとつひとつのモーターの速度を変えて細かく角度を変えています。
モーターは電源をONにするとすべてがカチっと動かない状態になるので、片方の足の角度だけを変えてももういっぽうが固まって踏ん張っているので傾きません。
もう一方の足も傾きをサポートするように同じように傾かせていくことで体が倒れて片足立ちができるようになります。
次のアクションとの間の time.sleep( )の中の数字を微調整すると少し早く歩いたり遅くあるいたりします。

テクニック3

すきな食べ物と嫌いな食べ物でアクションをします。



配列・無限ループの中の無限ループや繰り返し、乱数などちょっと複雑なプログラムです。
好きな番号を乱数で取得し、次に嫌いな番号をもう一度乱数で取得しますが、嫌いな番号が好きな番号と同じなら違う番号になるまで乱数を無限ループで発生させます。
好きな番号と嫌いな番号ができたら、配列から食べ物のことばで喋りアクションを付けます。