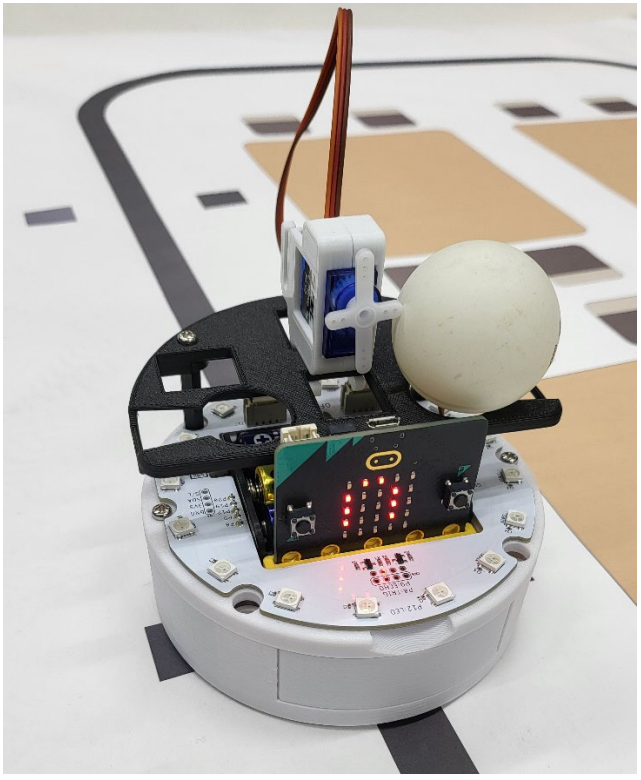
配膳ロボプリント 2023

（基本編）



注意事項

このプリントはなくさずに，毎回持ってくること！

＿＿＿＿組＿＿＿＿番　氏名＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

はじめに

このプログラムは、70 行ほどから成り立ちます。大きいプログラムになると、上から順番に書いて、実行していくことはできないので、作成順序を分割し、動作行為を小目標としてクリアしていく方法を取ろうと思います。また，今回から配膳ロボット用のプログラム haizen.py というファイルも読み込み，これを使ってプログラムを書いていきます。

基本編作成順序

作成順序は以下の通りです。

STEP1. コードはコメントアウトを使い行為を分解しておく。
STEP2. モジュールを読み込み，まっすぐ走る。
STEP3. 黒い線まで走っていき，黒い線になったらとまる。
STEP4. STEP3 のプログラムをコメントアウトし，ライントレースする関数をかいて，ライントレースする。
STEP5. テーブルをカウントしていくプログラムを書く。
STEP6. テーブル番号のセットをするプログラムを書く。
STEP7. もし，配膳テーブルと一緒なら，とまり，配膳する。(完成)
STEP8. (発展編) 動き始めたら，青く光り，配膳後は赤く光り，キッチンについたら消える。
STEP9. (発展編) 動き始めたら，青く点滅し，配膳後は赤く点滅し，キッチンについたら消える。
STEP10. (発展編) キッチンについたら，時計回りに青い光が移動し，青い光を一周させる。
STEP11. (発展編) 動き始めたら，時計回りに青い光が移動し，配膳後は反時計回りに赤い光が移動し，キッチンについたら消える。

注意事項

- **授業が終わるタイミングでいままで書いたプログラムをダウンロードしておく！**
- **コメント以外はすべて半角で入力！**
- **スペースは全角スペースを使ってはいけない！**
- **字下げする場合、tab 一つ分か、半角スペース 4 つ分を開ける！**

  ※コロン(：)をうって、エンターキーを押せば、自動的に字下げがされます。

【テスト対策部分】

1．関数

プログラムを書いていると，コードの一部をひとまとまりの関数として定義し，何度も呼び出したい場合があります。例えば，今回，配膳ロボットのプログラミングでは，黒線のラインをトレースしながら走るという行為は trace_line()という関数にして，あとはそれを使いまわしながら扱っていくコードを書いています。どのように定義しているのでしょうか。Python では def と書いて定義します。

呼び出される側	呼び出す側
#②ライントレースの関数 def trace_line(): 　sensor1 = pin1.read_analog() 　sensor2 = pin2.read_analog() 　th = 400　　#白と黒の閾値を定義する 　if sensor1 < th and sensor2 < th:　　#白　白ならまっすぐ走る 　　motor.set_speed(50,50) 　elif sensor1 >= th and sensor2 < th:　#黒　白なら左へ曲がる 　　motor.set_speed(0,50) 　elif sensor1 < th and sensor2 >= th:　#白　黒なら右へ曲がる 　　motor.set_speed(50,0) 　else:　　#黒　黒（ゴールライン）なら停まる 　　motor.stop() 　　reset()	#④メインプログラム while True: 　trace_line()

今回，trace_line()と定義しており，このようにプログラムを書くことでライントレースが行われます。()には何も入っていませんが，関数には，呼び出す側の()の中に数値なども入れることもでき，この数値を使って計算した結果を返すこともできます。例えば，以下のようなコードを書くと，三角形の面積のプログラムが実行されます。

コード	説明
def area(x ,y): 　z = x * y / 2 　return z a = 4 b = 5 c = area(a, b) display.scroll(c)	呼び出される側にも引数を定義する。 その引数を使って計算し，z の変数に計算結果を入れる。 その結果を z として返す。この値のことを（　　）という。 呼び出す側の a や b のことを（　　）という。

2．オブジェクト指向型プログラミング

これまで，○○○.△△△というプログラムを多く書いてきました。どういう意味なのか深く考えていきます。そもそもオブジェクト指向とは，「ある役割を持ったモノ」ごとにクラス（プログラム全体の設計図）を分割し、モノとモノとの関係性を定義していくことでシステムを作り上げようとするシステム構成の考え方のことです。モノというのは micro:bit でいうと，display,microphone, accelerometer が該当します。これらは○○○の部分に書かれることが多い単語でした。この部分のことを（　　）といい，名詞になっていることが多いのが特徴です。そして，それらをどのように動かすか指定した部分を（　　）といい，display オブジェクトで使えるメソッドは scroll や show などがあり，メソッド名は動きを定義することが多いので，動詞になることが多いのが特徴です。では，その scroll や show はどのように作られているでしょうか。それには設計図が必要です。設計図のことを（　　）といいます。micro:bit のクラスは python 以外の言語で複雑に書かれているので，今回は配膳ロボットで使うクラスを見てみることにしましょう。

```
# microbit-module: haizen@0.2.2
from microbit import *
"""
SWITCH EDUCATION
ロボットベースを配膳ロボットとして使うためのモジュール
find_marker の threshold を 600 に変更。
"""


class Lcd():
    """
    I2C 接続のキャラクター液晶を制御します。
    """
    def __init__(self):
        try:
            self.address = 0x3E
            i2c.init()
            self.lcd_init()
        except:
            pass
 (略)
class WheelMotor:
    """
    走行するためのモーター2 個を制御します。
    pin13,pin14,pin15,pin16 を使用します。
    """
    def __init__(self,lm_in1 = pin13, lm_in2 = pin15, rm_in1 = pin14, rm_in2 = pin16):
        self.lm_in1 = lm_in1
        self.lm_in2 = lm_in2
        self.rm_in1 = rm_in1
        self.rm_in2 = rm_in2

    def set_speed_l(self, speed_l):
        """
        左モーターの速さを-100～100 の範囲で指定します。
         0 にすると停止します。符号によって回転の向きが変わります。
        :param int speed_l :左モーター速さ
        """
        if speed_l < -100:
            speed_l = -100
        elif speed_l > 100:
            speed_l = 100
        if speed_l >= 0:
            self.lm_in1.write_digital(0)
            self.lm_in2.write_analog(int((1023 / 100) * speed_l))
        else:
```

```
            self.lm_in2.write_digital(0)
            self.lm_in1.write_analog(int((1023 / 100) * (-speed_l)))

    def set_speed_r(self, speed_r):
        """
        右モーターの速さを-100～100 の範囲で指定します。
        0 にすると停止します。符号によって回転の向きが変わります。
        :param int speed_r : 右モーターの速さ
        """
        if speed_r < -100:
            speed_r = -100
        elif speed_r > 100:
            speed_r = 100
        if speed_r >= 0:
            self.rm_in1.write_digital(0)
            self.rm_in2.write_analog(int((1023 / 100) * speed_r))
        else:
            self.rm_in2.write_digital(0)
            self.rm_in1.write_analog(int((1023 / 100) * (-speed_r)))

    def set_speed(self, speed_l, speed_r):
        """モーターの速さ設定
        左右のモーターの速さを-100～100 の範囲で指定します。
        0 にすると停止します。符号によって回転の向きが変わります。
        :param int speed_l: 左モーターの速さ
        :param int speed_r: 右モーターの速さ
        """
        self.set_speed_l(speed_l)
        self.set_speed_r(speed_r)

    def stop(self):
        """
        ブレーキをかけて停止します。
        """
        self.rm_in1.write_digital(1)
        self.rm_in2.write_digital(1)
        self.lm_in1.write_digital(1)
        self.lm_in2.write_digital(1)
(略)
```

3．なぜこのように分けるのか？

例えば，配膳ロボットで，stop()という関数を作ったとします。しかし，この stop は，何を stop するかを明確にすることはできません。motor なのか，LED なのか，LCD（液晶ディスプレイ）なのかがぱっと見分かりません。そこで，モノと動作行為はお互い連動しているように設計し，motor の中で stop()メソッドを作ったり，led の中で stop()メソッドを作れば，motor.stop()とか led.stop()というようにすれば分かりやすくなります。このようにするとプログラム全体の見通しがつき，わかりやすいプログラムになってきます。※関数は単独で存在できますが，メソッドはオブジェクトに属するものでクラス内で定義されるものです。

4．class（設計図）からどうやって使っていくのか？

class からオブジェクトを作っていくことを（　　）といいます。また，インスタンス化されたものは（　　）または（　　）といいます。例えば以下のコードを例にとります。

```
motor = WheelMotor()
```

「WheelMotor クラスを motor オブジェクト（インスタンス）としてインスタンス化する」といいます。

STEP１　コードはコメントアウトを使い行為を分解しておく。

プログラムの全体像は以下のようになっています。だいたいは上から順番に書いていきますが，上に戻ったり，下に飛んだりし，コードを記入するので，行番号がコロコロ変わります。そのため，コードの部分をある程度は分類わけし，その中でコーディングをする方法をとります。まずは，コメントアウト(#)を先頭に入力すれば，そのあとはプログラムとしては無効化されるので，まずは以下の分類わけを記述してください。

#①モジュールの読み込み，初期化，インスタンス化など #②ライントレースの関数 #③テーブル番号のセットプログラム #④メインプログラム

合格判定

上記を撮った写真を見せに来る。

STEP２　モジュールを読み込み，まっすぐ走る。

よりプログラミングが理解できるコラム　①オブジェクト指向プログラミングの考え方

処理の流れに沿ってプログラミングしていくことを手続き型プログラミングという。しかしこれでは、同じ行為を繰り返す際に、もう一度コードを書かなければいけなかったり、コードをひとまとまりにできなかったりと効率が悪い場合がある。そこで、関連するデータ（変数）とそれに対する操作（関数）をまとめて、一つのモノ（オブジェクト）として捉え、プログラミングしていくことをオブジェクト指向型プログラミングという。

haizen.py を読み込み，以下のコードを記述する。

コード部分	解説
#①モジュールの読み込み，初期化，インスタンス化など	
from microbit import *	配膳ロボを動かすためのコード集を読み込む
from haizen import *	音楽再生のためのモジュールを読み込む
motor = WheelMotor()	モジュールからインスタンス（オブジェクト）を作る。

解説：

まず初めに，今回のプログラミングに必要なファイルを呼び出し，モーターを動かす「モノ」を作ります。

コード部分	解説
#④メインプログラム	
while True:	ずっと繰り返す
motor.set_speed(50,50)	モーター（モノ）のスピードを左右 50,50（動作）にする。

ここで走らせに行く

合格判定

まっすぐ走ったら合格。少しずれる場合，50,50 を微調整してもよい。

※調整方法

右に曲がっていく場合，左側を小さくし，右側を大きくする。例：motor.set_speed(48,52)

左に曲がっていく場合，左側を大きくし，右側を小さくする。例：motor.set_speed(52.48)

STEP３　黒い線まで走っていき，黒い線になったらとまる。

コード部分	解説
#④メインプログラム **while True:** 　　**motor.set_speed(50,50)** 　　sensor1 = pin1.read_analog() 　　sensor2 = pin2.read_analog() 　　if sensor1 >=400 and sensor2 >= 400: 　　　　motor.set_speed(0,0) 　　　　break	 黒いラインを挟む二つのセンサーの値を読み取る。 もし，sensor1 と sensor2 両方とも 400 以上なら， モータをとめ， 繰り返しを抜ける。

配膳ロボットの電源は，コースにいってから入れること。

合格判定

黒い線までいったらとまる。距離をかえて二回試すこと。

STEP4　STEP3 のプログラムをコメントアウトし，ライントレースする関数をかいて，ライントレースする。

コード部分	解説
#②ライントレースの関数	
def trace_line():	ライントレース関数の定義
sensor1 = pin1.read_analog()	左側のセンサーのアナログ値を取得
sensor2 = pin2.read_analog()	右側のセンサーのアナログ値を取得
th = 400　　#白と黒の閾値	白と黒の境目の値（閾値）を初期設定
if sensor1 < th and sensor2 < th:　　#白　白	両方とも白であれば
motor.set_speed(50,50)	まっすぐ進む
elif sensor1 >= th and sensor2 < th:　　#黒　白	左側が白で，右側が黒なら
motor.set_speed(0,50)	左に曲がる。
elif sensor1 < th and sensor2 >= th:　　#白　黒	左側が黒で，右側が白なら
motor.set_speed(50,0)	右に曲がる。
else:　　#黒　黒（ゴールライン）	それ以外（両方とも黒）なら
motor.stop()	とまって，
reset()	リセットする。

※if 文の th（閾値）のところをすべて 400 と書いてもいいが，これを変えたい場合，６か所変えないといけないので，th という変数を作っておいて閾値を初期化しておく。

コード部分	解説
#④メインプログラム	
while True:	
# motor.set_speed(50,50)	STEP3 のものはコメントアウトしておく。
# sensor1 = pin1.read_analog()	
# sensor2 = pin2.read_analog()	
# if sensor1 >=400 and sensor2 >= 400:	
#　　motor.set_speed(0,0)	
#　　break	ここまでコメントアウト
trace_line()	ライントレース関数を記述する。

配膳ロボットの電源は，コースにいってから入れること。

合格判定

コースを一周し，キッチンで止まる。

STEP5　テーブルをカウントしていくプログラムを書く。

コード部分	解説
#①モジュールの読み込み，初期化，インスタンス化など# from microbit import * from haizen import * import music motor = WheelMotor() table_counter = 0 display.show(table_counter)	 ここも忘れずに書くこと！ テーブルカウンターを初期化する。

コード部分	解説

#④メインプログラム
while True:
　# motor.set_speed(50,50)
　# sensor1 = pin1.read_analog()
　# sensor2 = pin2.read_analog()
　# if sensor1 >=400 and sensor2 >= 400:
　#　　motor.set_speed(0,0)
　#　　break
　trace_line()
　if find_marker() > 50:
　　music.play(music.BA_DING, pin=None, wait=False)
　　table_counter = table_counter + 1
　　display.show(table_counter, wait = False)
	
	その長さが 50 以上なら（ここは調整の可能性あり）
	音楽を鳴らして，
	テーブルのカウンターを 1 増やして
	ディスプレイにずっと表示。

合格判定

スタートから始まり，テーブルマーカーごとにカウントがあがっていけば合格

STEP6　テーブル番号のセットをするプログラムを書く。

コード部分	解説
#①モジュールの読み込み，初期化，インスタンス化など# from microbit import * from haizen import * import music motor = WheelMotor() table_counter = 0 display.show(table_counter) target_table = 0 max_table = 4	 配膳先のテーブルを初期化 テーブル数を設定

コード部分	解説

#③テーブル番号のセットプログラム
display.show(target_table)
 	
while True:
　if button_a.was_pressed():
　　target_table += 1
　　if target_table > max_table:
　　　target_table = 1
　　display.show(target_table)
　if button_b.was_pressed():
　　if target_table == 0:
　　　display.show(Image.NO)
　　else:
　　　display.show(Image.YES)
　　　sleep(1000)
　　　break
	
	配膳先のテーブルを表示。初期値は 0
	
	a ボタンがおされたら，
	配膳先のテーブルを 1 プラスし，
	もし，配膳先のテーブルがテーブル数以上なら，
	1 に戻す。
	配膳先のテーブルを表示
	b ボタンがおされたら，
	もし，配膳先のテーブルが 0 のままだったら（a が押されていなければ）
	×と表示させ，
	そうでなければ，
	✓
	と 1 秒表示させ，
	繰り返しから抜ける。

合格判定

b ボタンを押したら，×になり，a ボタンを押したら，カウントが増えていき，4 の次は 1 に戻り， b ボタンを押したら，チェックマークが 1 秒現れれば OK。

STEP7　もし，配膳テーブルと一緒なら，とまり，配膳する。

コード部分	解説
#①モジュールの読み込み，初期化，インスタンス化など# from microbit import * from haizen import * import music motor = WheelMotor() table_count = 0 display.show(table_counter) target_table = 0 max_table = 4 serving = True servo = Servo180(pin8) servo.set_angle(180)	配膳中フラグを真にする。 サーボ 180 度モーターは pin8 に刺している 最初の角度を 180 度にする。

コード部分	解説

while True:
　# motor.set_speed(50,50)
　# sensor1 = pin1.read_analog()
　# sensor2 = pin2.read_analog()
　# if sensor1 >=400 and sensor2 >= 400:
　#　　motor.set_speed(0,0)
　#　　break
　trace_line()
　if find_marker() > 50:
　　music.play(music.BA_DING, pin=None, wait=False)
　　table_counter = table_counter + 1
　　display.show(table_counter, wait = False)
　　if table_counter == target_table:
　　　motor.stop()
　　　sleep(2000)
　　　servo.set_angle(0)
　　　sleep(2000)
　　　servo.set_angle(180)
　　　sleep(2000)
　　　serving = False
	もし，テーブルのカウントと，配膳先のテーブルが同じなら，とまって
	2 秒停止し，
	サーボモータを 0 度にして，
	2 秒停止し，
	再びサーボモータを 180 度にして
	2 秒停止し，
	配膳フラグを偽にする。

合格判定

指定されたテーブルに配膳できたら OK。

サーボモータは以下のように使ってください。

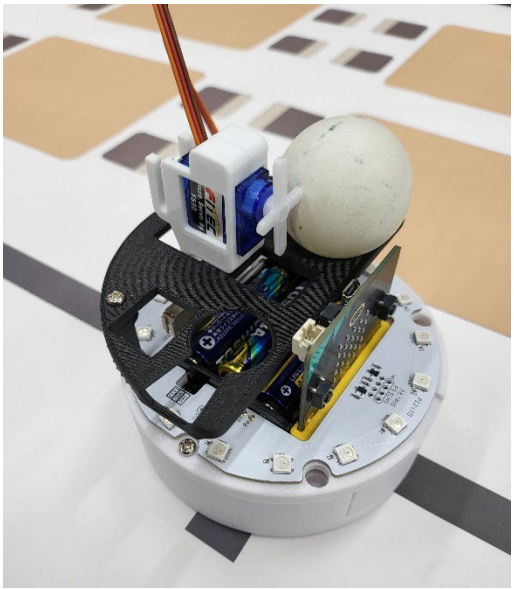
モータ

※十字のパーツを差し込みます

差す位置

※左の外側(pin8)に差してください。茶色い線が進行方向側です。

以下のようにサーボモータをつけ，ピンポン玉をおいてください。



クリアできたら

クリアおめでとうございます！

下の save ボタンをおし， complete というファイル名で保存し，ダウンロードフォルダにダウンロードされるので，自分のフォルダ（R:）に移動しておきましょう。

STEP8　（発展編）動き始めたら，青く光り，配膳後は赤く光り，キッチンについたら消える。

まず，円形の LED を装着します。
次に以下のコードを追加します。

コード部分	解説
#①モジュールの読み込み．初期化　インスタンス化など ***from microbit import ＊*** ***from haizen import ＊*** ***import music*** import neopixel np = neopixel.NeoPixel(pin12,16) #（以下略)	 neopixcel を使うことを宣言 neopixcel を pin12 にさして，ピクセル数が 16 であることを宣言

#④メインプログラム　の中で

np.fill((0, 0, 100))
np.show()

とすれば，LED を全部青に光らせることができます。かっこの中は第一引数（一番左）は赤，第二引数（真ん中）は緑，第三引数（一番右）は青でそれぞれの光の強さを表しています。

消す場合は,
np.fill((0,0,0))
np.show()
あるいは
np.clear()

とすれば消すこともできます。これ適切な場所に追加していきましょう。

ヒント 1：serving 変数に関して，初期化で True にしておいて，配膳が終わったら False にしています。この変数を使って挙動を考えます。
ヒント 2：if serving == True:　または if serving:
とどこかに書き，配膳中は青，配膳終わったら，赤にしましょう。

STEP9（発展編）動き始めたら，青く点滅し，配膳後は赤く点滅し，キッチンについたら消える。

配膳ロボットでは，動いている最中は sleep 関数は使うことができません。（もちろん，とまっている最中は sleep 関数が使えます。）そこで，動いている最中に，挙動を変えていきたい場合，割り込みプログラムを書いていきます。

コード部分	解説
#①モジュールの読み込み，初期化，インスタンス化など	
record_time = 0	記録時間
TIME_INTERVAL = 500	何ミリ秒間隔で変えていきたいか
motion_number = 0	動作番号を 0 に初期設定

コード部分	解説
#④メインプログラム	trace_line()関数のすぐ下に割り込みプログラムを書いていく。
while True:	
# motor.set_speed(50,50)	
# (略)	
# break	
trace_line()	
if running_time() - record_time > TIME_INTERVAL:	動いている時間から記録時間をひき，それが TIME インターバル以上であれば，記録時間を動いている時間に上書き。
record_time = running_time()	
if motion_number == 0:	もし，動作番号が 0 番なら
#ここにしたい処理	したい処理を記述し
motion_number = 1	動作番号を 1 番にしておく。
elif motion_number == 1:	もし，動作番号が 1 なら
#ここにしたい処理	したい処理を記述し，
motion_number = 0	動作番号を 0 番にしておく。

このようなプログラムを書くことで，２つの動作を交互に実行させるようなプログラムが作れます。点灯と消灯のプログラムを「#ここにしたい処理」の中に記述すれば，課題を解くことができます。このようにすることで 500 ミリ秒間隔で動作番号 0 の行為と動作番号 1 の行為を交互に実行させることができます。

STEP10（発展編）キッチンについたら，時計回りに青い光が移動し，青い光を一周させる。

一つの LED だけ光らせる場合は配列（リスト）を用います。例えば，0 番目を光らせたいのであれば,

```
np[0] = (0,0,100)
np.show()
```

とすれば，0 番の LED のみ光らせることができます。
他にも

```
i=1
np[i]=(0,0,100)
np.show()
```

とすれば，1 番の LED を光らせることができます。
for 文や while 文を使ってぐるっと光がまわるプログラムを作りましょう。

STEP11（発展編）動き始めたら，時計回りに青い光が移動しつづけ，配膳後は反時計回りに赤い光が移動しつづけ，キッチンについたら消える。

最終問題です。これはノーヒントでお願いします。

全体完成プログラム

	コード	
1.	#①モジュールの読み込み，初期化，インスタンス化など	
2.	from microbit import *	
3.	from haizen import *	
4.	import music	
5.		
6.	motor = WheelMotor()	
7.		
8.	table_counter = 0	
9.	display.show(table_counter)	
10.	target_table = 0	
11.	max_table = 4	
12.		
13.	serving = True	
14.	servo = Servo180(pin8)	
15.	servo.set_angle(180)	
16.		
17.	#②ライントレースの関数	
18.	def trace_line():	
19.	sensor1 = pin1.read_analog()	
20.	sensor2 = pin2.read_analog()	
21.	th = 400 #白と黒の閾値	
22.		
23.	if sensor1 < th and sensor2 < th: #白　白	
24.	motor.set_speed(50,50)	
25.	elif sensor1 >= th and sensor2 < th: #黒　白	
26.	motor.set_speed(0,50)	
27.	elif sensor1 < th and sensor2 >= th: #白　黒	
28.	motor.set_speed(50,0)	
29.	else: #黒　黒（ゴールライン）	
30.	motor.stop()	
31.	reset()	
32.		
33.	#③テーブル番号のセットプログラム	
34.	display.show(target_table)	
35.		
36.	while True:	
37.	if button_a.was_pressed():	
38.	target_table += 1	

```
            if target_table > max_table:
                target_table = 1
            display.show(target_table)
        if button_b.was_pressed():
            if target_table == 0:
                display.show(Image.NO)
            else:
                display.show(Image.YES)
                sleep(1000)
                break

#④メインプログラム
while True:
    # motor.set_speed(50,50)
    # sensor1 = pin1.read_analog()
    # sensor2 = pin2.read_analog()
    # if sensor1 >=400 and sensor2 >= 400:
    #     motor.set_speed(0,0)
    #     break
    trace_line()
    if find_marker() > 50:
        music.play(music.BA_DING, pin=None, wait=False)
        table_counter = table_counter + 1
        display.show(table_counter, wait = False)
        if table_counter == target_table:
            motor.stop()
            sleep(2000)
            servo.set_angle(0)
            sleep(2000)
            servo.set_angle(180)
            sleep(2000)
            serving = False
```

メモ：