Markdownテストかねて https://www.whizz-tech.co.jp/6026/#Markdown

EV3 MicroPython のエッセンス BALUS_Dagron 2021/8/7

～visual programming から text な言語へ～

もくじ

1. Introduction
2. 前提知識 ～おおまかなプログラムの構成～
3. 基本的なオブジェクトの作成 ～モータとセンサの定義～
4. インテリジェントブロックとモータの動かし方
5. 制御フローとセンサの使い方
6. 変数の定義
7. 公式リファレンスの読み方
8. リンク集

1: Introduction ～ For whom?

このマニュアル？リファレンス？は，普段 EV3 software を使ってプログラムを普通に書いている，小学校5,6年生から特に中学生が，公式リファレンスを読む前に読むためのリファレンスです．基本的な設定方法と，リファレンスの読み方についてざっくり書いてます．「プログラムを普通に書いている」は，EV3 software でスイッチ/ループあたりを不自由なく使えてるくらいのイメージです．また，そこからさらに変数やマイブロックの使い方を把握できていればなおヨシ．あとはさらっと英語を扱えるとうれしい．公式の reference は最新ですが，英語しかないので．（まあgoogle翻訳とかっていう文明の利器があるけども，たぶん翻訳使わなくてもわかるようになると思います．）

ふだんかいているブロックを並べていくスタイルの visual なプログラミング言語はもう見慣れたものだけど，文字ばっかりのプログラムは初めてやる，よくわからん，本当にできるの？と思う人も多いと思います．**大丈夫です，必ずできます！** 特にvisualな言語では難なく書ける，つまりプログラミングに必要な思考方法と，プログラムそのものの組み立て方（ライントレースはスイッチ使ってセンサの値に応じて右行くか左行くかやってループさせれば良いよね，みたいな）が分かっているのなら，それがそのまま活きてきます．その思考をどう表現するかを，このマニュアルで軽く紹介します．このマニュアルを一通り読めば，一通りのプログラムが書けると思いますし，公式のリファレンスを読んだときにその内容を良く理解できるようになると思います．

なおこのマニュアルは私，大山の完全主観で書いています．「一応この解釈で動かせば動く」「わからんけどこう考えれば辻褄が合うし，たぶんこうでしょ」みたいなほんわりとした認識で書いている部分が多数存在（全部）するので，つっこみがあればお願いします．

また MicroPython そのものの導入方法や使い方， microSD カードの焼き方等は，アフレルさんから出ている日本語版のリファレンスが存在しているので割愛します．

2: 前提知識 ～おおまかなプログラムの構成～

実際に書き始める前に，プログラムの概要について把握しておきましょう．プログラムそのものは大まかにわけて3つのブロックに分かれています．はじめにプログラムそのものを動かすための呪文，次にロボットの構成と変数/関数の定義，そして最後にプログラムそのものが書いてあります．（ライブラリのインポート，オブジェクトと関数の定義，プログラム本体）

さっそく見てみましょう，新規プロジェクトを作成すると，その中にmain.pyというものが入ってると思います．ev3からビープ音をならすだけの，テスト用プログラムみたいな感じですね．

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile


# This program requires LEGO EV3 MicroPython v2.0 or higher.
# Click "Open user guide" on the EV3 extension tab for more information.

# Create your objects here.
ev3 = EV3Brick()

# Write your program here.
ev3.speaker.beep()
```

…はい．初っぱなから謎なものが出てきたと思うかもしれませんが，私もよく分かってません．分かってなくても書けるので問題ないです．ここの特に初めの方にあるものたちは考えたら負けです．呪文ですね．ここも一部触らないと行けない場合があるので，まあそのときに説明します．

まずは魔法の呪文の場所について，ここを触ることはあまりないです．NXTのセンサを使いたいときに触るかもしれませんが，後述します．

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

# This program requires LEGO EV3 MicroPython v2.0 or higher.
# Click "Open user guide" on the EV3 extension tab for more information.
```

ここにはプログラムを動かすために必要な魔法の呪文たちがつらつらと書かれています．（ライブラリのインポート）ここ消すと動かなくなるので，間違えて消した場合は新しいプロジェクトを作って移植しましょ

う．

下の2行はコメントですね．プログラムを書く上での注意事項みたいなことが書いてありますね．

python では，文頭に # をつけることで，プログラム中にメモを残すことができます．文字で書くプログラムは全体を俯瞰しにくいのはたしかなので，プログラムを書いたときに何を考えていたのか，どういう理屈で動かすのか，その変数の意味は何なのかなど，随時メモをしていくことを推奨します．(でないと，深夜テンションで書いたプログラムを翌日見たときに「なんもわからん…」ってなります．)

なお，呪文セクションの一番はじめにも#で始まる行がありますが，あれはコメントではなく呪文の一部なので，触らないようにしましょう．動かなくなります．

```
# Create your objects here. オブジェクトをここで作成
ev3 = EV3Brick()
```

ここが私たちが書かなきゃ行けない一番はじめのプログラムです．ここでロボットに接続されているモータやセンサの定義，変数の定義，関数（function，マイブロック）の定義をします．

ここでは，EV3Brick（インテリジェントブロック）を，"ev3" と呼称することを定義しています．また，定義したこれをオブジェクトと呼びます．

```
# Write your program here. プログラムをここに書く
ev3.speaker.beep()
```

ここがプログラム本体です．実際の動作等をここに書きます．

ここでは，「先程定義した "ev3" 上のスピーカーで，ビープ音を鳴らす」を指定しています．

以上がプログラム全体の構成です．個々について見ていきましょう．なお，私達が書くのはブロックの定義とプログラムそのもののセクションだけなので，基本的にはそこのみについて書きます．呪文は基本すべて共通なので，省きます．

3: 基本的なオブジェクトの作成 ～モータとセンサの定義～

ここでは，インテリジェントブロック本体につないだモータやセンサが，どのようなものなのかをオブジェクトという形で定義します．内容は，公式 reference の ev3 devuces 等の項目に則っているので，できれば並べながら参照してください．

基本的な構成は以下の通り．たぶん実例みてくとわかると思います．

```
(任意の名前) = (クラス:そのモータ/センサの定義)
```

モータやセンサをオブジェクトとして命名して．それがどんなものなのか（ポート，モータ極性など）をクラスで定義します．名前は自由につけてください．下で私が書いているのもあくまで一例です．

1: インテリジェントブロックの定義

これはサンプルプログラムにもありましたね．

```
ev3 = EV3Brick()
```

です． ev3 のところは任意なので別に変えてもらっても構わないですが，ここはサンプルでも ev3 なので，そのままを推奨します．インテリジェントブロックを2つ使うこととかそうそうないですからね．

最後にカッコがついてますが，これは必要です．モータ等の定義ではここに引数（ひきすう）が入ります．説明みたいな感じ．

2: モータの定義

```
R_motor = Motor(Port.C, Direction.CLOCKWISE)
L_motor = Motor(Port.B, Direction.CLOCKWISE)
arm = Motor(Port.A, Direction.COWNTERCLOCKWISE)
```

たぶん見ての通りかな．

```
(任意の名前) = Motor(ポートの指定, 回転方向の指定)
```

モータの説明をするクラスのところで，引数としてポート番号とモータの回転方向を指定します．

ポートの指定は以下の通り．

- Port.A
- Port.B
- Port.C
- Port.D

なお，センサ側は

- Port.S1
- Port.S2
- Port.S3
- Port.S4

回転方向については，

- Direction.CLOCKWISE
- Direction.COWNTERCLOCKWISE

で，どちらを正転方向にするか指定します．

ちなみに，Mモータも NXT のモーターも，EV3 Lモータとおなじプログラムで動かせます．また，実はもう一つ引数を増やせますが，気になる人は公式 reference を読んでみてください．使いどころさえつかめば便利そうなので．

3: センサの定義

EV3 カラーセンサ

```
colour_sens = ColorSensor(Port.S3)
```

カラーセンサを光センサとして使うつもりでも，これを書きます．定義セクションではあくまでも刺さってるセンサを定義するだけです．カラーか光かどちらのモードで動かすかは，プログラム中でその都度指定します．後述します．

EV3 超音波センサ

```
range_sens = UltrasonicSensor(Port.S4)
```

NXT センサについて

NXT センサについても基本的な定義方法は同じです．しかし，ev3 micropython にはデフォルトでは NXT のセンサ類を動かすためのプログラムが入っていません．そのため，NXT のセンサを使うためには，これを import してあげる必要があります． …まあ，プログラム一番はじめの魔法の呪文のところに，以下の文言を書き足すだけですけどね．

```
#ライトセンサを使いたい場合
from pybricks.nxtdevices import (LightSensor)
```

とか

```
#とにかく全部使いたいとき
from pybricks.nxtdevices import (TouchSensor, LightSensor, ColorSensor,
UltroasonicSensor, SoundSensor)
```

とか．使いたいセンサの分だけ import: 読み込ませてあげればいいです．足りないとエラー吐きますが，余分な分には問題ないので．

これらを魔法の呪文セクション（ライブラリのインポート）に追加してから，定義セクションで

```
light_sens = LightSensor(Port.S1)
```

とかって定義してあげましょう.

4: インテリジェントブロックとモータの動かし方

とりあえず実際に動かしてみましょう．オブジェクトとして以下を定義している前提で書きます.

```
ev3 = EV3Brick()
R_motor = Motor(Port.C,Direction.CLOCKWISE)
L_motor = Motor(Port.B,Direction.CLOCKWISE)
```

インテリジェントブロックを ev3 と呼称，左右のモータをそれぞれ R_motor, L_motor と呼称してますね.

1: インテリジェントブロックの制御

1: スピーカの制御

いちばん初めのビープ音を鳴らすプログラムがありましたね.

```
# Write your program here.
ev3.speaker.beep()
```

ev3 上のスピーカーから，ビープ音を鳴らす，というそのまんまです.

ほかには

```
ev3.speaker.say("Hello World")
```

スピーカーから，任意の単語や文章を読み上げさせたりすることができます．けっこうしっかり発音してくれるので面白いです.

2: ディスプレイの制御

スクリーン上に任意の文字列を表示させます.

```
#スクリーンのリセット
ev3.screen.clear()
#スクリーンに文字を出力
ev3.screen.print("Hello World")
ev3.screen.print("LEGO EV3")
```

screen.print を分けて実行することで，改行した状態で表示させることができます.

出力結果は以下みたいなかんじ

```
Hello World
LEGO EV3
```

2: モータの制御

```
L_motor.run(360)
```

EV3 Software でいうところの，モータON の状態です．次にそのモータに対して指示が来るまで回り続けます．後ろの数字は回転速度で，[deg/s] で指定します．今回だと360度毎秒ですね．普段ならパワーで指定してるとこです．

```
L_motor.run_angle(360, 180)
```

モータの回転角を角度で指定します．引数はそれぞれ

```
L_motor.run_angle(回転速度[deg/s], 回転角[deg])
```

です．

逆に止めたいときは

```
R_motor.stop() #モータ止めて惰性で回転
R_motor.brake() #ブレーキ
R_motor.hold() #hold したときのモータの角度でピタッと止める，超えたら戻る
```

の3種類があったりします．

3: DriveBase の使い方

上のようなモータの制御方法では，複数のモータを同時に動かすことができません．python では一応並列処理をする方法もあるのですが，ちょっとハードルが高すぎるのでここではクラス DriveBase を使います．

定義セクションで，ロボットを以下のように定義してあげます．

```
(任意の名前) = DriveBase(left_motor, right_motor, タイヤ直径[mm], ホイール間距離[mm])
```

具体的には

```

```

```
# Create your objects here. ここの3つは前提として
ev3 = EV3Brick()
R_motor = Motor(Port.C,Direction.CLOCKWISE)
L_motor = Motor(Port.B,Direction.CLOCKWISE)

#ここが今回の DriveBase のキモ
robot = DriveBase(L_motor, R_motor, 57, 180)
```

ここであれば，駆動部がタイヤ直径 57mm，ホイール間距離 180mm のロボットになります．

実際に動かすときは

```
robot(任意の名前).settings(直進スピード[mm/s]，直線加速度[mm/s^2]，回転速度[deg/s]，回転角速度[deg/s^2])
```

でロボットの動く速度の初期値を設定してあげてから，（ここでの回転速度はロボット自体が回転する速度です）

```
robot.straight(距離[mm])

robot.turn(角度[degree])
```

で指定した分だけ移動/回転させたり，

延々と動かすときは

```
robot.drive(移動速度[mm/s], 回転速度[deg/s])
```

で動き方を指定して

```
robot.stop()
```

で止めたりできます．

5: 制御フローとセンサの使い方

さて，ここまでの項でとりあえずロボットを支持通りに動かすだけのプログラムは書けるようになりました．が，もちろんこれだけでは物足りないとおもうので，もう一つ進めて見ましょう．ループとスイッチです．が，そのまえに python 中における数値の扱い，条件文の扱いについて説明します．ココらへんは普通のpythonとおなじ(ハズ)なので，詳しくやりたい方は一般のpythonの書籍を片手にやってもいいと思います．というかそっちを推奨します．おすすめ参考書は巻末に書いときます．(多分)

1: 変数・数値の扱い 演算

変数について，基本的な考え方はこれまでと同じです．名前をつけた変数の入れ物に対して，値や真偽，文字列をいれて保管をすることができます．

```
hensu = 3               #int    整数型
i = 0
PI = 3.14159            #float  浮動小数点型 負の数もここに含まれる
k = -3.6
check = True            #bool   ブール，真偽 True/False で定義
words = "Hello World"   #str, string 文字列 "ダブルクォーテーション”で囲む
list1 = [0, 1, 2, 3]   #list   配列・カンマを使って要素を区切る.
#list の中の要素は型がバラバラでもok 一番はじめの要素は matrix[0]で呼び出す.  2個めなら
matrix[1]
matrix = [[1, 2, 3], [4, 5, 6]]
#listの応用 listの中にlistを入れて，行列のようにすることもできる.
```

EV3 Software では変数の定義をするときに，その変数が数値なのか，真偽なのか，文字列なのかなどの”型"を指定する必要がありました．pythonでは，変数を宣言する際に，型について言及する必要はありません！！ ただし，型そのものは存在しているので，時々気にしなければ行けないときが出てくるので注意です．なお，型の種類はここに上げたもの以外にもいくつか存在するので，

ちなみに，定数は存在しないので，プログラム中で間違えて定数にしたいものに新しく数値を書き込まないように注意が必要です．変数の名前を大文字にして定数であることをアピールしたりします．（上で言う PI)

また，数値の足し算等も可能です．

```
a = 1 + 2
b = a + 5
c = a / (12 + 5)
d = d + 1   #もとの d に 1 を足して d に代入し直す.
```

一番左にある変数に，イコール右側の計算結果を代入します．

計算で使える演算子で，代表的なものを以下に上げます．

2: センサの使い方

3: 条件式 比較演算 論理演算

4: while文 ループに相当

```
while 条件式:
    (条件式が True である間続けてほしいこと)
```

これで，条件が True となっている間は，while 内での処理が繰り返し実行されます．なお，while 内であることはインデント（文頭の空間）によって示されます．これは Tab キーで入力します．また，入れ子状にするときはインデントをどんどん増やしてきます．

たとえば

```
while True:
    L_motor.run(500)
    R_motor.run(500)
```

であれば，L,Rモータを 500[deg/s] で走らせ続けるというものですし，

```
while L_sens.reflection() > 50:
    L_motor.run(500)
    R_motor.run(250)
```

であれば，Lセンサの値が50以上の間,while中の処理が繰り返し実行されます．（センサの使い方は後述）

5: if文 スイッチに相当

```
if 条件式:
    (条件式が True であればやってほしいこと)
```

while文のときと同じように，インデントでif文の中であることを示します．

例えば

```
#cを定義
c=0
#センサの値が50以上ならcを1にする
if L_sens.reflection() > 50:
    c=1
```

また，条件を満たさなかった場合にしてほしいことを書くこともできます．

```
if 条件式:
    (条件式 True でしてほしいこと)
else :
    (条件式 False でしてほしいこと)
```

分岐先をたくさん作りたければ

```
if 条件式1:
    (条件式1が True でしてほしいこと)
else if 条件式2:
    (条件式2が True でしてほしいこと)
else if 条件式3:
    (条件式3が True でしてほしいこと)
else
    (全部 False でしてほしいことがあれば，ここに書く．なければ else もいらない)
```

6: 関数の定義

1: 変数

2: 関数

5: 基本的なプログラム