ダウンロード・データあります

第3章 PCで効率良くプロトタイプ開発したプログラムをマイコン（MicroPython）で本番実装するときの勘所

移植事例③… Pythonプログラム

宮田 賢一

● Pythonでプロトタイプ開発，MicroPythonで本番実装

MicroPythonとPythonは多くの部分で言語仕様に互換性があり，同じプログラムがPythonとMicroPythonでそのまま動作する可能性があります．そのため開発環境が充実しているPC上でプロタイピングを行い，本番実装の段階でマイコン・ボードに転送して最終的な調整を行うという，段階的開発を採用することができます．その際に，PythonとMicroPythonの違いを意識しつつ，なるべくプログラムの修正範囲を小さくするようなプログラミング・テクニックが求められます．

● 落とし穴もある

互換性が高いとはいえ，PythonとMicroPythonとの間には，特にMicroPythonの実行環境におけるリソース制限に起因する非互換性があり，気を付けていないと落とし穴にはまりがちです．本稿では，よくあるパターンを幾つか取り上げ，プログラミング・テクニックや注意点をコード例とともに紹介します．

技1 ハードウェアを抽象化

PythonからMicroPythonに移植することが決まっている場合は，最初からそれを見据えたプログラミングをしておきます．

ハードウェアや実行環境の違いを吸収するのによく使われる手段は，クラスによる抽象化です．コード例を使って抽象化の手法を説明します．

● 共通インターフェースを持つクラスを定義する

GPIOを介してLEDの点灯，消灯を制御する例を取り上げます．通常のPCには自由に制御できるLEDは搭載されていません．そこで次の実装方針を採ることにします．

- PythonではLED点灯，消灯の代わりにprint関数でメッセージを出力する
- MicroPythonでは実際にGPIOを制御してLEDの点灯，消灯を行う

これを実現するために，どちらの実行環境でも必要な共通のインターフェースを持つクラスを定義します．**リスト1**は4つのメソッド，value()，on()，off()，toggle()を持つクラスです．それぞれLEDの点灯状態の取得，点灯，消灯，トグルを意味します．つまりLEDDeviceクラスのオブジェクトに対しては，これらのメソッドを呼び出せるということです．**リスト1**はled_device.pyという名前で保存しておきます．

リスト1　共通インターフェースのクラス（led_device.py）

```
class LEDDevice:
    def value(self): pass
    def on(self): pass
    def off(self): pass
    def toggle(self):
        if self.value():
            self.off()
        else:
            self.on()
```

▶1，インターフェースは空の実装とする

リスト1で注目すべきなのは，メソッドvalue，on，offは実装が空（pass文のみ）であることです．つまりLEDDeviceクラスのオブジェクトを生成してこれらのメソッドを呼び出しても何も起こりません．具体的な実装はLEDDeviceクラスを継承した具象クラスで行うということです．

▶2，インターフェースだけを用いてプログラム可能

もう1つのポイントは，toggleメソッドの内容です．実はこのメソッドだけはメソッドの内容を実装しています．これが可能なのは，toggleの動作はインターフェースであるvalue，on，offだけを使って実装できるからです．つまりtoggleは現在の状態がTrueならLEDを消灯，FalseならLEDを点灯するというものであり，具体的な実装がprint文であれGPIO制御であれ，それとは独立に実装できるということです．