第4章 Cで実装するから高速動作可能

# C/C++でMy拡張モジュールを作る

常田 裕士

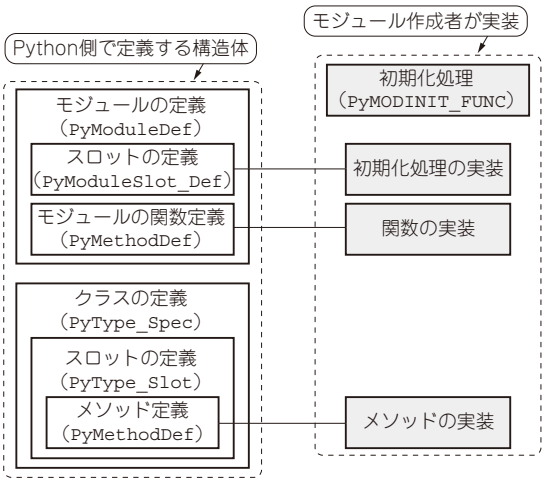


図1 モジュールの静的な構造

Pythonにはネイティブ・コードと連携するためのAPIであるPython/C APIがあります．これは，Pythonのネイティブ拡張機能を作ったり，Pythonのインタプリタをほかのプログラムから使ったり，Pythonの内部にアクセスするためのさまざまな機能を持っています．

Pythonを使った処理で，どうしても速度が要求を満たさない場合，上記の機能を使ってC/C++で記述したネイティブ・コードを呼び出し，高速な処理を実現できます．

実際には，Pythonにはさまざまな機能を持つモジュールが既に用意されており，高速な処理が必要な場合は，そういったモジュールを利用できるので，自身でC/C++を使ったネイティブ・コードによるプログラミングが必要な場面は，そう多くはないと思います．

しかし，C/C++で書かれた既存のモジュールを利用する場合でも，デバッグでC/C++部分のコードまで追跡する場合には，Pythonからネイティブ・コードを呼び出す仕組みを知っておく必要があります．

ここでは，この仕組みを利用して独自のクラスを定義し，ネイティブ拡張モジュールを作成する方法を紹介します．

Python/C APIの仕様や基本的な情報は，Python/C APIのリファレンス(1)を参照してください．

リスト1 ネイティブ拡張を作るためのスクリプト(setup.py)

```
from distutils.core import *

setup(name='MyModule',
      version='1.0',
      ext_modules=[Extension('mymodule',
                             ['mymodule.c'])],
     )
```

## ネイティブ拡張モジュールの全体構成

Pythonのネイティブ拡張モジュールの仕様に準拠させるには最低限，次の要素が必要になります．

- エントリ・ポイント(モジュールの初期化処理)
- モジュール初期化処理で使われるモジュール定義情報

これだけだとインポートできるだけなので，実際に使えるモジュールを作るためには，さらに以下のような要素も必要になります．

- モジュールの関数の実装
- モジュールの初期化処理
- クラスの定義の実装
- クラスのメソッドの実装

ネイティブ拡張モジュールはおおよそ，**図1**のようなデータで構成されています．以下で，これらの定義の仕方，使い方を見ていきます．

## 事前準備

ネイティブ拡張を作るために，setup.pyを作成します．setup.pyは，Pythonのモジュール作成機能のdistutilsで使われるファイルです．

ここではミニマムな構成として，**リスト1**の内容でsetup.pyのファイルを作成します．

次のコマンドの実行で，それぞれの環境に合わせて