第 4 回 Python モジュールゼミ

ゼミ担当者　：　小林祐介，森智弥
開催日　：　2009 年 5 月 12 日

ゼミ内容: 本ゼミでは，Python のモジュールについて学習する．

1　はじめに

モジュールとは，一連の関数やプログラムの実行文が記述された Python のスクリプトファイルのことである．モジュールを使うことで，一度書いた便利な関数を，その定義を再度コピーすることなく個々のプログラムで使うことができ，コードの再利用が容易になる．また，コードの共有も可能となる．

本資料では，Python のモジュールについて，その使い方や使用上の注意点について述べる．また，複数のモジュールを管理するために用いられる”パッケージ”や Python と同時にインストールされるモジュールである”標準ライブラリ”について説明する．以降，モジュール，モジュールの紹介，パッケージ，標準ライブラリの順に解説を行う．

2　モジュール

2.1　概要

Python において大規模なプログラムを作成する場合，他のオブジェクト指向型のプログラミング言語のように，プログラムを部品単位に分割して作成することが可能である．このような部品を作成する方法の 1 つに，モジュールがある．モジュールは，import や from というキーワードを利用し，パッケージの階層をドット (.) で区切る等，Java とよく似た方法で使用することができる．モジュールを使う事で，外部のプログラムファイルから簡単に関数を利用する事が可能となる．

2.2　使い方

2.2.1　import 文

プログラムでモジュールを使うには，import 文と使用する．モジュールには名前がついており，処理の内容や利用目的によって分類されている．import の後に，利用したいモジュールの名前を記述する．

```
import モジュール名
```

Python を使用して，数学的な関数を扱う場合，math というモジュール名のモジュールをインポートする必要がある．例として，math モジュールを使用し，三角形の面積を求める計算を行う．

test.py

```
>>> import math
>>> 10*20*math.sin(math.radians(45))/2
70.710678118654755
```

このように，まず，math のようなモジュール名を指定してインポートを行う．こうすることで新しいモジュールが定義され，使用する事が可能となる．インポートしたモジュールは変数のように扱う．この例に出てくる sin(), radian() は math モジュールに定義されている関数である．

2.2.2　from 文

import 文の代わりに from 文を用いてモジュールを読み込むことも可能である．モジュール名を記述せず関数や変数を直接利用したい時に from 文を使用すると便利である．先ほどの test.py は from 文を使用すると，以下のように書ける．

test.py

```
>>> from math import sin,radians
>>> 10*20*sin(radians(45))/2
70.710678118654755
```

from 文には，読み込みの対象となるモジュール名を指定する．その後，import を使って，モジュールの中に定義された関数を指定する．また，モジュールに定義されているすべての関数・変数を取り込みたい時は「＊」を利用する．

2.3　モジュールファイルの作成

Python では，スクリプトファイルとモジュールとなるファイルをほぼ同じに扱う．スクリプトファイルのファイル名のうち,「.py」などの拡張子の前までがモジュール名となる．モジュールとして利用するスクリプトファイルにファイル名を付ける時は，以下の点に注意が必要である．

- 数字から始まる，または拡張子の前にドットを含んだファイル名は付けない．
- 変数名や関数名として使われるような一般的な名前はなるべく使わない．

- 特別な必要がない限り，アルファベットの小文字だけを使う．

例えば,「00module.py」というようなスクリプトファイルを作ったとする．このファイルは，Python では実行することはできるが，モジュールとしてインポートする事ができない．また，同じ名前のモジュールが複数あると，プログラム実行時に問題が発生する．標準ライブラリのモジュールと同じファイル名は問題を起こすため，特別な必要がない限りは使わない方が良い．

2.4 モジュールをインポートする時の挙動

モジュールをインポートすると，Python はモジュール名に相当するファイルを読み込む．ファイルを読み込むとき，トップレベルのブロックに定義されている命令を実行する．モジュールのトップレベルのブロックに変数や関数が定義されていれば，新しい変数や関数を作る．このとき定義された変数や関数は，モジュールが持っている作業領域のような場所に一時的に定義される．

ファイル実行時にだけ実行するブロックというものが存在する．例えば，if 文自体はトップレベルのブロックにあっても，モジュールをインポートしても実行しない．例えば,「if __name__ ==’__main__’」という if 文のブロックがあるとすると，そのブロックはインポート時に実行されない．代わりに,「this is code block」のようなモジュールファイルを Python の引数として渡して直接実行すると，if 文のブロックを実行する．このような if 文には，モジュールの機能をテストするためのテストコードを記述することがある．モジュールが正しく動くかどうかを確認するときには，モジュールをインポートするのではなく，ファイルを直接実行する．

2.5 モジュールを利用する際の注意点

2.5.1 from 文を使ったインポートの弊害

例を挙げると，Python の標準ライブラリでは，os モジュールと sys モジュールにそれぞれ path という名前が定義されている.「os.path」は，モジュールで,「sys.path」はモジュールを読み込むディレクトリを記憶したリスト (変数) である．この二つを from 文でインポートすると，先にインポートした「sys.path」を上書きして，次にインポートした「os.path」の内容を path という名前に代入され，元の「sys.path」が実行できなくなるというトラブルが発生する．

このような弊害が起こるため,「*」を使用したインポートには注意する．

2.5.2 モジュールの検索順

Python がモジュールをインポートする際には，決まった順書をたどってモジュールを探す．もし，重複した名前のモジュールがあった場合には，モジュール検索時に優先順序の高い場所にあるモジュールを優先して利用する．Python がモジュールを検索する場合に利用する順位は以下の用になっている．

1. ホームディレクトリ
   スクリプトファイルを指定して Python を起動している場合は，ファイルの置いてあるディレクトリがホームディレクトリになり，インタラクティブシェルで Python を稼働している場合は，カレントディレクトリがホームディレクトリとなる．

2. 環境変数 PYTHONPATH に設定されているディレクトリ
   環境変数 PYTHONPATH を指定することによって，Python がモジュールを読み込む時に検索対象となるディレクトリを指定する事が可能となる．環境変数が設定されていない場合は利用しない．

3. 標準ライブラリのモジュールディレクトリ
   このディレクトリは Python のインストールを行った環境によって異なる．複数バーションの Python がインストールされている場合には，バージョンにより別々のディレクトリとなる．

4. 追加のモジュールを設置するためのディレクトリ
   標準ライブラリのモジュールディレクトリにある site-packages ディレクトリを対象に検索を行う．ここには，追加でインストールしたモジュールを置く．

3 パッケージ

3.1 概要

パッケージとは，複数のモジュールを束ねて管理する仕組みである．このパッケージを使用する事で，複数のモジュールを一つのパッケージの配下に収める事が可能となる．パッケージの実態は，モジュールとなるファイルを収めたディレクトリである．

3.2 パッケージの使用方法

パッケージ内のモジュールをインポートするには，以下のように書く．

```
import パッケージ名．モジュール名
```

また，モジュールに定義されている関数を実行するには，以下のようにする．

```
パッケージ名．モジュール名．関数名 ()
```

この場合には，パッケージ名，モジュール名は省略できない．必ずモジュールまで指定する必要がある．

from 文を使用する事で，間の階層を飛び越えて，モジュールを呼び込む事が可能となる．

```
from パッケージ名　import　モジュール名
```

このように書く事で，モジュール名と関数名で実行する事が可能となる．パッケージの階層が深くなり，モジュールが多くなると，インポートの方法は何種類にもなり，複雑になる．

3.3 パッケージの作成

Python のパッケージの実態はディレクトリであるが，すべてのディレクトリがパッケージとしてインポートの対象となるわけではない．パッケージとして使用したいディレクトリには「__init__.py」というファイルを設置する必要がある．これを設置する事で，パッケージをインポートすると，このファイルがまず読み込まれ，トップレベルのブロックが実行される．

4 標準ライブラリ

4.1 概要

Python では，プログラムで実行する比較的高度な処理に必要な機能を標準ライブラリと言う形でまとめて提供されている．標準ライブラリを使用する事で，ネットワークアクセスやファイル操作のような処理を行うプログラムを比較的簡単に書く事が可能となる．

4.2 標準ライブラリの紹介

Python で利用できる標準ライブラリのうち，便利なモジュールを以下に紹介する．また，その他の Python モジュールを利用したい場合は，Python 標準ドキュメント [1)] を参照して頂きたい．これを参照する事で，モジュールをインポートするときに使うモジュール名や，各モジュールに定義されている関数や変数の詳細といったモジュールの仕様書を見る事ができる．

4.2.1 ファイル処理

Python には，コマンドラインから利用する，ファイル名を与えて処理するようなスクリプトを書く時に便利な「fileinput モジュール」が用意されており，これを利用すると，ファイルを扱うコードを簡単に書ける．

for ループで input() 関数を呼び出し，ファイルを開いた状態では，以下の関数が利用できる．

- input()
  for ループに添えるシーケンスと同じように動作する．スクリプトを呼び出す時にコマンドラインで指定されたファイルを一行一行読み込み，ループ編集に代入する．また，コマンドラインの引数に複数のファイルが指定された時は，指定した順番にファイルを読み込む．また，スクリプトを呼び出す時，コマンドラインに指定したファイルが存在しない時は，エラーを返す．

- filename()
  現在読み込み中のファイル名を返す．ファイルの読み込みが始まっていない時は None を返す．

- lineno()
  読み込みを行っている行数を数値で返す．複数のファイルを対象に読み込みを行っている場合は，これまで読み込んだ総行数を返す．読み込みが始まっていない時は「０」を返す．

- filelineno
  現在読み込み中のファイルの行数を数値で返す．

- nextfile()
  現在のファイルを閉じ，もし次に処理すべきファイルがある場合は，そのファイルの最初の行を読み込む．

- close()
  シーケンスを閉じ，ループを終了する．

4.2.2 数学関数，乱数の利用

Pyrhon では，数学的な処理をする演算に必要な関数は，math モジュールにまとまっている．ここでは，数学関数の他に，乱数に関連する関数の集合である，random モジュールについて紹介する．

まず，math モジュールについて紹介する．以下のような関数が用意されている．

- pi
  数学で利用する定数パイ (π) を定義した定数．

- e
  数学で利用する定数 e を定義した定数．

- pow(x,y)
  x を y で累乗した数値を計算して返す．「x**y」と同じ．

- sqrt(x)
  x の平方根を計算して返す．

- radians(x)
  x を角度からラジアンに変換して返す．

- degrees(x)
  x をラジアンから角度に変換して返す．

- sin(x),cos(x),tan(x)
  三角関数を計算して返す．

- asin(x),acos(x),atan(x)
  逆三角関数を計算して返す．

- sinh(x),cosh(x),tanh(x)
  双曲線を計算して返す．

- exp(x)
  数学定数 e の x 乗に相当する値を計算して返す.

- log(x[,base(底)])
  x の自然対数を計算して返す．base を底とする対数も扱う事ができる.

- log10(x)
  x の 10 を底とした対数を計算して返す.

次に random モジュールについて紹介する．以下のような関数が用意されている.

- randint(a,b)
  「a 以上 b 以下」のランダムな整数を発生させる.

- uniform(a,b)
  「a 以上 b 以下」のランダムな実数を発生させる.

- random()
  0 以上 1 以下のランダムな浮動小数点を発生させる.

- randrange([start(開始数)],stop(終了数)[,step])
  for ループなどで利用する組み込み関数 range() の乱数板．range() が発生する順番通りのシーケンスではなく，順番をランダムに並べ替えたシーケンスを作成し，そこから要素を返す.

- choice(seq(シーケンス))
  引数として渡したシーケンスの中から，ランダムに要素を選んで返す．空のシーケンスを渡すとエラーが発生する.

- shuffle(x(シーケンス))
  引数として渡したシーケンスの要素を，ランダムに入れ替える．引数として渡したシーケンス自体を変更する.

- sample(population(シーケンス),k())
  引数 population の中から k 個，ランダムに要素を抜き出したリストを返す．要素を抜き出すとき，同じ要素は選ばない．要素の並びもランダムになる．標本用の母集団から必要な個数のサンプルを選ぶときに利用する.

- seed([x])
  random モジュールが利用している乱数発生器を初期化する．引数を与えない場合は，システムの時間を利用する．この関数は，random モジュールがインポートされた時に呼び出す.

4.2.3 csv ファイルの操作

複数の要素をカンマ (,) で区切って並べた csv ファイルを扱うためのモジュール．csv ファイルは，表計算ソフトやデータベースからファイルを書き出す際に用いられるファイル形式である．csv モジュールでは，dialect という仕組みを使って，異なる形式の csv ファイルを扱う.

csv モジュールには以下のような関数が定義されている.

- reader(csvfile(ファイルオブジェクト))
  csv ファイルをファイルオブジェクトに指定して呼び出し，reader オブジェクトと呼ばれるオブジェクトを返す．reader オブジェクトは，for 文に添えるなどして利用する．reader オブジェクトはイテレータの一種で csv ファイルを一行ずつ読み込んで処理を進める．reader オブジェクトが csv ファイル行を読み込むと，要素に分割してリストに格納し返す．dialect という引数はオプションである.

- writer(csvfile(ファイルオブジェクト))
  要素を設定に区切った csv ファイルを書き出すために利用する．書き込みができるモードで開いたファイルオブジェクトを引数に与えて呼び出す．オプションの引数 dialect にはエクセルのファイル形式を指定する.

- writerow(row(シーケンス))
  シーケンスを引数を与えると，要素を設定に従って区切って書き出す．writer オブジェクトを作るときに指定したファイルオブジェクトに対して書き出しを行う.

- writerows(rows(シーケンス))
  シーケンスを要素に含むシーケンスを引数に渡すと，要素を区切って複数の行を書き出す.

5 演習問題

標準ライブラリを使用し，以下の数式を計算し，その結果を csv ファイルに出力せよ.

$$F = \sum_{i=1}^{5} 5sin(2\ \pi * i) + \sum_{i=1}^{5} icos(2\ \pi * r) \quad (1)$$

$$r = [0, 1] \quad (2)$$

参考文献

1) Python 標準ドキュメント
   http://www.python.jp/doc/

2) 柴田淳，みんなの Python，SoftBank Creative，2006