Lightweight Language Saturday

Python Language Update

柴田 淳
(shibata@webcore.co.jp)
日本Pythonユーザ会/ウェブコア株式会社

1) What's Python?

1) What's Python?

●基本情報

オブジェクト指向インタープリタ言語

クラス，関数などもオブジェクトとして扱うことができる

モジュール

例外処理

Unicode文字列のサポート

オープンソース
(2.0.1からOpen Source Initiative (OSI) Certifiedのライセンス)

最新バージョン 2.3
(7月29日にリリース)

Python.org(本家)
http://www.python.org/

日本Pythonユーザ会(PyJUG)
http://www.python.jp/

1) What's Python?

Pythonは潔い

●言語仕様がシンプルかつエレガント

習得が容易(特に，他のプログラミング言語を知っている場合)
直感を裏切らない

●可読性が高い

見れば分かる
ブロックの範囲をインデントで表現 - コーディングスタイルが安定する。
ブロックの範囲を判別しやすい。

●拡張性が高い

オブジェクトの挙動の多くが開発者に対して「開かれている」
コンストラクタ/ディストラクタやオペレーターだけでなく，
リスト，アトリビュートアクセス用のメソッドなどをオーバーライドし，
挙動を変更できる。
C,C++による拡張/連携が容易
動的な型チェック - オブジェクトの振る舞いを動的に変更できる。

1) What's Python?

Pythonは癒す（1/2）

●生産性が高い

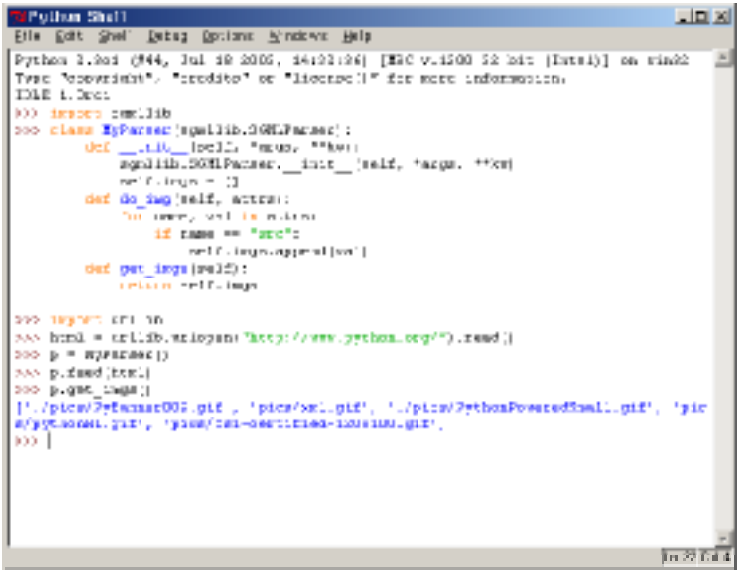
組み込みのデータ型が強力 - リスト，ディクショナリーなど
構造を持ったデータをソース中に表記できる
豊富なライブラリー - HTTP,FTP,POP/IMAP(ネットワークプロトコル)
GUIライブラリ/XML,HTMLなどのパーサー,etc...
タイプ量が少なくて済む
「Javaの1/5」という説がある
(http://www.artima.com/intv/speed.html)
関数型，オブジェクト指向のプログラミングスタイルを適宜切り換え，
混在できる

●対話モード

ちょっとしたテストコードなどを書ける

idle
(an Integrated DeveLopment Environment for Python)

Python のプログラミングを行うための統合環境，
クロスプラットフォームで動作
デバッガ(ブレークポイントの設定，ステップ実行)



1) What's Python?

Pythonは癒す（2/2）

●Jython

CPythonのJava実装

Javaバイトコードへの
動的コンパイル

既存のJavaクラスを
拡張する能力

オプションの静的コンパイル

Beanプロパティ

Jython.org(本家)
http://www.jython.org/

Jython.jp
http://www.jython.jp/

1) What's Python?

Pythonは裏切らない

●言語仕様が安定している

小さなコア，大きなモジュール
基本的な言語仕様はあまり変わらず，モジュールの拡張，追加がメイン

●マルチプラットフォーム

Unix,Windows,Macintosh,WindowsCE,LinuxZaurus...

●単機能スクリプトからサーバーまで

Mailman

PythonベースのMailingListエンジン
本家
http://www.list.org/
国際化 Mailman に関する日本語情報
http://mm.tkikuchi.net/

Zope

Pythonベースのアプリケーションサーバー
(あるいはオブジェクトパブリッシング環境)
本家　　JZUG(日本Zopeユーザ会)
http://zope.org/　http://zope.jp/

Twisted

Pythonベースの開発フレームワーク。あらゆるネットワークプロトコルを組み込むことを目指している。
Twisted Matrix Labs
http://twistedmatrix.com/products/twisted

OpenOffice 1.1
MacOS X 10.3(Panther)

グラフィックライブラリにPythonからアクセスできるライブラリを提供すると発表。
http://www.apple.co.jp/news/2003/jun/24panther.html

2) Language Update

2) Language Update

Python 2.0

2000年10月16日リリース

●Unicodeサポートの追加(正式には1.6から)

●List Comprehentions（リストの内包表記）

ロジックを使ってリストを生成する際などに便利

2.0以前

```
>>> L = ["pat","jhon","robben","jef"]
>>> sublist = filter( lambda s: s.find("j") == 0,L )
>>> print sublist
['jhon', 'jef']
>>>
```

2.0以降

```
>>> L = ["pat","jhon","robben","jef"]
>>> sublist = [ s for s in L if s.find("j") == 0 ]
>>> print sublist
['jhon', 'jef']
>>>
```

Ex.特定の条件 - "j"で始まる要素だけを抽出したい場合

●複合代入演算子の追加(+=,-=,*=など)

2) Language Update

Python 2.1

2001年4月17日リリース

●スコーピングルールの変更(オプショナル，2.2から標準)

「Nested Scopes」
これまで3種類のみだった名前空間に
「静的スコープ」を追加。
スコーピングのルールがよりスマートで直感的に。

※参考
Python Warts(Pythonの暗黒面)
Pythonの「暗黒面」について論じているエッセイ(英文)。
http://www.amk.ca/python/writing/warts.html

●比較演算子(<,>などの比較演算子を個別に実装可能に)

2) Language Update

Python 2.2

2001年12月21日リリース

●iterator(イテレーター)

```
>>> L = [1,2,3]
>>> i = iter(L)
>>> i.next()
1
>>> i.next()
2
>>> i.next()
3
>>> i.next()
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
StopIteration
>>>
```

リストやディクショナリー，Fileなどにもiteratorサポートが追加された

2.2以前

```
for line in file.readlines():
    # 行ごとに処理をするコード
    ...
```

2.2以降

```
for line in file:
    # 行ごとに処理をするコード
    ...
```

●generator(オプショナル，2.3から標準に)

2) Language Update

Python 2.2

●タイプ,クラスの変更

これまで，「タイプ」と「クラス」の実装が異なっていた
「タイプ」はCの構造体として実装　　＞　継承できない
「クラス」はPythonのオブジェクト　＞　継承できる

「新しいスタイルのクラス」の導入

Guido曰く...
"This is one of the biggest changes to Python ever"

```
>>> isinstance([], list)
1
>>> isinstance([], dict)
0
>>> isinstance([], object)
1
>>>
```

組み込み型(list,fileなど)のサブクラスが可能に
CやC++で書いたクラスも継承可能

```
class LockableFile(file):
    def lock (self, operation, length=0, start=0, whence=0):
        import fcntl
        return fcntl.lockf(self.fileno(), operation,
                           length, start, whence)
```

2) Language Update

Python 2.2

- インスタンス変数の制限(__slots__)
- "__new__"の追加
  immutable(不変)な組み込み型(数値，文字列など)の生成時に呼ばれる
- 新しいアトリビュートアクセス(property)

  従来__getattr__や__setattr__をオーバーライドして，アトリビュートへのアクセスをフックしていた
  記述は複雑になりがち/処理速度的なオーバーヘッドにもなりやすい

```
class C(object):
    def get_size(self):
        result = ... サイズを計算 ...
        return result
    def set_size(self, size):
        ... サイズに合わせてインスタンス内部の状態を変更 ...

    # sizeにアクセスするときに呼ばれる関数を定義
    size = property(get_size,    #sizeを参照
                    set_size,    #sizeを変更
                    None,        #削除用の関数は未定義
                    "インスタンスのサイズ")
```

2) Language Update

Python 2.3

7月29日リリース - MacOS X 10.3(Panther)に間に合わせるため多少無理をした

変更点一覧
http://www.python.org/2.3/highlights.html
http://www.python.org/doc/2.3c1/whatsnew/

●No new syntax.

"構文に変更なし"(generatorで使うyieldを除く)

●高速化

ベンチマークでは，2.2と比較して20〜30%高速に
http://mail.python.org/pipermail/python-dev/2003-July/036864.html

Pythonのバージョン	2.1	2.2	2.3
pystone	5.02	4.68	3.68
html	19.96	20.97	15.93

2) Language Update

Python 2.3

●generator(ジェネレーター)
Python2.2ではオプショナル

```
def generate_ints(N):
    for i in range(N):
        yield i

>>> gen = generate_ints(3)
>>> gen
<generator object at 0x8117f90>
>>> gen.next()
0
>>> gen.next()
1
>>> gen.next()
2
>>> gen.next()
Traceback (most recent call last):
  File "stdin", line 1, in ?
  File "stdin", line 2, in generate_ints
```

●Boolean型
定数"True"と"False"が追加

```
>>> obj = []
>>> hasattr(obj, 'append')
True
>>> isinstance(obj, list)
True
>>> isinstance(obj, tuple)
False
```

c.f.(2.3以前)

```
>>> obj = []
>>> hasattr(obj, 'append')
1
>>> isinstance(obj, list)
1
>>> isinstance(obj, tuple)
0
```

Open Source Convention 2002
State of the Python Union - Guidoのプレゼンテーション - to bool or not to bool
http://conferences.oreillynet.com/cs/os2002/view/e_sess/2672
http://conferences.oreillynet.com/presentations/os2002/vanrossum_guido.ppt(PPTファイル)

2) Language Update

Python 2.3

●「スライス」の拡張

```
>>> L = range(10)
>>> L[::2]          #リストの偶数番目を取得
[0, 2, 4, 6, 8]
>>> L[::-1]         #リストを逆順に取得
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
>>> del L[::2]      #リストの偶数番目を削除
>>> L
[1, 3, 5, 7, 9]
```

●enumerate()
リストの添え字と要素を生成する

```
for i, item in enumerate(L):
    # itemを元に新しい値を計算...
    L[i] = result
```

c.f.(enumerateを使わない)

```
for i in range(len(L)):
    item = L[i]
    # itemを元に新しい値を計算...
    L[i] = result
```

●dict()によるディクショナリーの生成

```
>>> D = dict(key1=1,key2=2,key3=3)
>>> D
{'key3': 3, 'key2': 2, 'key1': 1}
>>> D.pop("key1")
1
>>> D
{'key3': 3, 'key2': 2}
>>>
```

2) Language Update

Python 2.3

●新しいデータ型「Set」

それぞれの要素が単一になる(=重複しない)リスト「集合型」

```
>>> import sets
>>> S = sets.Set([1,2,3])      #Set型を生成
>>> S
Set([1, 2, 3])
>>> 1 in S                     #リストと同じ操作ができる
True
>>> 0 in S
False
>>> S.add(5)                   #追加
>>> S.remove(3)                #削除
>>> S
Set([1, 2, 5])
>>> S.add(5)                   #既存の値を追加
>>> S                          #なにも起こらない
Set([1, 2, 5])
>>>
```

```
>>> S1 = sets.Set([1,2,3])
>>> S2 = sets.Set([4,5,6])
>>> S1.union(S2)                    #結合
Set([1, 2, 3, 4, 5, 6])
>>> S1 | S2                         #結合の別表記
Set([1, 2, 3, 4, 5, 6])
>>> S1.union_update(S2)             #更新しつつ結合
>>> S1
Set([1, 2, 3, 4, 5, 6])
>>> S1.intersection(S2)             #交差
Set([])
>>> S1 & S2                         #交差の別表記
Set([])
>>>
>>> S1 = sets.Set([1,2,3,4])
>>> S2 = sets.Set([3,4,5,6])
>>> S1.symmetric_difference(S2)
Set([1, 2, 5, 6])
>>> S1 ^ S2                         #別表記
Set([1, 2, 5, 6])
>>>
```

2) Language Update

Python 2.3

- **Unicodeサポートの強化**
  compile(), eval(), exec,filter(),raw_input()
- **プラットフォームごとの「改行コード」の差を吸収**
  LF,CRLF,CRを"\n"で表現
- **import時のフック関数**
  モジュールをimportする時に，互換性をチェックしたり警告を表示できるようになった
- **Stringに演算子が追加**
  in,strip,rstrip,lstrip,startwith,endwith,zfill(Zero Fill)
- **モジュールの追加，改良**
  csv,shelve
  bsddb,logging
  zipimport
  タイムアウト対応のSocket
  etc...