

Jython

The logo for Jython, a Python implementation for the Java Virtual Machine. The word "Jython" is written in a bold, black, sans-serif font. A green snake is integrated into the letters "thon", with its head positioned above the 't' and its body winding through the 'h', 'o', and 'n'.

```
>>> from java.lang import *
```

Jythonチュートリアル

2008-08-28

@JJUGクロスコミュニティ

サイボウズ・ラボ株式会社

西尾泰和

時間割

- 1時間目 世界史
- 2時間目 国語
- 3時間目 音楽
- 4時間目 物理
- 5時間目 倫理

はじめの会

- Jythonの読み

Jythonの読み

- Jythonはジャイソンと読む
- たまにジェイソンと読む人がいる
- JSON(Java Script Object Notation)と紛らわしい
- あと殺人鬼とも紛らわしい

はじめの会

- Jythonの読み
- Jythonってなんなの？

Jythonってなんな

- Javaで実装されたPython
- Pythonはオブジェクト指向のスク립ト言語
- スクリプト言語からJavaのクラスを部品として使える

Jythonってなんな

- Javaはかっちり堅く書くのに向いている。製図用ペン
- Pythonはささっと軽く書くのに向いている。下書き用鉛筆
- 適材適所で使い分ける

Jythonってなんなの？

- 一言で要約すると

JRubyのよ

うなもの

はじめの会

- Jythonの読み
- Jythonってなんなの？
- 注意

注意

- 「Jythonではこんなことができる！」
- 「JRubyではできない！」
とか「PureJavaではできない！」
という意味ではない

注意

- スライドが62枚あるのでよそ見をするとわからなくなります



世界史

- Q: Javaが出現したのは何年ごろでしょう？
- A: 1995年 Java1.0

世界史

- Q: Pythonが出現したのは何年ごろでしょう？
- A: 1994年 Python1.0
- Javaより1年早い！

世界史

- Q: Jythonが出現したのは何年頃でしょう？
- A: 1997年 JPython1.0
- A: 2000年 Jython2.0

年表

- 1994年 Python1.0
- 1995年 Java1.0
- 1997年 JPython1.0
- 2000年 Jython2.0
- 2001年 Jython2.1

- 2001年 Jython2.1
- 2001年 AspectJ by Jim Hugunin
- 2001年 JRuby0.1.8
- (5年の月日が流れ...)
- 2006-05 JRuby0.9
- 2006-09 JRuby開発者Sunへ
- 2007-02 Jython2.2 beta1

- 2007-02 Jython2.2 beta1
- 2007-06 JRuby1.0
- 2007-08 Jython2.2
- 2008-03 Jython開発者Sunへ
- 2008-07-15 Jython2.5a1
- 2008-07-19 JRuby1.1.3

競争なくして

発展なし

by 小泉純一郎

ようやく

Jythonも

本気を出した

ようです



時間割

- 1時間目 世界史
- 2時間目 国語
- 3時間目 音楽
- 4時間目 物理
- 5時間目 倫理

国語

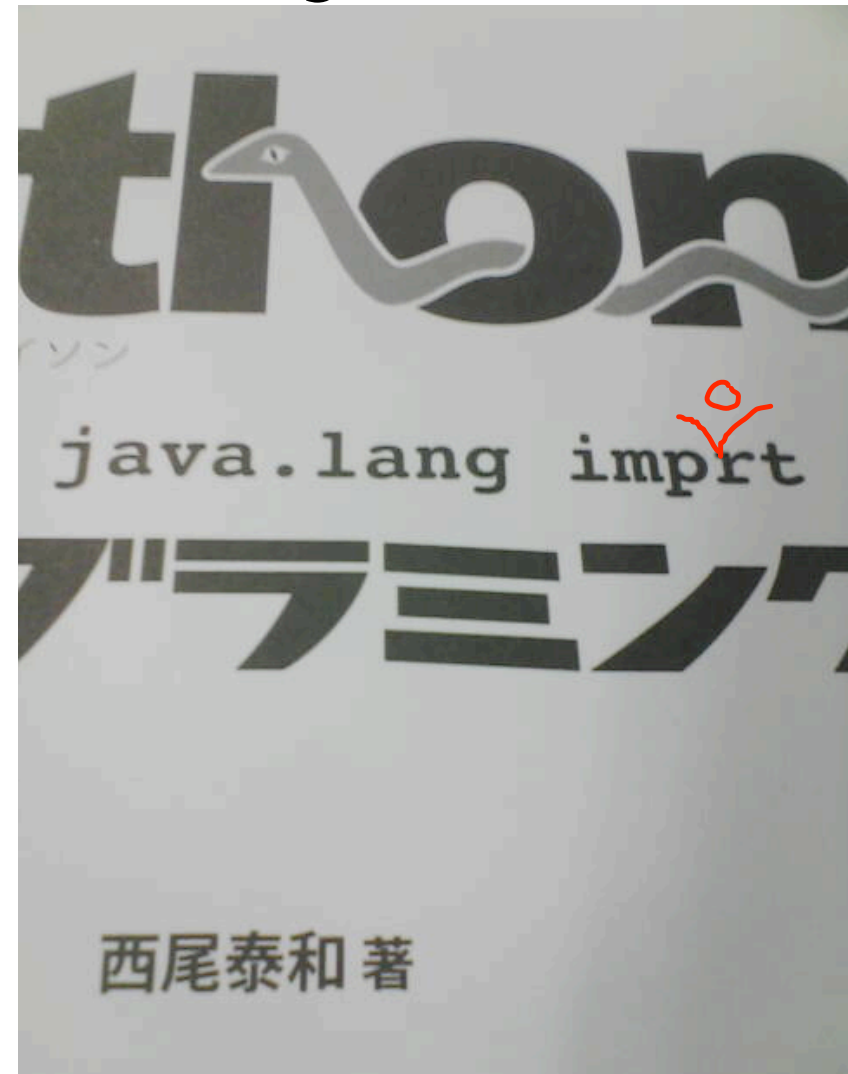
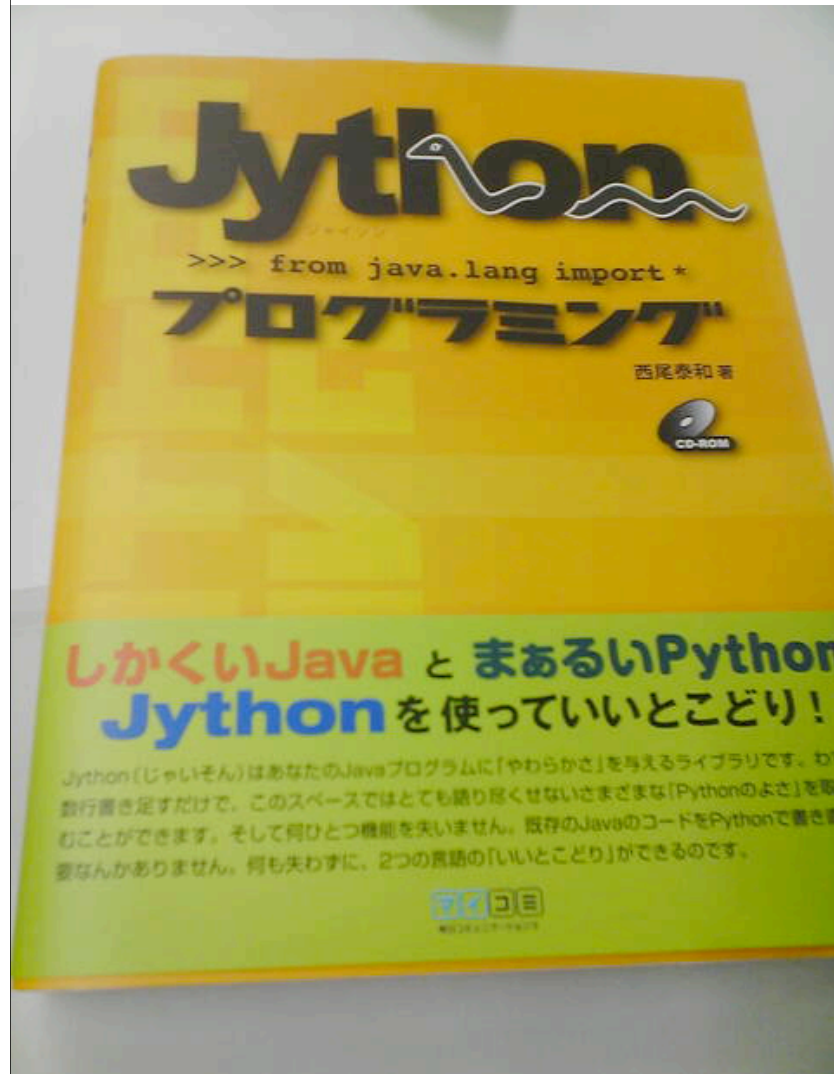
母国語は

Python

15分で学ぶPython

```
from java.lang import *  
import java.lang.*
```

15分で学ぶPython



15分で学ぶPython

(Jython起動)

15分で学ぶPython

`System.out.println("Hello")`

クラスの宣言いらさない

mainメソッドいらさない

セミコロンのもいらさない

15分で学ぶPython

System: クラス

System.out:

java.io.PrintStreamのインスタンス

System.out.println: メソッド

Javaのものを使い放題

15分で学ぶPython

しかもすべてが

第一級のオブジェクト

```
p = System.out.println  
p("Hello")
```

15分で学ぶPython

JyConsole起動

LLの話をする

「LLは補完がきかないから」

とおっしゃる方がいらっしゃる

<http://dev.artenum.com/projects/jyconsole>

15分で学ぶPython

補完デモ

```
import javax
from javax.swing import *
f = JFrame("hello")
f.setVisible(1)
f.setSize(100, 100)
f.title = "Hello, JyConsole" # 余談
```

15分で学ぶPython

もちろんJavaのほうが

静的な型情報がある分

補完が期待できる。

Pythonは動的型付け。

柔軟さと引き替え。

15分で学ぶPython

f.add(console)
コンソールが移動

"console"という予約語が
あるわけじゃなくて

pyi.set("console", console)
してあるだけ

15分で学ぶPython

```
JyConsole console = new JyConsole();  
pyi = console.getPythonInterpreter();  
pyi.set("console", console);  
pyi.set("frame", frame);
```

まとめ

- クラスやメソッドや型の宣言などいろいろなものがない
- Javaのクラスをインポートして使える
- メソッドを変数に代入できる
- JyConsoleでEclipseほどじゃないけど補完もきく
- Javaのオブジェクトも簡単に持ち込める



時間割

- 1時間目 世界史
- 2時間目 国語
- 3時間目 音楽
- 4時間目 物理
- 5時間目 倫理

音楽

Jythonで

MIDIをつかう

注) JREではなくJDKを使う必要がある

Jython 🐘 MIDI

```
>>> import javax.sound.  
midi.MidiSystem as MS  
>>> MS.getSynthesizer()  
..  
>>> s = _  
>>> c = s.getChannels()[0]  
>>> c  
..
```

JythonでMIDI

鍵盤を押さえる

```
>>> c.noteOn(48, 127)
```

鍵盤を離す

```
>>> c.noteOff(48)
```

楽器を変えてみよう

```
>>> c.programChange(69)
```

```
>>> c.noteOn(48, 127)
```

```
>>> c.noteOff(48)
```

Jython 🐘 MIDI

```
>>> from time import sleep
>>> def play(note, time=0.25):
...     c.noteOn(note, 127)
...     sleep(time)
...     c.noteOff(note)
>>> play(48)
>>> c.programChange(0)
>>> play(48)
```

JythonでMIDI

```
>>> for x in "NISHIO HIROKAZU":  
...     play(ord(x))  
# スペースは32なので4オクターブ  
くらい低い
```

まとめ

- インタラクティブ！



時間割

- 1時間目 世界史
- 2時間目 国語
- 3時間目 音楽
- 4時間目 物理
- 5時間目 倫理

GRINEdit

物理演算を使ったネットワーク可視化
物理法則を実行時に差し替えられる

```
grinedit.delLaw("PL_SpringEdge")  
grinedit.addLaw("PL_Flow", {})  
grinedit.modLaw("PL_Repulsion",  
{"repulsionRadius": 10.0})
```

GRINEdit

```
med.getVertexList()
```

```
v = med.getVertexList()[0]
```

```
v.__class__
```

```
v.setBackgroundColor((255,0,0))
```

GRINEdit

```
for v in med.getVertexList():  
    if v.getLabel() in ["23", "73", "28", "78"]:  
        v.setAnchored((0, 0))
```

demonstrate history by press UP-key

```
for v in med.getVertexList():  
    if v.getLabel() in ["23", "73", "28", "78"]:  
        v.setDisanchored()
```

まとめ

- インタラクティブ！
- Javaのオブジェクト
をリアルタイムに操
作できる



時間割

- 1時間目 世界史
- 2時間目 国語
- 3時間目 音楽
- 4時間目 物理
- 5時間目 倫理

String破壞

```
>>> s = String("Hello, Python")
>>> s.substring(7)
>>> python = _
>>> python
>>> s.value
>>> s.value[7] = 'J'
>>> s
>>> python
```

String破壞

```
public final class String  
    implements java.io.Serializable  
{  
    /** The value is used for char  
    private final char value[];
```

String破壞

respect

Java Accessibility

= false

```
System.setProperty(  
    "python.security.respectJavaAccessibility",  
    "false");
```

まとめ

- 俺にプライベート
はないのか！(J氏)
- でももちろんPure Java
でもできること



終わりの会

- 駆け足でJythonについて語った
- 「Jythonを使うとJavaのコードをPythonで書き換えなきゃ行けない」という誤解があるようだ
- Jythonはプロジェクトにやわらかさを与えるライブラリ
- Javaが適している部分はJavaでいい

終わりの会

- 例えばよく変わる処理だけスクリプトにくくりだして再コンパイルなしで変更できるようにしたいとき
- 例えば変更できるパラメータがたくさんあるけど実行時に変更できるようなGUIを作るのは面倒なとき

Javaと

Pythonが

結婚すれば

幸せ二倍

不幸半分

ご清聴

ありがとうございます

ございました



付録：ソースコード

```
package org.nhiro.jython;

import java.awt.BorderLayout;

import javax.swing.JFrame;

import com.artenum.jyconsole.JyConsole;
import com.artenum.jyconsole.python.JInteractiveInterpreter;

public class DemoCore {
    private static JInteractiveInterpreter pyi;
    private static JFrame frame;

    public static void main(String[] args) {
        frame = new JFrame("Demo of JyConsole");
        System.setProperty(
            "python.security.respectJavaAccessibility",
            "false");
        JyConsole console = new JyConsole();
        frame.add(console, BorderLayout.CENTER);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 400);
        pyi = console.getPythonInterpreter();
        pyi.set("console", console);
        pyi.set("frame", frame);
        frame.setVisible(true);
    }
}
```