

10年前から来たphp.py

"php.py" came from the 2010's



pixiv Inc.
USAMI Kenta

pixiv

お前誰よ



- うさみけんた (@tadsan) / Zonu.EXE / にゃんだーすわん
- ピクシブ株式会社 pixiv事業本部 エンジニア
 - 最近はピクシブ百科事典(dic.pixiv.net)を開発しています
 - 会社ではPython使われてますが私は触ってません！！！！
- Emacs Lisper, PHPer
 - Emacs PHP Modeを開発しています(2017年-)

3週間前はPHP
カンファレンスでした
(両方に携っているスタッフ
おつかれさまです)

Emacs-PHP



Friends of Emacs-PHP development

Join us!

5 followers

<?php

Follow

Overview

Repositories39

Projects

Packages

Teams1

People8

Settings

Pinned

Customize pins

php-modePublic

A powerful and flexible Emacs major mode for editing PHP scripts

Emacs Lisp

550

115

phpactor.elPublic

Interface to Phpactor (an intelligent code-completion and refactoring tool for PHP)

Emacs Lisp

38

11

phpstan.elPublic

Interface to PHPStan (PHP static analyzer)

Emacs Lisp

22

11

Repositories

Find a repository...

Type

Language

Sort

New

mldocPublic

Multi EIDoc integration

Emacs Lisp

4

GPL-3.0

0

0

0

Updated 4 days ago

php-modePublic

A powerful and flexible Emacs major mode for editing PHP scripts

Emacs Lisp

550

GPL-3.0

115

68

7

Updated 5 days ago

View as: Public

You are viewing the README and pinned repositories as a public user.

You can [create a README file](#) visible to anyone.

[Get started with tasks](#) that most successful organizations complete.

People

Invite someone

Top languages

Emacs Lisp

Ruby

PHP

CSS

Most used topics

Manage

emacs

melpa

php

major-mode

4

pixiv

PHPの静的解析とか
言語仕様の紹介をして
暮らしています

普段はEmacsの
php-mode開発したり

WEB+DB PRESS総集編



Software Design 11月号



この記事は執筆時点で10月2～3日に開催を予定しているPHPカンファレンス2021と連動した短期集中連載の最終回です。つまり、この記事が掲載されたSoftware Design 2021年11月号をみなさんが手にとられるころにはすでに開催されています。カンファレンスで発表されたセッションはYouTubeチャンネル^{注1}にアーカイブが公開されているはずですので、興味のある方はぜひご覧ください。筆者はPHPカンファレンス2021で「配列、ジェネリクス、PHPで書けない型」と題したトークを発表します。

最終回では近年の動的言語の型検査の動向についての概観に触れたあと、PHPの動向について説明します。

動的型付き(statically typed)の言語と呼びます。事前にデータの種類を固定せずに処理することを動的型検査(dynamic type checking)と呼び、実行時にデータに応じた処理をすることが基本のプログラミング言語を動的プログラミング言語(dynamic programming language)と呼びます。

メジャーなものでは、C言語、Java、C++、C#、Go、Swift、Rust、Scala、Kotlin、Haskell、OCamlなどは静的型付きの言語です。また、Python、JavaScript、Ruby、シェルスクリプト、ClojureなどLisp系の言語、Smalltalk、そしてPHPは動的言語と呼ばれます。俗に実行ファイルを生成するものをコンパイル言語、ソースコードを直接実行するものをインタプリタ言語として

Emacsで動的言語を
リアルタイム静的解析
しながら快適に開発で
きるようにしています

Pythonは1行くらい
書いたことがあります

nishioさんに影響されて…

西尾泰和のブログ

NISHIO HIROKAZU's Blog

現在Jython本執筆中～

西尾泰和のプロフィール 西尾泰和の活動ダイジェスト 西尾泰和のブログ@Cybozu Labs

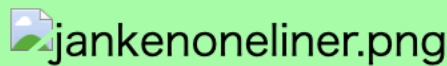
« [日記](#) | [Main](#) | [日記](#) »

« [Pythonでイテレータの復習](#) | [Python](#) | [Pythonの構文木を見る](#) »

Pythonでワンライナーを作成する際のノウハウ集

これは[LL Ring](#)というイベントの「じゃんけん2.0」に出場する際に「多くの構文に改行が必須であるPythonで書かれたじゃんけんエージェントをワンライナーにしていたらウケるかな」と思ってワンライナー化しているときに書いたメモです。自分用のメモのつもりだったので書き殴ってありますが、意外と人気のようなので近いうちに加筆します。実は後から書いた英語版([How to make oneliner in Python?](#))の方が整理されているのかも。

完成したワンライナー

jankenoneliner.png

def文を式にする

defは改行を要求するのでlambdaに置き換える必要がある。

```
def foo(x): return x + 1(ここに改行)

foo = lambda x: x + 1

globals().__setitem__("foo", lambda x: x + 1)
```

lambdaは式しか含むことが出来ないので、代入などの文は全部式に置き換える必要がある。

if文を式にする

```
if condition:
    p("True")
else:
    p("False")

condition and p("True") or p("False")
```

andやorが遅延評価することを利用している。このandとorを使った評価順の制御が、式しか使えないlambdaの中ではすべての基礎になる。

p("True")が0, NoneなどのFalseと判定される値になりうる場合は例えばこうする。

PythonでRubyを
書けるようにしたり

RubyをPythonっぽく置き換える



RELEASES ブログ GEMS ガイド ログイン 新規登録

pythonism *0.0.5*

This gem was made for friendship with Pythonistas.

バージョン履歴:

0.0.5 - February 22, 2016 (9KB)

0.0.4 - February 15, 2013 (8KB)

0.0.3 - February 15, 2013 (7.5KB)

0.0.2 - February 14, 2013 (7.5KB)

0.0.1 - February 14, 2013 (7.5KB)

オーナー:



作者:

USAMI Kenta

SHA 256チェックサム:

b34539a04018c8fb0485160786652aa25296aa98cb8b0ae0a1c1a0f6f7c55b37

← PREVIOUS VERSION

Star 0

累計ダウンロード数
12,437

このバージョンのみ
2,459

GEMFILE:

`gem 'pythonism', '~> 0.0.5'`

インストール:

`gem install pythonism`

ライセンス:

N/A

必要RUBYバージョン:

>= 0

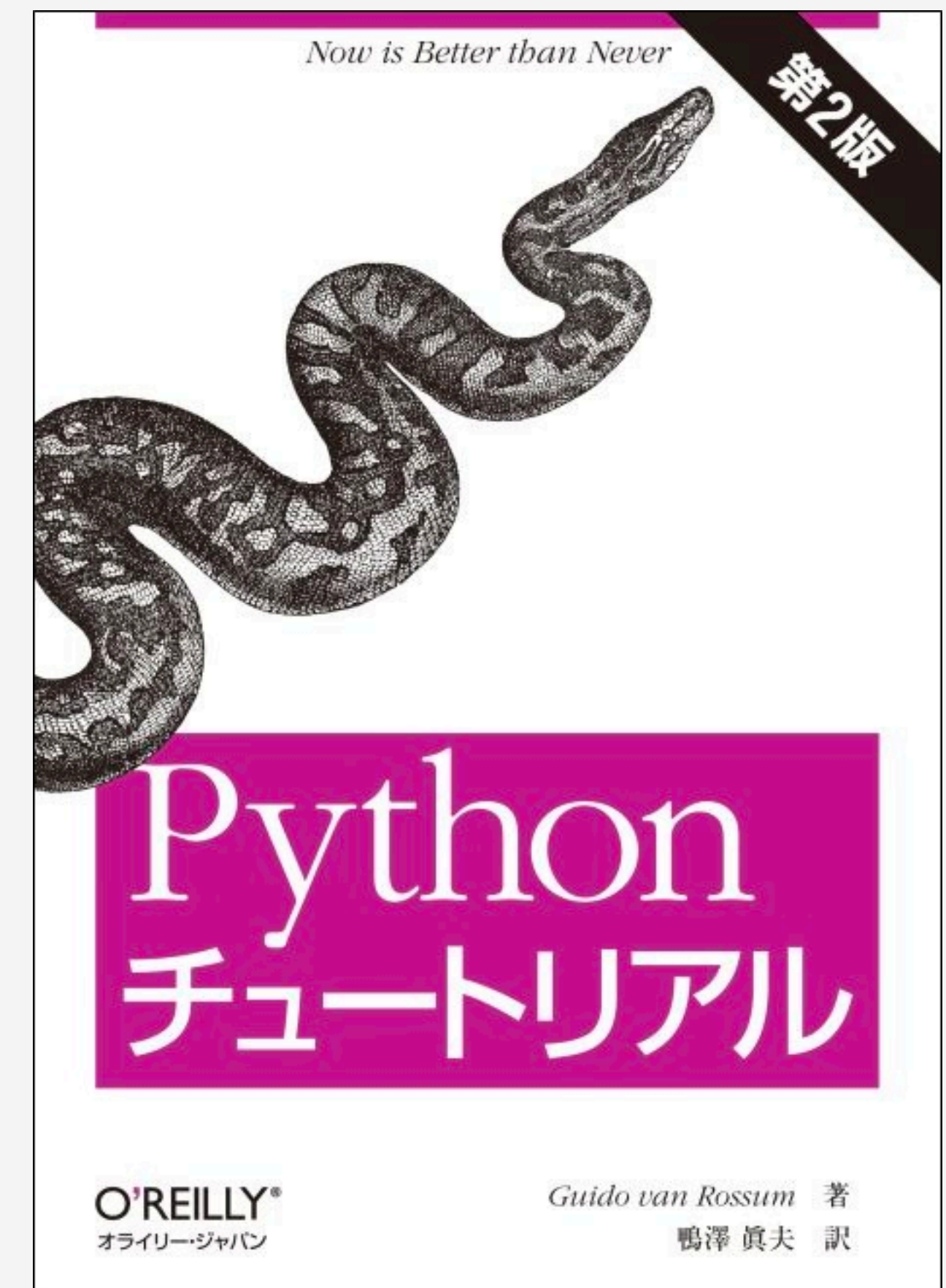
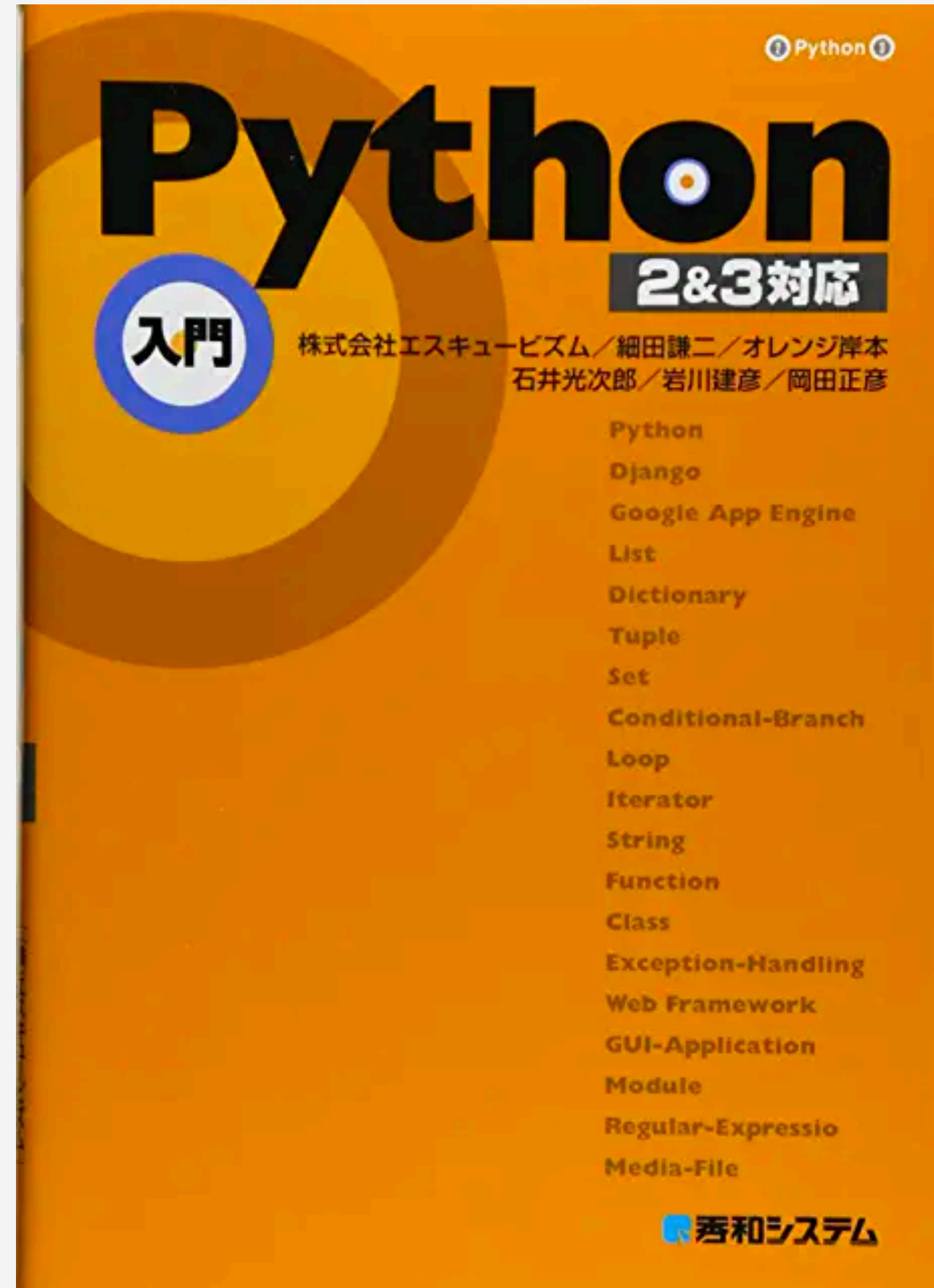
リンク:

ホームページ

冗談は置いておいて

2011年くらいに
Pythonやってみました

Pythonを学んだ本



札幌で読書会をやっていた

ATND BETA
PRODUCED BY RECRUIT

イベント開催支援ツール：ATND（アテンド）

ログイン or 新規登録

日本語
English

👉 オンライン決済サービスを使ったチケット販売ができる「eventATND(イベントアテンド)」がオープン！

▶ チケット販売はこちらから

Python初学者向け読書会@札幌 #8

Pythonで学ぶ、初めてのコンピュータサイエンス

 Tweet



日時： 2012/02/07 18:30 to 20:00

定員： 20 人

会場： 札幌市男女共同参画センター 3F OA研修室（札幌市北区北8条西3丁目 札幌エルプラザ）

URL： <https://groups.google.com/group/python-sapporo/>

管理者：  Zonu.EXE, tadsan.

ハッシュタグ： # pysap

 Google Mapsで表示

このイベントは終了しました

参加希望者 **6** / 20 人

参加： **6**

▼ 参加者 (6 人)

1.  Zonu.EXE, tadsan. : ハイサイ!

最初はRuby
並行してPython
就職してPHP

いわゆるL1言語を
浅く広く触っている

RubyとPHPの処理系 実装のことは ちょっとだけわかる

PyConは初参加！

隣の芝は青い

専門分野じゃない
カンファレンスに
飛び込むのは良い

というわけでPython
ニワカなりに10年前の
ことを知っています

Emacsで.py書くとき
どっちのpython-mode
使ってるんです？

PyCon楽しい！！

とはいえ
Pythonコミュニティに
影響を受けてきた

魂に刻まれた Pythonのヤバい機能

codec

文字コード変換する
やつでしょ？

昔はスク립トが
UTF-8以外で
書かれていた

Pythonは3で 文字列=Unicodeに 全振りした

ちなみにPHP6もやろうとしたけど失敗した

Pythonは任意のエン
コーディングで書かれ
たコードを実行できる
仕組みがある

LL Tiger (2010年)

渋谷日記@shibu.jp

渋谷よしきの日記です。ソフトウェア開発とか、ライフハックを中心に記事を書いていきます。

Top

Blog

Book List

Photos

Works

Translations

Profile

<< Sphinx 1.0 リリースパーティを開催しました！ | TOP | 日本語で Sphinx を使う時のストレスを減らす拡張機能 >>

2010年08月01日

LL TigerのLTでRuby on Python(仮)の発表してきました。

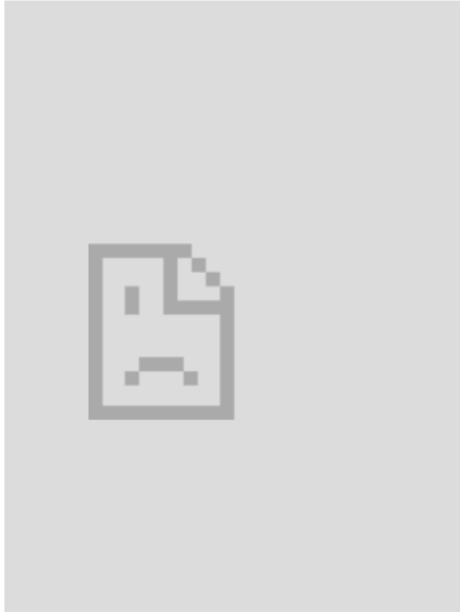


検索ボックス

検索

Twitter

www.flickr.com



What is this?

<< 2019年02月

>>

日 月 火 水 木 金 土

1 2

3 4 5 6 7 8 9

10 11 12

pixiv

<http://blog.shibu.jp/article/39920783.html> より引用

殺伐Python

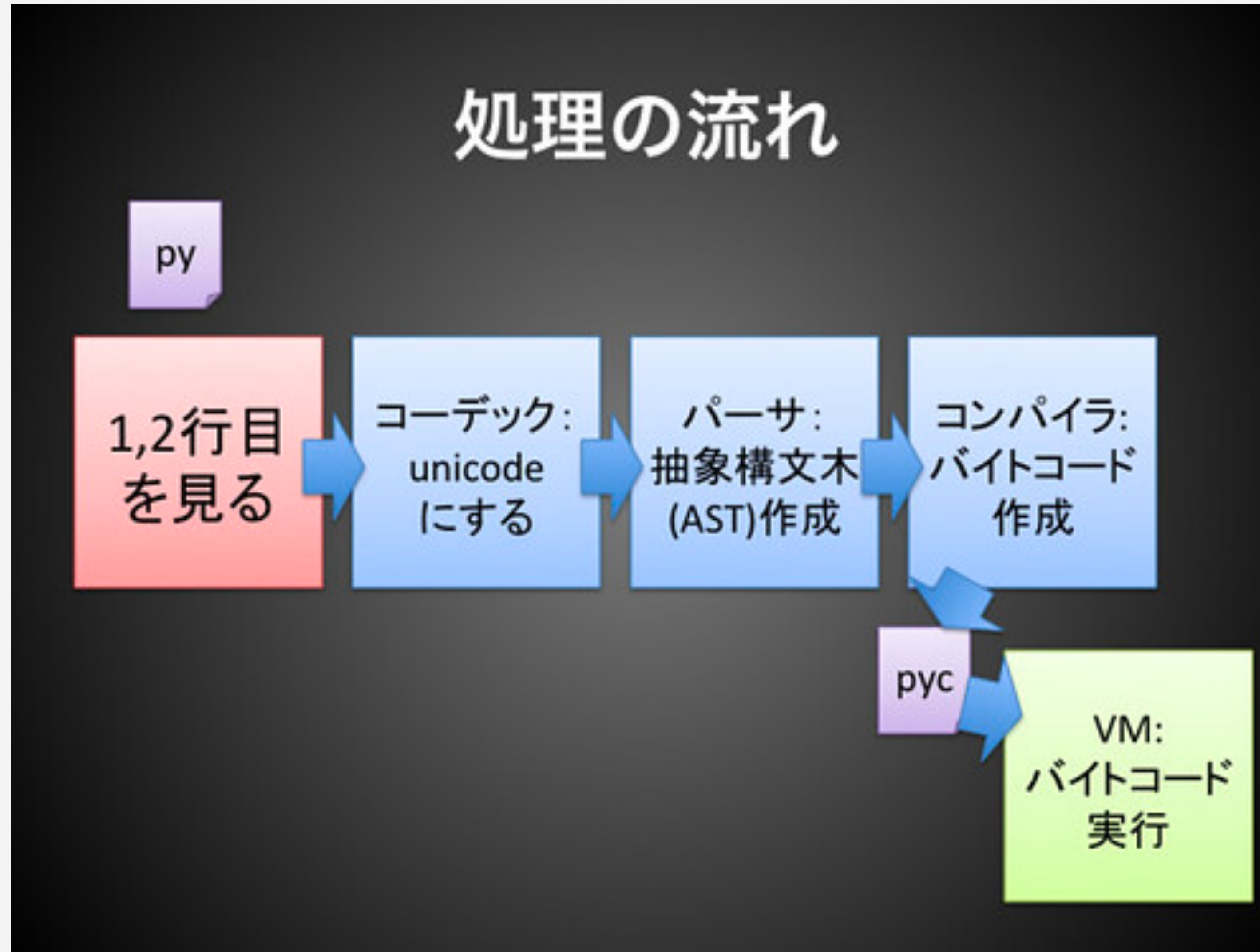
PEP-263

- たった1行で、あなたのPythonが！ -

殺伐Python
@shibukawa

ソースコードはすべて、<http://bitbucket.org/shibu> にアップロード済みです

殺伐Python



殺伐Python

```
# -*- encoding: utf-8 -*-  
  
amin1 = "私は逢坂さんがいいと思うなあ"  
taiga1 = "...なににい!?"  
amin2 = "ほら、だって逢坂さんって、すっごく"  
        "ちっちゃくてかわいらしいし?"  
taiga2 = "なに言ってるんだ超ばかちー!"
```

ソースコードに日本語が書けるように！
3系なら関数、クラス名も日本語OK

codecは追加できる

なんとcodecはプリプロセッサだったのだ

C言語風のマクロを実現するコーデックを実装してみました。(本番ではここでデモしました)

```
view plain  copy to clipboard  print  ?  
01.  # encoding: define  
02.  
03.  #define MSG "ll tiger"  
04.  
05.  print "hello", MSG
```

"define"というコーデックを指定すると、C言語のように、プリプロセッサとしてソースコードを加工することができます。

Shibu's Diary: LL TigerのLTでRuby on Python(仮)の発表してきました。
<http://blog.shibu.jp/article/39920783.html> より引用

!?

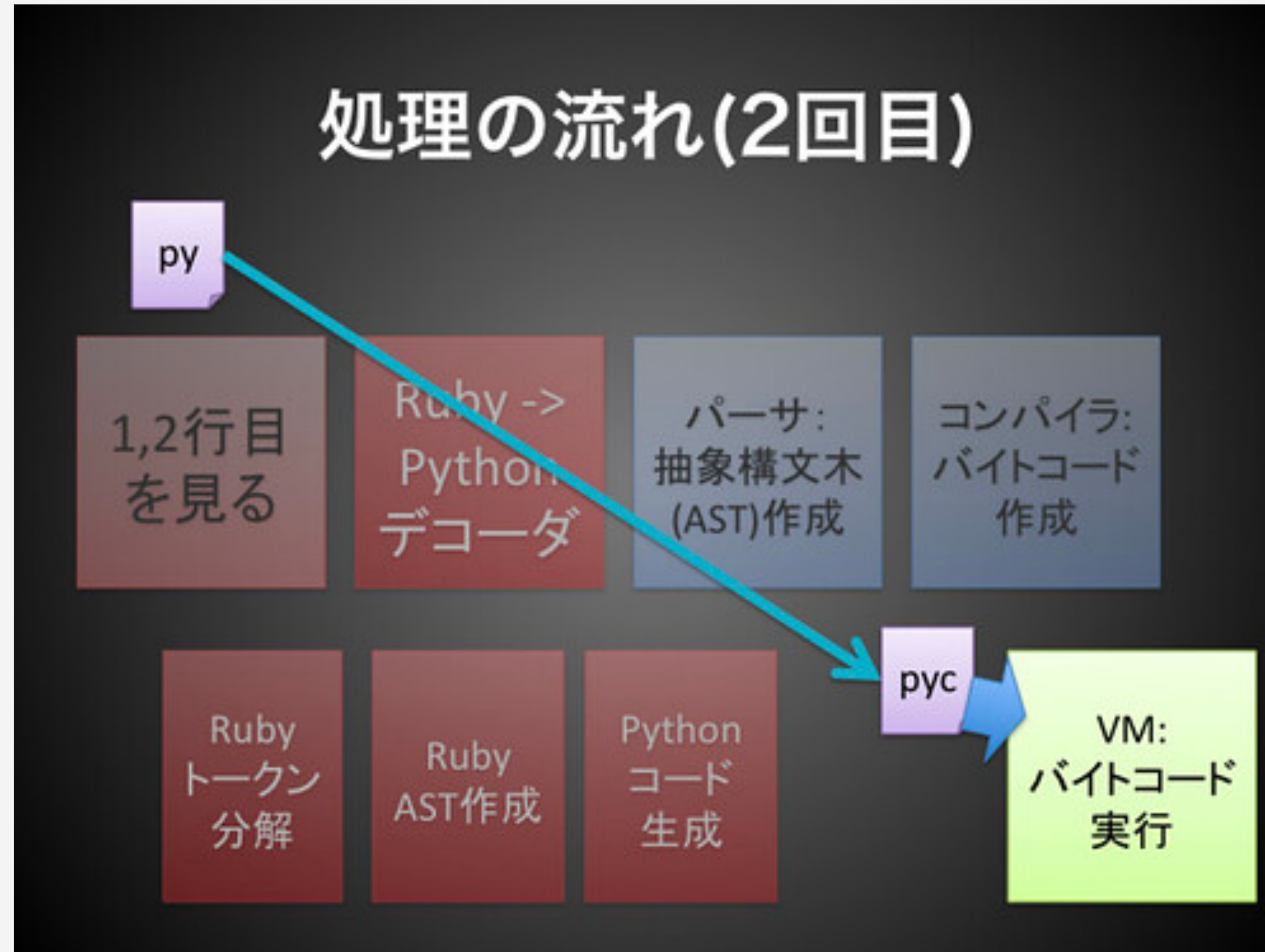
バイトコンパイル前に
介入できれば
事実上何でもできる

殺伐Python

Rubyでも動かしますか？

渋川さんはガチなので
Rubyのコードを
Pythonに翻訳します

殺伐Python



殺伐Python

ちなみに、重そうな処理に見えますが、うまくバイトコードファイルができて、一度キャッシュされてしまえば時間のロスはなくなります。(本番ではここでfizzbuzzのデモをしました)

```
view plain  copy to clipboard  print  ?  
01.  # encoding: ruby  
02.  
03.  (1..100).each do |i|  
04.    if i % 15 == 0  
05.      puts "fizzbuzz"  
06.    elsif i % 3 == 0  
07.      puts "fizz"  
08.    elsif i % 5 == 0  
09.      puts "buzz"  
10.    else  
11.      puts i  
12.    end  
13.  end
```


いま読み直しても
やばい

じゃあPHPでも
動かすか(安直)

私はガチではないので
お手軽実装で済ませる

というわけで
Pythonを書きました

1行くらい...

```
import encodings, codecs, sys, re, subprocess; from importlib import machinery; phpdecode=lambda input:('import helper;helper.run(b""{""}\n'.format("\n".join(str(input.tobytes(), 'utf-8').split("\n")[1:]).replace('"', '\\')), len(input)); search_function=lambda s:None if(s != "php")else codecs.CodecInfo(None, phpdecode, name='php'); codecs.register(search_function); loader=machinery.SourceFileLoader('php', sys.argv[1]); loader.load_module()
```


読みにくい...

phpcodec.py

```
import encodings, codecs, sys, re, subprocess
from importlib import machinery

def phpdecode(input):
    return (
        'import helper;helper.run(b"""{}""")\n'
        .format("\n".join(str(input.tobytes(), 'utf-8')
            .split("\n")[1:]).replace('"', '\\"')),
        len(input)
    )

def search_function (s):
    if (s != "php"):
        return None

    return codecs.CodecInfo(None, phpdecode, name='php')

codecs.register(search_function)

if (__name__ == '__main__'):
    loader = machinery.SourceFileLoader('php', sys.argv[1])
    loader.load_module()
```

(ファイルに書かれた
コードをPHPプロセス
に流すだけの雑実装)

みなさまおなじみのPHP

```
# encoding: php
<?=implode("\n",array_map(fn($n)=>match([( $n%3==0)+0,($n%5==0)+0])
{[1,1]=>'FizzBuzz',[1,0]=>'Fizz',[0,1]=>'Buzz',default=>$n},range(
1,100))))?>
```

改行したfizzbuzz.py

```
# encoding: php
<?php
foreach (range(1,100) as $n) {
    print match ($n % 3 == 0) {
        true => match ($n % 5 == 0) {
            true => 'FizzBuzz',
            false => 'Fizz',
        },
        false => match ($n % 5 == 0) {
            true => 'Buzz',
            false => (string)$n,
        },
    } . PHP_EOL;
}
```

実行

```
Welcome to the Emacs shell
```

```
~/repo/python/phpppy $ python3 -m phppcodec fizzbuzz2.py
```

```
1  
2  
Fizz  
4  
Buzz  
Fizz  
7  
8  
...
```


codecを現在動かすにあたって

- 公開されていた渋谷さんのコードが読めない (!!?)
 - BitBucketはMercurialサポート終了 <https://bitbucket.org/shibu>
 - 10年前はGitHub普及前でMercurialとBitBucket使われてましたね…
- 渋谷さんのコードを再利用したと思しき記事を断片的に参考に
 - <https://doloopwhile.hatenablog.com/entry/20100801/1280688409>

codecを現在動かすにあたって

- 元コードではStreamReaderでコードを書き換えていたが動かなかった
 - Pythonの処理系実装を軽く読んでみたけど理由わからず、教えてください
- sitecustomizeとかusercustomizeで暗黙的にコーデック追加して
#coding: php なファイルをダイレクトに起動できるようにしたい…
- 仕方ないので現在は `-m phpcodec` で実行したらダイナミックインポート
- もっといい感じにしたいね！

おまけ

エンコーディング指定

```
# encoding: utf-8
```

テキストエディタのエンコーディング指定に由来
(Rubyもだいたい同じ仕様)

```
# -*- coding: utf-8 -*-
```

エンコーディング指定

```
# encoding: utf-8
```

Emacs記法

これなに

```
# -*- coding: utf-8 -*-
```

PEP 263

```
^[ \t\f]*#.*?coding[:=][ \t]*([-_.a-zA-Z0-9]+)
```

Vim記法

それっぽい指定をいい感じに読み込んでもくれる

```
# vim: set fileencoding=utf-8 :
```