Python Distilled 試飲会

Hayao Suzuki

BPStudy #195

November 30, 2023

自己紹介

お前誰よ

Name Hayao Suzuki（鈴木　駿）

~~Twitter~~ X @CardinalXaro

Work Software Developer @ BeProud Inc.

- 株式会社ビープラウド
  - IT 勉強会支援プラットフォーム 

  - Python 独学プラットフォーム 

  - システム開発ドキュメントサービス 

自己紹介

発表したトーク（抜粋）

- SymPy による数式処理 (PyCon JP 2018)
- インメモリーストリーム活用術 (PyCon JP 2020)
- 組み込み関数 pow の知られざる進化 (PyCon JP 2021)
- Let's implement useless Python objects(PyCon APAC 2023)

`https://xaro.hatenablog.jp/` に一覧があります。

自己紹介

翻訳した本

- Python Distilled(O'Reilly Japan) 本日の主役

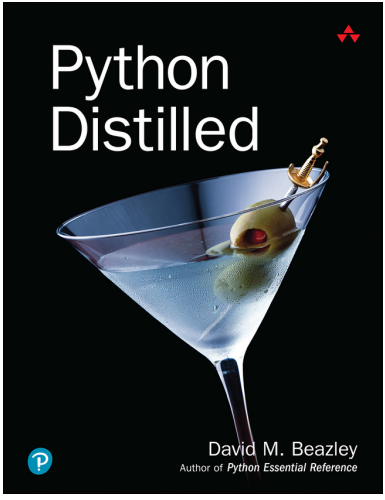
監訳した本

- 入門 Python 3 第 2 版 (O'Reilly Japan)
- ロバスト Python(O'Reilly Japan)

今日のテーマ


O'REILLY
オライリー・ジャパン
Python
Distilled
プログラミング言語Pythonのエッセンス
David M. Beazley 著
鈴木 駿 訳

今日のテーマ


Python
Distilled
David M. Beazley
Author of Python Essential Reference

Python Distilled

原著：Python Distilled

著者 David M. Beazley
出版年 2021 年 9 月
出版社 Addison-Wesley(Pearson)

邦訳：Python Distilled プログラミング言語 Python のエッセンス

訳者 鈴木 駿
出版年 2023 年 10 月
出版社 オライリー・ジャパン

Pearson との契約上、邦訳の表紙は動物ではない。

翻訳の流れ

翻訳出版までの軌跡

- 2022 年 4 月 興味本位で原著電子版を購入
- 2022 年 5 月 オライリーの編集者に原著を紹介する（雑談レベル）
- 2022 年 6 月 翻訳版権取得に向けて動き出す
- 2022 年 7 月 翻訳版権取得、翻訳の打診、翻訳に挑戦しようと決意
- 2022 年 9 月 翻訳を開始する（ロバスト Python の監訳と並行）
- 2023 年 4 月 一通り翻訳が完了、推敲の日々
- 2023 年 9 月 翻訳作業完了

Python Distilled ってどんな本？

原著者「はじめに」より

この『Python Distilled』は Python によるプログラミングについての書籍です。Python で可能なことや、あるいは行われたことをすべて文書化しようというわけではありません。本書の目的は、現代的であり厳選、つまり蒸留（distilled）されたプログラミング言語 Python の核心を紹介することです。（中略）しかし、それはまた、ソフトウェアライブラリを書き、Python の何たるかを知り、何が最も役に立つかを見出した結果でもあるのです。

つまり、どんな本？

一言でまとめると

プログラミング言語 Python そのものに特化した本

書評

Twitter / X で見かけた書評

No human should be allowed to write Python code before reading it.

渋川さんによる書評

着実に Python を自らの血肉にしていきたい人向けの本です。

君たちはどう Python を学ぶか

Python の学び方

https://docs.python.org/ja/3/

Python の学び方

Python は公式ドキュメントが充実

- https://docs.python.org/ja/3/

アレはどこに書いてあるの？

そうそう、アレだよ、アレ、あそこにあるよ。

発表者は親の影響で中日ファンでしたが、最近はまったく野球を見ていません。

突然のクイズ

Python のアレ、どこに書いてあるかなクイズ！

第 1 問

問題：シングルトンの比較

`None` や `True`、`False` はシングルトンであり、シングルトンは `is` 文で比較します。
この注意事項はドキュメントのどこに書かれているでしょうか？

第 1 問

問題：シングルトンの比較

`None` や `True`、`False` はシングルトンであり、シングルトンは `is` 文で比較します。
この注意事項はドキュメントのどこに書かれているでしょうか？

解答：2 箇所

- 言語仕様（注意喚起）
- PEP 8（簡単な理由も）

実質的には PEP 8 だけとも言える。

第 2 問

問題：関数のデフォルト引数

関数のデフォルト引数を使う際はイミュータブルなオブジェクトを使います。この注意事項はドキュメントのどこに書かれているでしょうか？

第 2 問

問題：関数のデフォルト引数

関数のデフォルト引数を使う際はイミュータブルなオブジェクトを使います。この注意事項はドキュメントのどこに書かれているでしょうか？

解答：2 箇所

- チュートリアル（注意喚起）
- プログラミング FAQ（デフォルト引数の仕組みについて）

第 3 問

問題：with 文

Python 2.5 から with 文が導入されました。with 文の使い方はどこに書かれているでしょうか？

第 3 問

問題：with 文

Python 2.5 から with 文が導入されました。with 文の使い方はどこに書かれているでしょうか？

解答：3 箇所

- チュートリアル（存在を示唆するだけ）
- 言語リファレンス（with 文の構文とコンテキストマネージャについて）
- PEP 343（with の導入経緯や背景について）

第 4 問

問題：__init__() と __new__()

クラスのインスタンスを実際に生成するのは__new__()、インスタンスの初期化は__init__() です。この関係について書かれているのはどこでしょうか？

第 4 問

問題：__init__() と __new__()

クラスのインスタンスを実際に生成するのは__new__()、インスタンスの初期化は__init__() です。この関係について書かれているのはどこでしょうか？

解答：1 箇所

- 言語リファレンス

__init__() はコンストラクタじゃないよ！

第 5 問

問題：from module import *

from module import *が可能なのはモジュールレベルのインポートで、クラスや関数内部ではできません。この事実について書かれているのはどこでしょうか？

第 5 問

問題：from module import *

from module import *が可能なのはモジュールレベルのインポートで、クラスや関数内部ではできません。この事実について書かれているのはどこでしょうか？

解答：1 箇所

- 言語リファレンス

ただし、from module import *は使うなと注意喚起されている

Python の学び方

Python は公式ドキュメントが充実

- 大体公式ドキュメントや PEP に書かれている
- チュートリアルと標準ライブラリだけで何とかなる

公式ドキュメントは膨大すぎる

- 突っ込んだ内容だと言語リファレンスや PEP を探ることになる
- 言語リファレンスは「そっけない書き方」、読み物的に読めない。

Python の学び方

書籍から学ぶ：入門書

- 基本的には初心者向き、突っ込んだ内容には触れていない
- 筆者の工夫として取捨選択が行われている

入門書の具体例

- 『入門 Python 3 第 2 版』(800 ページ)、入門部分は 270 ページ
- 『Python チュートリアル 第 4 版』(264 ページ)、底本は公式ドキュメント

Python の学び方

書籍から学ぶ：(比較的) 高いレベルの本

- Python 言語を網羅しようとしている
- 必然的に分厚くなり、読み通すのが大変

高いレベルの具体例

- 『初めての Python 第 3 版』(808 ページ)、原書第 5 版は 1648 ページ
- 『Fluent Python』(832 ページ)、原書第 2 版は 983 ページ

Python の学び方

インターネット、または検索で探す

- 玉石混合
- 結局は公式ドキュメントに落ち着く
- 調べたいことがわかっていないと使えない

品質が高い Web 資料

- 『Python Boot Camp Text』初心者向けチュートリアルイベントの資料
- 『Python Tutor』Python の動きを視覚的に確認できるサイト

Python の学び方

最近の流行り：GPTs に聞いてみる

- 俺たちの ChatGPT 先生
- 調べたいことがわかっていないと使えない
- たたき台として最適
- ある程度 Python をわかっていないと使いこなせない（私見）

エレミヤ 14:14（聖書協会共同訳より）

主は私に言われた。「預言者たちは、私の名によって偽りの預言をしている。私は彼らを遣わしたこともなく、彼らに命じたこともなく、彼らに語ったこともない。彼らは偽りの幻と空しい占いと自分の心の欺きをあなたがたに預言しているのだ。

Python の学び方

Python Distilled

- Python 言語そのものに特化した本
- 言語リファレンスに書いてあることが 336 ページにまとまっている

Python Distilled に書いていないこと

- 型ヒント周り（『ロバスト Python』読もう）
- 非同期処理
- 3rd パーティライブラリ、エコシステム周り

FAQ

本当によくある質問

『Python Distilled』と『入門 Python 3 第 2 版』の違いは？

Python Distilled に書いていないこと

- 『入門 Python 3 第 2 版』はエコシステムもしっかり触れている
- 『Python Distilled』は言語コアに特化した本

結局、何が書いてあるの？

1 章 Python の基礎

- 変数やデータ型、式、制御構造、関数、クラス、入出力についての概説
- 原書は Python 3.8 以降を想定、翻訳では 3.11 まで対応できるようにした
- わかる人は飛ばしても大丈夫

例：スタックベースの計算機

大半の場合、継承は最良の解決策ではありません。例えば、単純なスタックベースの計算機を作りたいとします。

翻訳時に抱いた懸念

- 「スタックベースの計算機」は説明なしで通じるのか
- 逆ポーランド記法の説明を追加するべきか

2 章 演算子、式、データ操作

- 式、演算子、評価規則について
- 基本的なデータ構造についても説明

暗黙的な真偽値評価の罠

```
def f(x, items=None):
    if not items:
        items = []
    items.append(x)
    return items
```

実行例

```
>>> f(4)
[4]
>>> a = []
>>> f(3, a)
[3]
>>> a  # 更新されない！
[]
```

3 章 プログラムの構造と制御構造

- 条件分岐、ループ、例外、コンテキストマネージャについて
- 「3.4 例外」は例外なく読もう！

例：例外値を保持する変数の生存範囲

```
try:
    int("N/A")
except ValueError as e:
    print("Failed:", e)
print(e)  # NameError
```

例：例外の連鎖

```
try:
    x = int("N/A")
except Exception as e:
    raise ApplicationError("It failed") from e
```

例：想定外の連鎖

```
try:
    x = int("N/A")
except Exception as e:
    print("It failed:", err)  # NameError
```

__cause__属性と__context__属性

- __cause__属性は意図して例外を連鎖した時に参照する
- __context__属性は例外処理中の想定外の例外発生時の情報源

4 章 オブジェクト、型、プロトコル

- Python の基本的なオブジェクトモデルとメカニズムについて
- PyCon APAC 2023 の発表の元ネタの 1 つ

例：参照カウント

```
>>> a = 37   # 値 37 を持つオブジェクトを作成する
>>> b = a   # オブジェクトの参照カウント増加
>>> c = []
>>> c.append(b)   # オブジェクトの参照カウント増加
```

例：参照とコピー

```
>>> a = [1, 2, 3, 4]
>>> b = a  # b は a の参照
>>> b is a
True
>>> b[2] = -100  # b の要素を変更する
>>> a  # a の要素も変更される
[1, 2, -100, 4]
```

例：整数型と浮動小数点数型

```
>>> a = 42
>>> b = 3.7
>>> a.__add__(b)
NotImplemented
>>> b.__radd__(a)
45.7
```

5 章 関数

- 関数定義、適用、スコープ、クロージャ、デコレータ、関数型プログラミング
- 5.16 節と 5.17 節のコールバック関数に関する説明は内容が濃い

例：動的な関数生成

```
def make_init(*names):
    params = ", ".join(names)
    code = f"def __init__(self, {params}):\n"
    for name in names:
        code += f"    self.{name} = {name}\n"
    d = {}
    exec(code, d)
    return d["__init__"]
```

NamedTuple や@dataclass で活用されているテクニック

6 章 ジェネレータ

- ジェネレータの基礎から応用まで
- 「コルーチン」の歴史も

例：ジェネレータの委譲

```
def flatten(items):
    for i in items:
        if isinstance(i, list):
            yield from flatten(i)
        else:
            yield i

a = [1, 2, [3, [4, 5], 6, 7], 8]
for x in flatten(a):
    print(x, end=" ")
```

例：拡張ジェネレータ

```
def receiver():
    print("Ready to receive")
    while True:
        n = yield
        print("Got", n)

>>> r = receiver()
>>> r.send(None)
Ready to receive
>>> r.send("Hello")
Got Hello
>>> r.close()
```

7 章 クラス

- クラスに関する概念をトップダウンで学ぶ
- 到達点は相当マニアックだが、知っていると役に立つこともあるかもしれない。

メソッド解決順序のルール

1. 派生クラスは常に基底クラスよりも先にチェックされる
2. 基底クラスが複数ある場合は、継承した順にチェックされる

例：C 3線型化アルゴリズムで決定できない例

```
class X:
    pass

class Y(X):
    pass

class Z(X, Y):
    pass  # TypeError
```

8 章 モジュールとパッケージ

- モジュールの概念が焦点
- パッケージングについては触れていない

import module と from module import func の違い

- import module は新たに名前空間を生成する
- from module import func は実行された名前空間に func を追加する

from module import func の方が速い？

- 気のせいです。
- Python が裏側で import module をするので、関係ありません。

9 章 入力と出力

- I/O 処理に必要なテクニックと抽象化
- 後半の I/O ライブラリの概要は流し読みで OK

例：open() 関数の裏側

```
import io

raw = io.FileIO("filename.txt", "r")  # 生バイナリモード
buffered = io.BufferedReader(raw)  # バッファ付きバイナリ読み込み
file = io.TextIOWrapper(buffered, encoding="utf-8")  # テキストモード
```

10 章 組み込み関数と標準ライブラリ

- 組み込み関数と標準ライブラリ、組み込み例外の一覧
- 流し読みで OK

まとめ

さっそく購入しよう

- https://www.oreilly.co.jp/books/9784814400461/
- https://www.ohmsha.co.jp/book/9784814400461/

オライリー学習プラットフォームとは

- https://www.oreilly.co.jp/online-learning/
- 6 万冊以上の書籍と 3 万時間以上の動画（日本語もある！）
- 業界エキスパートによるライブイベント
- インタラクティブなシナリオとサンドボックスを使った実践的な学習
- 公式認定試験対策資料
- 『Python Distilled』もオライリー学習プラットフォームで読み放題（すごい）