

</>

2023年度未踏IT人材発掘・育成事業

Pythonにトランスパイル可能な 静的型付けプログラミング言語の開発

早稲田大学先進理工学部 物理学科

芝山駿介

近年人気の言語Python、
しかし……



Pythonに対する(やや偏った)人々の声

「Pythonの型がづらい」

「中途半端に動いてしまうからPythonはづらい」

「Pythonの文法がおかしい」「動的型付けに対する憎悪が高まる」

「ライブラリによるロックインがなければ.....」

「Pythonはカス」

「Pythonの型ヒントは役に立たない」

「Pythonの環境構築がづらい」

大別すると……

「Pythonの型がづらい」

「Pythonの文法がおかしい」

「Pythonの環境構築がづらい」

「Pythonの型がづらい」

「Pythonの文法がおかしい」

「Pythonの環境構築がづらい」

Pythonの欠点

動的型付け

動的型付け≒実行時にコードの正当性を検証する
→ (静的型付けに比べて)バグの発見が遅れる
=コード品質に影響する？ [1] [2]

[1] http://www.washi.cs.waseda.ac.jp/wp-content/uploads/2016/11/HASE_2017_paper_25.pdf

[2] <https://danluu.com/empirical-pl/>

動的型付けが厄介なバグを引き起こした例

Qiita

記事、質問を検索

ホームタイムライントレンド質問公式イベント公式コラム募集Organization

♡
1222

🔖
413

✕

f

B!

...

@bohemian916 (yota)

祖母が就寝するとDBインサートができなくなる

RaspberryPiinfluxdbgrafana照度センサー

投稿日 2024年01月07日

世の中には、一見関係なさそうな物理現象がITシステムに不可思議な影響を及ぼすことがあります

例えば、500マイル以上離れた場所にメールが送れないという話だったり
中国人のAさんがお茶を入れると会社のネットが繋がらなくなる
という話があります。

私の場合は、祖母が就寝するとDBインサートが失敗する、という状況でした

実家の見守りシステム

データが来てない時間がある

原因を考える

照度が関係してる？

実家に帰って答え合わせ

まとめ

類似例

余談

Ref: <https://qiita.com/bohemian916/items/46f9f1e8bb32fc0a1f99>

7

動的型付けが厄介なバグを引き起こした例

バグの原因

- センサーが観測した照度が0の時だけ変な数値を取得してしまう

バグの原因箇所^[1]

```
182
183 > def calculate_lux(self):
204     ... self.illuminance = round(max([0, lux1, lux2]), 1)
```

↓ float (DBが受け付ける型)

× ↑ int (0.0にすべき)

意図せず型が混ざってしまっている 🤢

[1] <https://github.com/IndoorCorgi/cgsensor/blob/865c074cd905ca8edfd35dea7dc10fcf1b9c231e/cgsensor/tsl2572.py#L204>

(※静的型付けでも型の混合は可能だが、意図されない場合当然コンパイルエラー)

「Pythonの型がづらい」

「Pythonの文法がおかしい」

「Pythonの環境構築がづらい」

Pythonの欠点

歪な文法

~~Pythonは初心者にもわかりやすい文法~~
→ 極めて表面的な部分だけ

オブジェクトの参照や変数のスコープに非直感的な部分[1]
があり、バグの温床となっている

[1] <https://github.com/satwikkansal/wtfpython>

歪な文法

Pythonクイズ

c is d, e is fの結果は？

```
Python 3.11.0 (main, Jul 12 2023, 00:27:36)
Type "help", "copyright", "credits" or "license()"
>>> a = 256
>>> b = 256
>>> a is b
True
>>> c = 257
>>> d = 257
>>> c is d
???
```

```
>>> e, f = 257, 257
>>> e is f
???
```

歪な文法

Pythonクイズ

c is d, e is fの結果は？

```
Python 3.11.0 (main, Jul 12 2023, 00:27:36)  
Type "help", "copyright", "credits" or "license()">  
>>> a = 256  
>>> b = 256  
>>> a is b  
True  
>>> c = 257  
>>> d = 257  
>>> c is d  
False  
>>> e, f = 257, 257  
>>> e is f
```

???

歪な文法

Pythonクイズ

c is d, e is fの結果は？

```
Python 3.11.0 (main, Jul 12 2023, 00:27:36)  
Type "help", "copyright", "credits" or "license()" for more  
>>> a = 256  
>>> b = 256  
>>> a is b  
True  
>>> c = 257  
>>> d = 257  
>>> c is d  
False  
>>> e, f = 257, 257  
>>> e is f  
True
```

歪な文法

Pythonクイズ

eの中身は？

```
Python 3.11.0 (main, Jul 12 2023, 00:27:36)  
Type "help", "copyright", "credits" or "license()" for more  
>>> e = 1  
>>> e  
1  
>>> try:  
...     raise Exception()  
... except Exception as e:  
...     pass  
...  
>>> e
```

???

歪な文法

Pythonクイズ

eの中身は？

```
Python 3.11.0 (main, Jul 12 2023, 00:27:36)
Type "help", "copyright", "credits" or "license()"
>>> e = 1
>>> e
1
>>> try:
...     raise Exception()
... except Exception as e:
...     pass
...
>>> e
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'e' is not defined
```

「Pythonの型がづらい」

「Pythonの文法がおかしい」

「Pythonの環境構築がづらい」

Pythonの欠点

開発環境の構築が難しい

Pythonは公式の提供する開発ツールが比較的貧弱
→ サードパーティの開発ツールが乱立している

仮想環境: venv, pyenv, pyenv-virtualenv, pipenv, Anaconda

フォーマッタ: black, autopep8, yapf, autoflake

パッケージマネージャ: pip, poetry, pipenv, Anaconda

Lint: flake8, pylint, Prospector, ruff

→ 無数の組み合わせの中から選定する必要がある
他人のコードを動かす場合.....

</>

Pythonは、 づらい

</>

解決策はあるか？

</>

— Pythonの欠点を克服した 新言語を作る

The logo for the Erg Programming Language is a large, stylized blue letter 'E' with a 3D effect, tilted slightly to the right. It is composed of three parallel blue outlines. The word 'Erg' is written in a large, white, sans-serif font, centered over the top half of the 'E'. Below it, the words 'Programming Language' are written in a smaller, white, sans-serif font, also centered and spanning the width of the 'E' logo.

Erg

Programming Language

新言語Ergのコンセプト

PythonのAPIを直接呼び出せる
(トランスパイルされる)

Q. 欠点があるにもかかわらず
Pythonが未だ人気なのはなぜ？

A. 圧倒的な量のコード資産など

しかしコード資産を流用できる言語
があればそちらに移行できる



PythonのAPIを直接呼び出せる (トランスパイルされる)

Pythonの代替を名乗る
からには、Pythonで
できる事は全てできるべき

→ 標準ライブラリ
+ 著名なPythonパッケージ
(例. NumPy, PyTorch, Pandas)
の型定義を完備

```
importlib: PyModule("importlib.d") = pyimport "importlib"
math: PyModule("math.d.er") = pyimport "math"
random: PyModule("random.d.er") = pyimport "random"
sub: PyModule("subprocess.d.er") = pyimport "subprocess"
sys: PyModule("sys.d.er") = pyimport "sys"
{DateTime: {datetime.DateTime};} = pyimport "datetime"

print! random.choice! seq:= 1..10
print! math.pi
print! importlib.util.MAGIC_NUMBER
_: importlib.machinery.ModuleSpec or NoneTyp... = importlib.util.find_spec "os"
discard obj:= sub.run! args:= ["echo", "hello"], shell := True
sys.exit 111
print! DateTime.max
print! DateTime.today! ()
initial_commit: datetime.DateTime = DateTime year:= 2022, month:= 8, day:= 10
print! initial_commit
```

新言語Ergのコンセプト

高度な静的型システム

静的型付けのメリット

△実行効率


○コードの堅牢性を高める

Ergが持つ特筆すべき型

- 依存型

- ^{ふるい}篩型

など

```
tests > should_err >  dependent.er
1  dic = {"a": 1, "b": 2}
2  print!(dic["c"]) # ERR
3
4  arr = [1, 2, 3]
5  print!(arr[5]) # ERR
6  |
```


高度な静的型システム

numpy.ndarrayの例

```
np: PyModule("numpy.d") = pyimport "numpy"
```

↓依存型

↓shape(要素数)情報を型が持つ

```
a1: numpy.NDArray(Nat, [3, 3]) = np.zeros(shape:= [3, 3])
```

```
_ : numpy.NDArray(Nat, [1, 9]) = a1.reshape(shape:= [1, 9]) # OK
```

```
_ : numpy.NDArray(Nat, _ : {A: Array(Nat, _ : N... = a1.reshape([1, 7]) # ERR
```

実行時検査するしかなかったエラーを静的に検知可能

高度な静的型システム

numpy.ndarray.reshapeの型定義

```
reshape: [T, Old: [Nat; _], S: {A: [Nat; _] | A.prod() == Old.prod()} | (  
    self: .NDArray(T, Old),           ↑篩型           ↑現在のshapeの総積と  
    shape: {S},           ↑依存型           新しいshapeの総積が一致  
) -> .NDArray(T, S)
```

プログラムが満たす条件を型として詳細に記述できる

新言語Ergのコンセプト

開発ツールの統合

- パッケージマネージャ
 - Language Server
 - 仮想環境マネージャ
- などをすべてコマンド一つに統合
→ Go言語などが採用

メリット

- 環境構築が容易
- 高い再現性

```
~ via ▲ v3.23.2 via 🐧 v3.11.0
at 21:54:34 > go
Go is a tool for managing Go source code.
```

Usage:

```
go <command> [arguments]
```

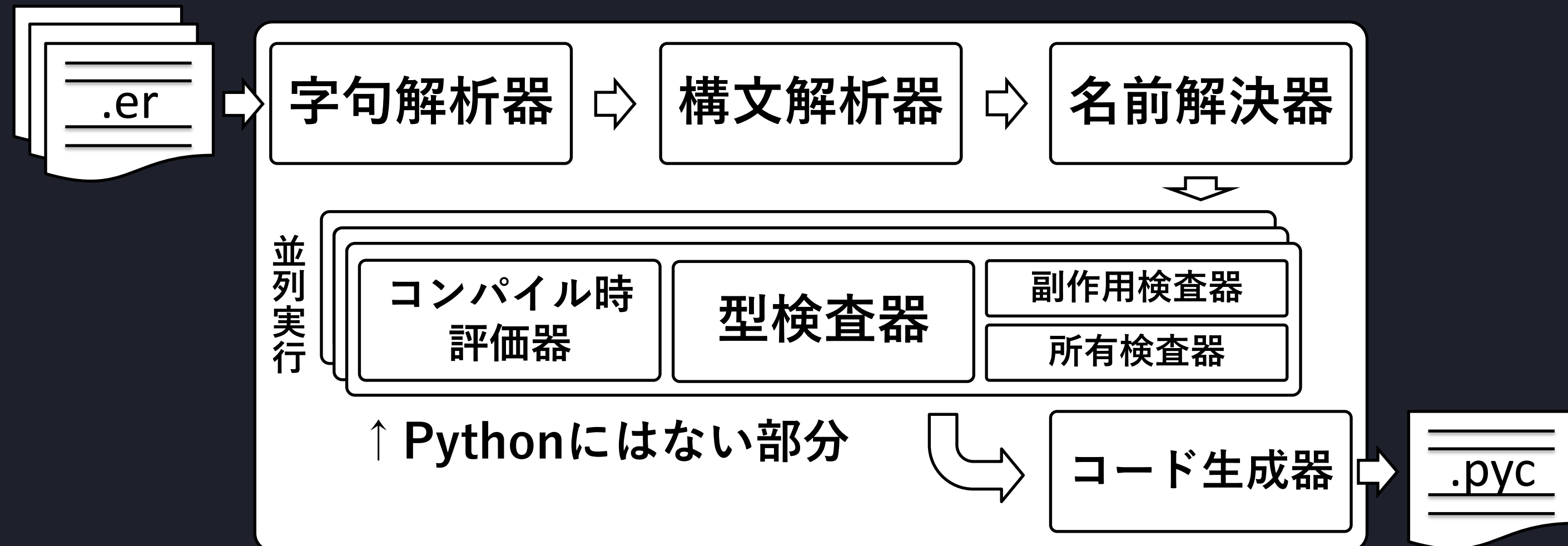
The commands are:

bug	start a bug report
build	compile packages and dependencies
clean	remove object files and cached files
doc	show documentation for package or symbol
env	print Go environment information
fix	update packages to use new APIs
fmt	gofmt (reformat) package sources
generate	generate Go files by processing source
get	add dependencies to current module and install them
install	compile and install packages and dependencies
list	list packages or modules
mod	module maintenance
work	workspace maintenance
run	compile and run Go program
test	test packages
tool	run specified go tool
version	print Go version
vet	report likely mistakes in packages

開発ツール

コンパイラ本体はRustを用いて実装

コンパイラの構成図



開発ツール

Language ServerもRustを用いて実装

- エラー報告
- 補完
- Rename
- 参照の表示
- ホバー
- Inlay hintの表示
などが可能

The screenshot shows an IDE interface. On the left is the 'Explorer' (エクスプローラー) panel showing a project structure with folders 'crates' and 'els', and various files including '.gitignore', 'Cargo.toml', and several '.rs' files. Below the explorer is a sidebar with options like 'CLIPBOARD HISTORY: CLI...', 'アウトライン', 'タイムライン', and 'RUST DEPENDENCIES'. The main editor window shows a file named 'server.rs' with the following code:

```
crates > els > server.rs > {} impl Server<Checker, Parser> > handle_request

751 fn_name!(),
752 );
753 fn start_service
754
755 fn handle_request(&mut self, msg: &Value, id: i64) {
756     match method {
757         "initialize" => self.init(msg, id),
758         "shutdown" => self.shutdown(id),
759         Rename::METHOD => self.rename(msg),
760         Completion::METHOD => self.parse_send::<
761         ResolveCompletionItem::METHOD => self.pa
762         GotoDefinition::METHOD => self.parse_send
763         GotoImplementation::METHOD => self.parse
764         HoverRequest::METHOD => self.parse_send:
765         References::METHOD => self.parse_send::<
```

開発ツール

パッケージマネージャはErg自身を用いて実装

パッケージの

- 初期化
 - 依存関係の更新
 - ビルド・実行
 - テスト
 - レジストリ登録
- などが可能

The screenshot displays the Erg development tool interface. On the left is a file explorer with a tree view showing the project structure. The 'src' directory is expanded, listing files like 'cfg.er', 'download.er', 'runn.er', 'main.er', 'build.er', 'help.er', 'initializ.er', 'install.er', and 'metadata.er'. The 'main.er' file is selected. On the right is a code editor showing the content of 'main.er'. The code defines a 'main!' function that takes a configuration object and performs various actions based on the 'mode' field. The code is as follows:

```
src > main.er > ...
13 {update!: (cfg: cfg.Config) => NoneType;} = import "updat
14
15 main!(): NoneType =
16     cfg: cfg.Config = Config.parse!()
17     match cfg.mode:
18         ("build"): NoneType => build! cfg
19         ("check"): NoneType => check! cfg
20         ("clean"): NoneType => clean! cfg, path:= "."
21         ("help"): NoneType => help! cfg
22         ("init"): NoneType => initialize! cfg, path:= "
23         ("run"): NoneType => run! cfg, "."
24         ("install"): Never => install! cfg, path:= cfg.
25         ("uninstall"): Never => uninstall! cfg, path:=
26         ("publish"): NoneType => publish! cfg
27         ("metadata"): NoneType => metadata! cfg
28         ("test"): NoneType => test! cfg
```


デモ

デモ1 エラー報告

デモ1: エラー報告

デモ2

Erg Language Server





erg



test_completion.er ×

test_completion.er > [?] i

1   を押して、GitHub Copilot Chat に何らかの操作を依頼します。 入力を開始して閉じます。

デモ2: Erg Language Server

デモ3

PyTorchを使用した画像認識
(+パッケージマネージャの紹介)

デモ3: PyTorchを使用した画像認識



まとめ

プログラミング言語Pythonの課題に基づき
その課題を解決する新言語**Erg**を開発した

コンパイラ本体に加えて開発ツール群
(Language Server、パッケージマネージャ、
レジストリなど)を実装した

今後の展望

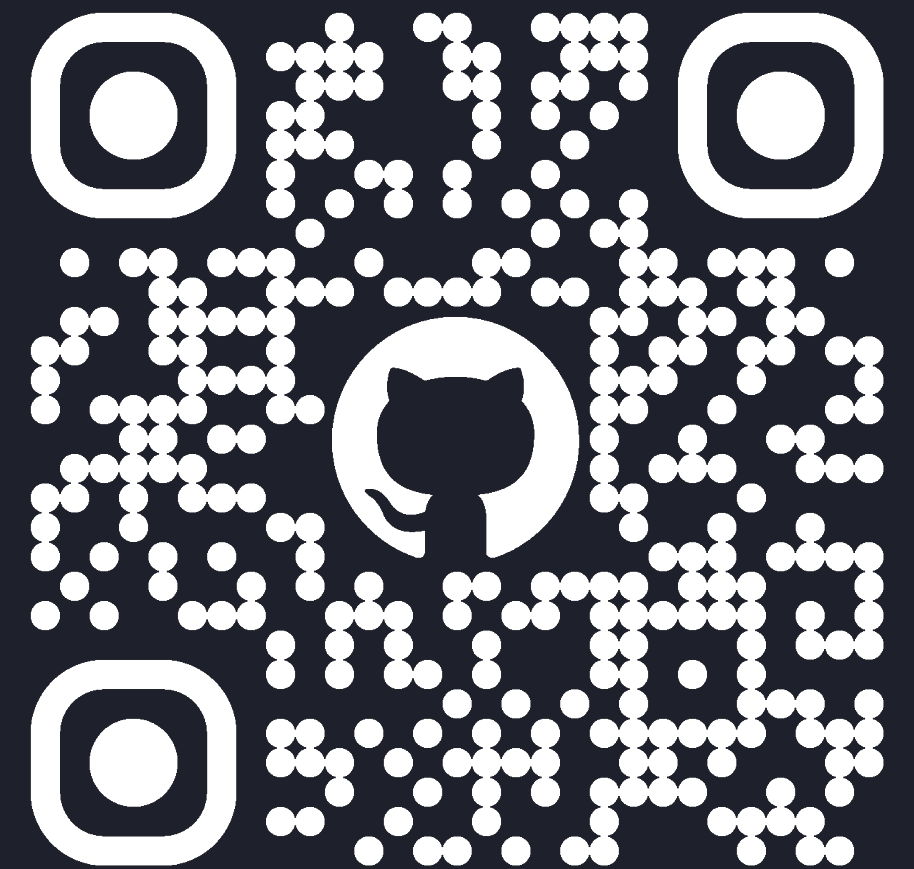
ユーザーを集めたい

- ドキュメントを整備
- コントリビュータを増やす
- 普及活動の実施



____人人人人人人人人人____
> Pythonの牙城を崩す <
____Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^____

↓ Please star it 🙏



<https://github.com/erg-lang/erg>