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 – Erg Programming Language –

1. 背景

Python は，産業用・教育用途を問わず世界中で幅広く利用されている汎用プログラミング言語である．特に，機械学習やデータ分析の分野での利用が増加しており，その需要は今後も続くと予想される．

しかし，Python は実行時にコードの正当性を検証する動的型付け言語であるため，静的型付け言語に比べてバグの発見が遅れ，開発効率が低下するという傾向がある． また，Python はそのシェアの大きさにも関わらず，公式の提供する開発環境が不十分であるという問題もある．

動的型付け言語のデメリットを解消するべく，動的型付け言語に静的型システムを導入したプログラミング言語を開発するというプロジェクトがある．代表的なものが TypeScript である．TypeScript は近年大きな注目を集めており，Python に対して静的型システムを導入したプログラミング言語を開発することも潜在的に大きな需要があると考えられる．

Python をトランスパイル（変換）ターゲットとするプログラミング言語はいくつか存在するが，その多くは動的型付け言語であり，Python の問題をそのまま引き継いでいる．Python との互換性を謳う静的型付け言語もあるが，Python API の静的型付け化を指向しているとは言い難い．

Python には型アノーテーションの文法があるため，これを用いて擬似的に静的型付けを行おうとするプロジェクトもある．しかし，Python 自体が静的解析を考慮していない設計のため，このアプローチにも根本的な限界がある．

2. 目的

前章上で説明した問題を解決するために，本プロジェクトでは 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 Erg をおよびその開発ツール群を開発する．Erg 言語は Python にトランスパイルされるため Python のコード資産をそのまま再利用でき，また静的型システムによる高い静的検証能力を持つ．また開発ツール群をコマンド一つで呼び出せるようにすることで，低い環境構築コストと高い再現性を実現する．

3. 開発内容

Erg は以下の設計思想に基づいて設計されている．

- 強い静的型付け

- ミニマルな基礎文法
- 高い表現能力
- Python API との互換性
- 高い開発効率

以上に基づき，コンパイラ，Language Server，パッケージマネージャ，インストーラ，パッケージレジストリ，パッケージレジストリサイトなどの開発を行った．

3.1. コンパイラ

Erg コンパイラは Rust で実装されており，MIT/Apache 2.0 ライセンスで公開されている．Erg コンパイラは Erg 言語のソースコードを解析・検査し，Python バイトコードに変換する（図１）．

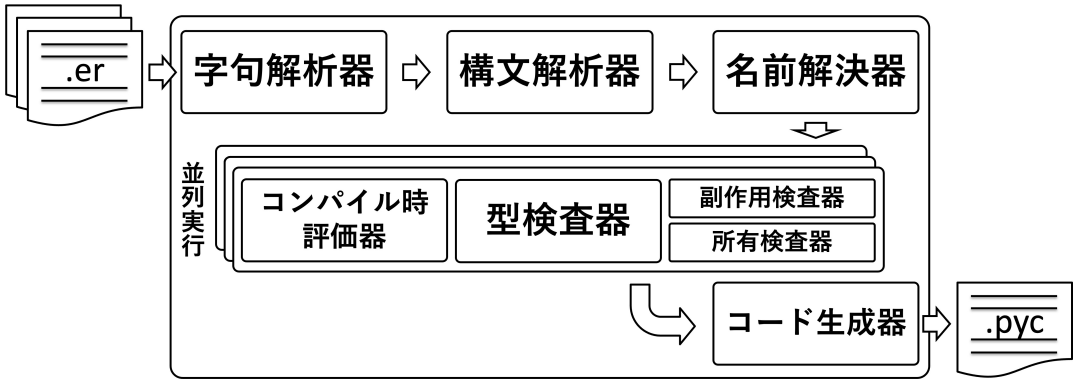

図１：コンパイラの概説図

コンパイルの各ステージで発生したエラーは収集され，一度にユーザーに報告される．エラーは該当箇所のコードとともに提示され，エラーメッセージもユーザーにとって理解しやすいものとなるよう努めた．

3.2. Python パッケージとしての Erg コンパイラ

Erg コンパイラは，Python パッケージとして Python インタプリタが動的に呼び出すこともできるようにした． これにより，Erg モジュールを実行時に読み込んだり，Erg コードの AST を直接操作したりすることが可能になる．API の型定義も行ったので，Erg から Erg コンパイラを操作することも可能である．

3.3. Language Server

Language Server（Erg Language Server）も Rust で実装されており，コンパイラと同じライセンスで公開されている．Language Server とは，対象言語のコーディングを支援する機能（エラー表示，補完機能，定義ジャンプなど）をエディタに提供するためのソフトウェアである．Language Server Protocol（LSP）と呼

ばれるプロトコルに対応することでエディタとの通信を可能にする．

3.4. パッケージマネージャ

パッケージマネージャ（開発コードネーム：poise）は Erg 自身を用いて実装し，コンパイラと同じライセンスで公開されている．パッケージの作成・ビルド・テスト・公開・インストールなどパッケージ開発のライフサイクル全般を管理することができる．

3.5. インストーラ

インストーラは，Erg 言語のコンパイラ，パッケージマネージャ等を一括でインストールするためのスクリプトである．インストール先コンピュータのアーキテクチャに適合するコンパイラのバイナリと標準ライブラリを GitHub release から取得する．また，パッケージマネージャは，ソースコードを取得しコンパイルすることでインストールする．

3.6. パッケージレジストリ

パッケージレジストリはコードホスティングサイト GitHub を用いて構築した．パッケージマネージャは，パッケージレジストリのリポジトリへ圧縮されたパッケージを Pull Request として送信し，メンテナの確認ののち merge することでパッケージの登録を行う．

3.7. パッケージレジストリサイト

パッケージレジストリに登録されたパッケージの情報を閲覧するためのウェブサイトを構築した（図２）．この Web サイトは https://package.erg-lang.org で公開されており，パッケージの検索やメタデータの閲覧が可能である．

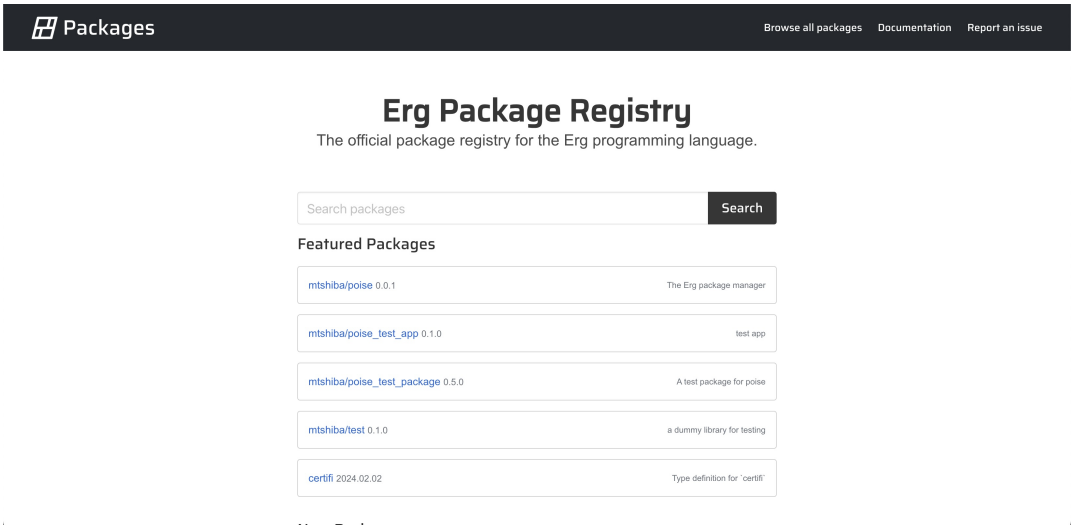

図２：パッケージレジストリサイトのスクリーンショット

3.8. 著名な Python パッケージの型定義パッケージ群

Erg コンパイラは Python の組み込み API や標準ライブラリの型定義を同梱して

いるが， サードパーティの Python パッケージの型定義は Erg パッケージとしてユーザーにオンデマンドで提供されるか，ユーザー自身で型定義ファイルを記述する方式になっている．

このうち著名なサードパーティの Python パッケージについては， 型定義パッケージを用意しレジストリに登録した．これらの型定義は pytypes というリポジトリで管理されている．

4. 従来の技術（または機能）との相違

Erg は Python にトランスパイル可能・あるいは互換性を持つとされる他の言語と比べ， 完全に静的型付けであり，Python API の型定義を自前で行っている点が特徴である．また，Erg の強力な型システムにより，Python API はその使用感をほとんど損なわない形で型定義されている．先行する Hy や Coconut，Julia などの言語は動的型付けであり， 本プロジェクトの解決したい問題に応えるものではない．Mojo は 2023 年に発表された Python との互換性を謳う静的型付け言語であるが，実際の API は Mojo 独自のものが多く，また開発環境も整備されていない．

本プロジェクトではコンパイラ本体に加えて，Language Server， パッケージマネージャ，インストーラ，パッケージレジストリ，パッケージレジストリサイトなどの周辺ツールも開発した．Erg 言語は単なる概念実証ではなく， 実用言語としての基盤を持っていると言える．

5. 期待される効果

ソフトウェア開発において Python の代わりに Erg を用いることで，Python のコード資産をそのまま再利用しつつ， 静的型付けによるバグの早期発見， そして Language Server による豊富なコーディング支援を得ることができる． 標準で付属するパッケージマネージャはパッケージを開発・公開するに必要十分な機能を持っており，開発者は環境の構築に時間を費やすことなく，簡単にパッケージを開発・公開することができる．

6. 普及（または活用）の見通し

Erg コンパイラの GitHub リポジトリは既にスター数が 2500 近くあり，多くのプログラマが興味を持っていることが伺える．Erg 言語はまだまだ発展途上であるものの，本プロジェクトの完遂によって実用言語としての基盤が整ったと言えるので，ユーザー獲得へ向けて広報活動を実施していきたいと考える．Python や Rust などプログラミング言語系のコミュニティの主催するカンファレンスに参加して登壇したり， ある程度 Erg を学習したいというユーザーが集まればワークショップを開催などしたいと考えている．

7. クリエータ名（所属）

芝山駿介（早稲田大学先進理工学部物理学科）

8. 関連 URL

- 公式サイト：https://erg-lang.org
- パッケージレジストリサイト：https://package.erg-lang.org
- GitHub リポジトリ：https://github.com/erg-lang/erg
- 公式ドキュメント：https://erg-lang.org/the-erg-book/