プロジェクト名：Python にトランスパイル可能な静的型付け言語の開発
申請者名：芝山駿介

【提案プロジェクト詳細】

① なにをつくるか

スクリプト言語 Python へトランスパイル(変換)可能な静的型付けプログラミング言語「Erg」・及びその開発支援ツールを開発する。Python は産業用途・学術用途問わず世界中で広く使用されている言語と言える[1]が、実行時にコードの正当性を検査する動的型付けという方式を採用しているため、実行時までバグが発見されない可能性が比較的高いこと、実行性能が低いことなどが問題となっている。動的型付けは、かつては静的型付け(実行前に型検査を行う)よりも簡便と言われ一定のメリットを持っていたが、高度かつプログラマフレンドリーな静的型システムの研究・実用化が進んだ現在、その優位性は薄れている。また Python は公式の開発ツール群がそのシェアに比して貧弱である。結果として代替となるサードパーティのツールがオープンソース・プロプライエタリ問わず乱立し、教育用言語としてのニーズも増大しているにもかかわらず、本格的な環境構築のハードルが比較的高くなっているとされている。これは Python が標榜するモットー、Zen of Python の“Simple is better than complex.”(単純であることは複雑であることよりも優れている)や” There should be one-- and preferably only one --obvious way to do it.”（一つの、できるならばただ一つの、明らかな方法があるべきである）に反する以上に、特に再現性が重要視される学術計算分野に深刻な悪影響を及ぼす。提案者の構想する新言語「Erg」はそれらの問題を全て解決することを目的とする。

Erg は篩型(既存の型に対し更に条件を加える型、「空でない配列型」や「0 以上の整数型」などが指定できる)や依存型(値を情報として持つ型、「要素型 T、長さ N の配列型」などが指定できる。Python は要素型の指定も長さの指定も静的にはできない。多くの静的型付け言語でも長さの指定までは出来ない)など強力な静的型システムを持ち、より多くの潜在的なバグをコード実行前に検出可能である。これらを型システムに搭載するプログラミング言語はまだ少ないものの、定理証明支援系などの形式検証ツールで有用性が実証されてきた。さらに型推論(プログラマがコード中で明示的に型を指定しなくてもコンパイラが自動で推論する機能)などのプログラマフレンドリーな機能を備えており、静的型システムを搭載することによってプログラマの負担が増加するなどといった事態を抑える。Erg コードは Python コードにトランスパイルされるため、Python で実装された任意のコードを利用可能である。それらのコードも Erg の静的型システムが検証するため、Python から Erg へ移行した開発者は、再実装の苦労なく堅牢な開発を実現できる。

動的型付けのプログラミング言語に対し静的型システムを持ち込む試みは、JavaScript に対し静的型システムを追加する TypeScript などの先例がある。TypeScript はプログラマ向けナレッジコミュニティ Stack Overflow が毎年行う開発者調査の 2022 年度「開発者に愛される言語ランキング」で 4 位にランクインする[2]など、大きな成功を収めている。また未踏事業においても、LaTeX の代替を目指す静的型付け組版システム SATySFi や静的型付けスクリプト言語 Cotowali などの例がある。これらの例から、Erg 言語の潜在的ニーズも高いと提案者は考えている。

大まかなコンセプトで言えば、Erg は TypeScript の Python 版を目指す。ただし Erg は TypeScript と

違い、トランスパイル対象言語との文法の互換性に固執しない。Python は 30 年以上前に設計された言語であり、その後文法の変更などが加えられたものの、幾つかのコンセプト、例えば多重継承を使用するオブジェクト指向や貧弱な関数型プログラミング(不変なデータ構造や関数を柔軟に駆使するプログラミングパラダイム。理論的に扱いやすいことや、テストの容易さなどから近年注目を集めている)の支援機能は時代遅れの感を否めないものとなっている。また Python はスコープの扱い(if 文や for 文がスコープを作らない)やオブジェクトの可変性[3]などに非直感的部分があり、これら文法のおかしさを列挙・揶揄する GitHub リポジトリが存在するほどである[4]。これらは言語の基礎的な部分を成しているため、たとえ新たな Python のメジャーバージョンが計画されたとしても修正は困難である。Erg は Python の良い部分を踏襲しながら問題点を修正し、更にトレイトや関数型プログラミングなど他言語(Rust, Haskell, Scala など)で有用性の実証されているコンセプトを導入する。

Erg は静的型システムを持つだけでなく、Python コードを出力するバックエンドの他にネイティブコードを出力するバックエンドも実装予定であり、高速な実行も可能とする(このバイナリは単体で実行も可能であるが、Python インタープリタと容易に連携できるようにする予定である)。また Erg は linter(コードのスタイル的問題を指摘する)・フォーマッタ(コードのスタイルを自動で整理する)・パッケージマネージャ(コードの集成であるパッケージの管理を補助・自動化する)・テストツールなど開発に必要なツールを全てコマンド一つで呼び出せるようにし、低い環境構築難度と高い再現性を保証する。このような統一的アプローチは、他言語では Go 言語などが採用しており、その利便性が実証されている。

本プロジェクトは大規模であるため、未踏事業期間中の完遂、即ちバージョン 1.0 のリリースは目標としない。本事業期間中に実装を予定するのはネイティブコードバックエンド、及びパッケージマネージャなど開発ツール群である。既にコンパイラのプロトタイプは実装しており、Python へのトランスパイルは部分的に成功している。開発ツールは Erg 自体を用いて実装する予定である。これに付随してコンパイラ本体のバグ修正や必要となる機能追加などを行う。開発ツールの実装はコンパイラの実用性を更に高める目的もある。

② どんな出し方を考えているか

オープンソースで公開する。先述の通り既にプロトタイプとなるバージョン 0 は実装済みで、コードホスティングサイト GitHub 上で公開している[5]。コンパイラのライセンスは MIT/Apache 2.0 であり、商用非商用問わず誰でも使用できる。

③ 斬新さの主張、期待される効果など

既に Python との連携やトランスパイルを謳う言語は複数存在する[6][7][8]が、Python の API を静的型付けするタイプで、実用レベルを目指す（学術研究用言語や Toy 言語でない)言語は提案者が確認した限り Erg の他に存在しない。また Python コードに対して静的解析を行う試みは複数存在する[9][10]が、Python 公式の提供するものではなく、また根本的に Python の文法・API 自体が静的解析に向いていないため、問題の解決には至っていない。

本言語が実用可能な段階に達した場合、Python から Erg に移行した開発者の開発者体験は大幅に向上すると期待される。Erg は既に Language Server（コード検査やコード補完などの機能をエディタに提供する開発支援ツール。一般に静的型付け言語の方がコンパイル時に多くの情報を得られるため、より高度な機能を提供できる）を実装しており、開発初期ながらその体験の良さは提案者自身が感じているところである。また、ネイティブコードバックエンドが完成した暁には、これまでパフォーマンスが要求さ

れるにも関わらず Python を使わざるを得なかった分野で大きな効率化が見込まれる。

④　具体的な進め方と予算

開発は提案者の所持するデスクトップ PC 及びノート PC を用いて進める。よって、開発の場所は特に指定しないが、自宅での開発が主になると考えられる。デスクトップ PC は、OS が Windows 11、CPU は AMD Ryzen 9 5900X、32GB RAM、512GB SSD+4TB HDD である。ノート PC は MacBook(OS: macOS Monterey、1.1GHz デュアルコア Intel Core m3、8GB RAM、256GB SSD)と Surface Pro 6(OS: Windows 11、Intel Core i5-8250U、 8 GB RAM、256GB SSD)である。また、Windows 内で WSL という Linux を実行できるソフトウェアを使用する。これにより、Windows/macOS/Linux と主要なプラットフォーム上でのコードの検証を行い、マルチプラットフォームのサポートを実現する。

コンパイラ本体の実装に使用する言語は Rust である。その他、開発を補助するスクリプトとして Python、シェルスクリプト、及び Erg 自体を使用する。エディタは VSCode を主に使用する。また、ソースコード管理のため git 及び GitHub を使用する。開発手法は主にトランクベース開発を採用し、既に構築されたテスト基盤による安定性を生かしてメインブランチに次々とコードを追加・修正することで迅速な開発イテレーションを目指す。

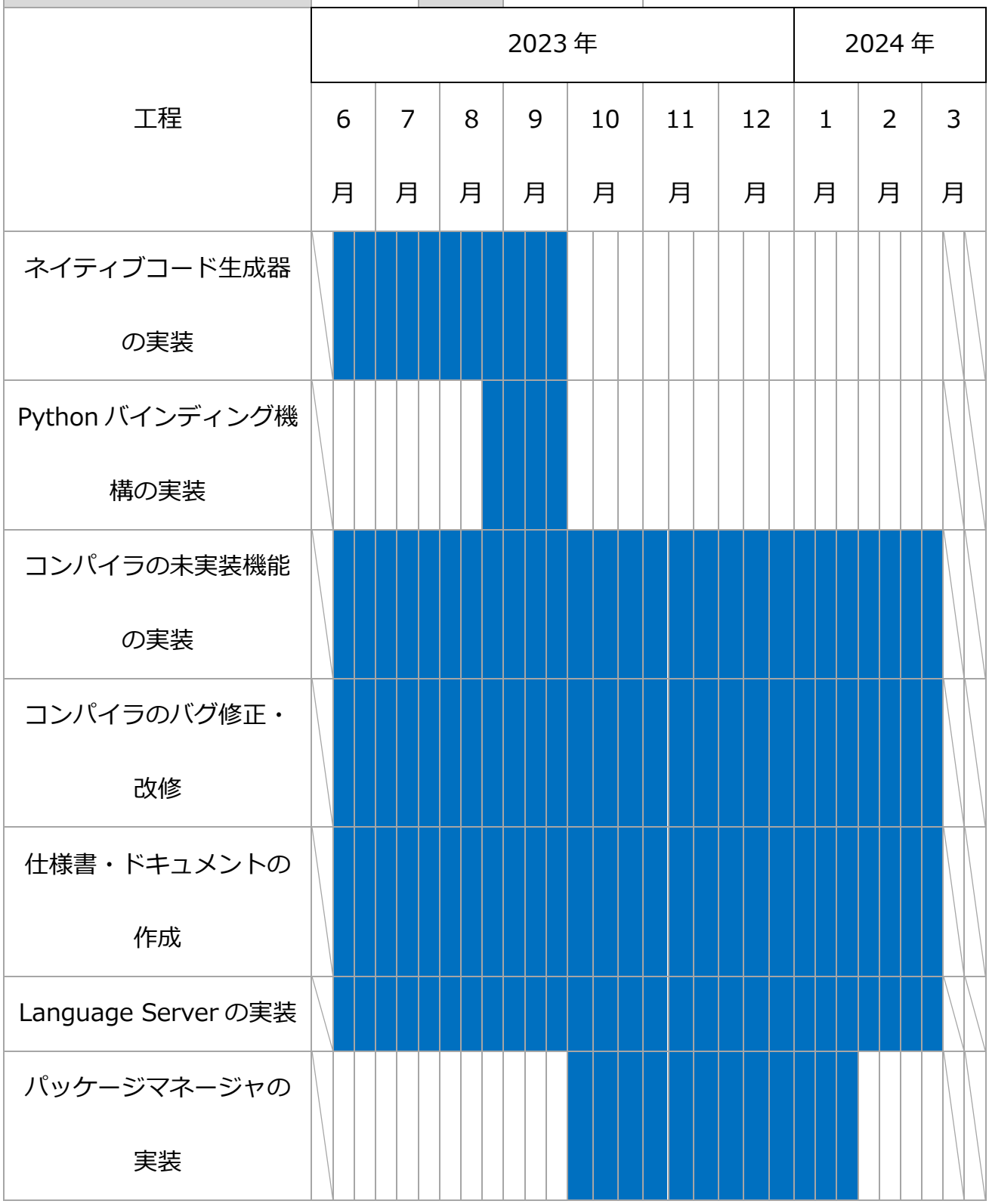
開発は提案者一人で行うが、Erg は既にオープンソースプロジェクトとして公開されているため、外部からの(本事業に直接関連しない)コントリビューションも受け付ける。

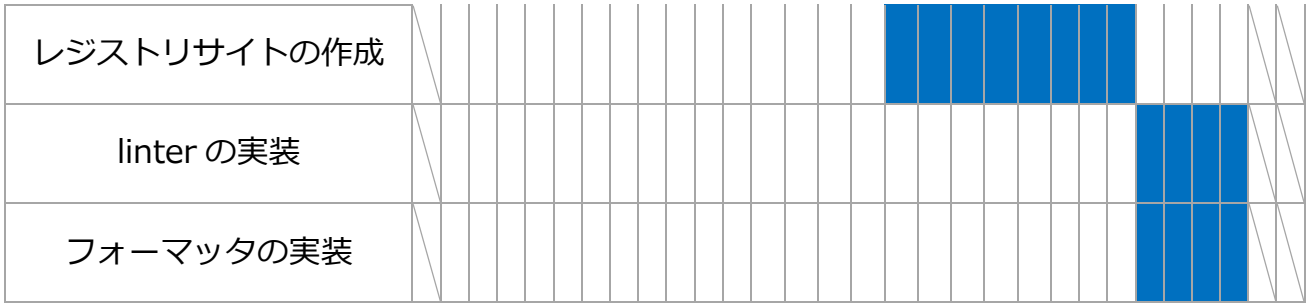
開発は主に大学の講義・ゼミのない曜日、および休暇中に行う。自宅で主に開発するため、時間帯は特に指定しないが、一日の作業時間は最低 4 時間、最高で 12 時間程度とする予定である。

具体的な工程表を以下に示す。青で塗りつぶされている期間が、各工程の開発を行う期間である。各工程の順序は変わらないと考えられるものの、進行状況に応じて前後したり、期間が短縮・延長される可能性がある。

未踏事業工程表

作成者	芝山駿介	作成日	2023/3/8





具体的な作業時間は以下を予定する。これも進行状況に応じて変更となる可能性がある。各ソフトウェアを実装する上で必要となるテストの実装や継続的インテグレーションの整備などはこの作業時間に含むとする。

- ネイティブコードバックエンドの実装: 480 時間

うち、

  - コード生成器の実装: 360 時間
  - Python とのバインディング機構の実装: 120 時間

- コンパイラの未実装機能の実装・バグ修正・改修: 440 時間
- 開発ツール群の実装: 480 時間

うち、

  - Language Server の実装・改修: 120 時間
  - パッケージマネージャの実装・レジストリサイトの作成: 240 時間
  - Linter・フォーマッタの実装: 120 時間

- 各ソフトウェアの仕様書・ドキュメントの作成: 40 時間

予算は 2,736,000 円を申請する。内訳は以下の通りである(時給 1,900 円として作業時間から計上した)。

予算内訳

工程	予算(円)
コード生成器の実装	684,000
Python とのバインディング機構の実装	228,000
コンパイラの未実装機能の実装・バグ修正・改修	836,000
Language Server の実装・改修	228,000
パッケージマネージャの実装・レジストリサイトの作成	456,000
Linter・フォーマッタの実装	228,000
各ソフトウェアの仕様書・ドキュメントの作成	76,000
合計	2,736,000

⑤　提案者の腕前を証明できるもの

先述の通り、Erg は既にプロトタイプ及び詳細な言語仕様[11]がオープンソースとして公開されてい

る。これは提案者が本計画を実現可能である蓋然性が高いことの証明になると考える。提案者は実装言語である Rust の他、コンパイル言語では C/C++, Nim などの使用経験があり、スクリプト言語では Python や TypeScript/JavaScript などの使用経験がある。成果物は提案者の GitHub[12]上で公開されている。また提案者は Erg コンパイラプロトタイプの GitHub 上での開発を通じて、継続的インテグレーション(CI)やテストに関する知識及び運用方法を習得しているほか、コントリビューターとの協調的開発の経験がある。これは提案者の Erg 開発リポジトリでの記録から確認できると考える。

⑥　プロジェクト遂行にあたっての特記事項

提案者は現在大学(学部)3 年生であり、2023 年度より 4 年生となる予定である。大学は早稲田大学先進理工学部物理学科に所属している。4 年生から早稲田大学先進理工学部物理学科中里研究室に配属される予定である。2024 年度は同大学大学院に推薦入学することを計画している。仕事は現在アルバイトをしているが、未踏事業期間前に退職する予定である。研究室での活動予定は既に告知されており、本事業との両立は可能と判断している。

⑦　ソフトウェア作成以外の勉強、特技、生活、趣味など

提案者は大学の物理学科に所属しており、物理学を専攻している。ソフトウェア作成は高校生からの個人的趣味である。物理学の専攻を通じて Python の学術計算における重要性と問題点に気づき、本提案を構想するきっかけの一つとなった。配属される研究室は量子力学基礎論を専門としており、提案者は量子情報・量子重力理論の研究に関心を持っている。趣味は読書・映画鑑賞などがあり、ブログで技術的な記事や鑑賞した作品の感想などを公開している。

⑧　将来のソフトウェア技術について思うこと・期すること

提案者はオープンソースプロジェクト(OSS)に対する支援が現状不足していると考え、その是正を期待する。オープンソースは今や営利企業及び公的機関の技術的インフラとなっており、その社会的意義は大きい。しかし一部の大規模なプロジェクト以外は資金面などの支援が乏しく、広く使用されているにもかかわらず開発が滞る・あるいは放棄・破壊されてしまうなどといった事例[13]が発生している。開発者が後顧の憂いなく開発に専念し社会貢献できるよう、企業や公的機関はこういったプロジェクトに対し積極的・継続的に支援を行うべきだと考えている。

技術そのものに関して最近注目しているのは、大規模言語モデル(LLM)などの機械学習技術である。機械学習技術については、過去に軽く触れたのみで全くの素人ではあるが、個人的所感としては、これまで人間のみが行えるとされてきた知的作業の多くが、将来的には所謂 AI による補助を受けられたり、或いは完全に代替されたりするだろうと予想している。特にプログラミング作業については GitHub Copilot などの専用 AI が登場しており、提案者も既にそのユーザーの一人である。これらを体験して感じたこととしては、AI はアイデアの提案・列挙において非常に大きな力を発揮するものの、複雑なタスクを正確に行うことが現時点では不得手であるということである。プログラミング言語の観点から AI をみると、まだコンパイラを代替することは出来ない(計算量的にも効率が悪い)ものの、エラーに対する修正案の提案、すなわち linter の発展形として言語エコシステムに組み込まれれば、大きな効果を発揮するだろう。

また、LLM などを搭載したハードウェアの開発にも人々が注目すべきであると考えている。LLM の出現によって頭脳労働者・創作者の失職が懸念されているが、肉体労働者には今のところあまり関係のない話となっている。肉体労働の中には運送業・看護業など過酷な労働量が問題になっているにもかか

わらず人材不足の悪循環に陥っている業界も少なくない。特に日本は少子高齢化の問題があり、放置しておけばこれから更に悪化するだろう。解放されるべき労働者はそのまま、頭脳労働者ばかり解放もとい失職という最悪の未来も考えられる。これを防ぐためには肉体労働者を補う産業用ロボットの開発が不可欠と考える。ロボット開発といえば精密機械に強い日本の得意分野にも思えるが、恐らくこれから必要とされるのは LLM など高度なソフトウェアを搭載しハードウェアは簡素かつ堅牢なロボットであろう。肉体労働者の労働は過酷であり、精密な機構を持つロボットでは耐久性に難があるからである。量産の問題もある。となるとその代わりはソフトウェアが担うしかない。これには AI だけでなくドメイン特化型の高度なソフトウェアを設計・実装できる IT 人材が必要とされるだろう。OS やリアルタイムシステムなど基盤技術の重要性も高い。高信頼であることを要求されるから、高度な形式検証技術など理論の援用も必要になるかもしれない。

以上、後半については全くの門外漢の素人考えであり、本プロジェクトの提案内容ともあまり関係ないが、個人的に強く思うところを述べた次第である。

参考:

各リンクは 2023 年 3 月 8 日にアクセス可能であることを確認した。

[1] https://www.tiobe.com/tiobe-index/

[2]https://survey.stackoverflow.co/2022/#section-most-loved-dreaded-and-wanted-programming-scripting-and-markup-languages

[3] https://chokkan.github.io/python/06list.html#id14

[4] https://github.com/satwikkansal/wtfpython

[5] https://github.com/erg-lang/erg

[6] http://hylang.org/

[7] http://coconut-lang.org/

[8] https://julialang.org/

[9] https://github.com/microsoft/pyright

[10] https://github.com/google/pytype

[11] https://erg-lang.org/the-erg-book/

[12] https://github.com/mtshiba

[13] https://www.itmedia.co.jp/news/articles/2201/11/news160.html