プログラミング入門教材と実プロジェクトの間にある文法要素の隔たりの実証分析

福島 和希　嶋利 一真　石尾 隆　松本 健一

プログラミング入門教材では，限られた授業時間に合わせて教える文法要素を選ぶ必要がある．しかし，その選択が実プロジェクトのソースコードで使われる文法要素をどの程度カバーしているかは明らかでない．本研究では，2 大学の教材と公式チュートリアル，4 ドメイン 118 の Python プロジェクトを対象に，出現する予約語・演算子，コードの理解・記述に要する習熟度を 6 段階で評価するツール pycefr が分類する構文的な用法，そのレベルの分布の 3 観点から文法要素の隔たりを分析した．その結果，条件式と相対インポートは教材でほとんど扱われない一方，実プロジェクトの大部分で使われていた．いずれも教材に現れる予約語を用いる構文であり，入門教材と実プロジェクトの間には，予約語や演算子の出現の違いにとどまらず，それらを用いる構文の違いがあることを示した．

1　はじめに

プログラミングの入門教育は多くの大学のカリキュラムに組み込まれており，Python はその主要な言語の 1 つである [11]．現代のプログラミング言語は多様な機能を持つ一方，授業時間は限られているため，教員は教える文法要素を選択する必要がある [10]．

この選択は，入門の範囲に閉じた問題ではない．学生が授業を終えた後に読み書きするのは，Web アプリケーションやデータ処理，機械学習といった実プロジェクトのソースコードであり，そこで使われる文法要素が授業で教えたものと一致する保証はない．しかし，教材の文法要素の選択が実プロジェクトで使われる文法要素をどの程度カバーしているかは明らかでない．必要とされる文法要素が，学生の目標とするドメインによって異なる可能性もある．本研究では，ドメインを比較の軸に取り，隔たりがドメインを問わず成立するかと，ドメインによって異なる要素は何かの両方を確かめる．

我々の先行研究 [6] では，教科書・演習課題の模範解答・実プロジェクトのソースコードの間で語彙（予約語と演算子の集合）を比較し，プログラミング入門授業で扱う語彙の実用性を評価する手法を提案した．奈良先端科学技術大学院大学（以下 NAIST）の Python 授業と Kaggle プラットフォームに投稿されたデータサイエンスのプログラムを比較した事例研究では，`from`, `None`, `is` などの予約語が，Kaggle プログラムで頻繁に使用されているにもかかわらず授業の課題には含まれていなかった．ただし，この先行研究には 2 つの限界があった．第一に，比較対象が単一のドメイン（データサイエンス）にとどまっていた．第二に，予約語と演算子の有無だけを比較しており，それらがどのような構文の中で使われるかは考慮していなかった．例えば，単純な `for` ループとリスト内包表記はいずれも `for` を使用するが，理解に要する習熟度は異なる．

本研究の目的は，プログラミング入門教材と実プロ

---

* An Empirical Analysis of the Gap in Language Constructs between Introductory Programming Course Materials and Real-World Projects

This is an unrefereed paper. Copyrights belong to the Author(s).

Kazuki Fukushima, Kenichi Matsumoto, 奈良先端科学技術大学院大学 先端科学技術研究科, Graduate School of Science and Technology, Nara Institute of Science and Technology.

Kazumasa Shimari, 和歌山大学 システム工学部, Faculty of Systems Engineering, Wakayama University.

Takashi Ishio, 公立はこだて未来大学 システム情報科学部, School of Systems Information Science, Future University Hakodate.

ジェクトのソースコードの間に，どのような文法要素の隔たりがあるかを明らかにすることである．本稿では，予約語や演算子といった字句と，それらを組み合わせた構文をあわせて文法要素と呼び，教材と実プロジェクトの間にある文法要素の使用の差を隔たりと呼ぶ．本稿で教材がある文法要素を扱う・扱わないと言うときは，教材に含まれるコードにその要素が出現するか否かを指し，授業でその要素が説明されたか否かを意味しない．先行研究の限界に対処するため，比較対象を Golubev ら [8] が公開した Web，Media，Data，ML+DL の 4 ドメイン 118 の Python プロジェクトへ広げ，教材も NAIST 授業に加えて東京大学の公開授業教材「Python プログラミング入門」 [19] と Python 公式チュートリアル [13] を取り込む．その上で，出現する予約語・演算子，条件式（三項演算子）や相対インポートといった pycefr [16] の Class が表す構文的用法，pycefr が構成要素へ付与する A1–C2 のレベル分布の 3 つの観点から隔たりを分析する．予約語・演算子の出現だけでは，同じ字句を用いる構文の出現を判定できないため，字句・構文・全体構成を捉える 3 つの観点を併用する．

本研究が行うのは，この隔たりの記述と，その記述が分析条件の選び方に依存しないことの確認までである．検出した差分を教材で扱うべきかどうかの判断には，難易度や学習順序についての教育的な評価を要するため本稿では行わず，5 節で検証可能な仮説として整理する．

本論文の貢献は以下の 3 点である．

1. NAIST，東京大学，Python 公式チュートリアルの 3 教材ソースと Golubev [8] データセットの 4 ドメイン 118 プロジェクトを同一手続きで比較した，予約語・演算子，構文的用法，レベル分布の 3 観点による文法要素の隔たりの実証分析．単一ドメインの語彙比較にとどまっていた先行研究を，複数教材・複数ドメイン・複数観点の比較へ広げるものである．
2. 予約語・演算子の水準のカバーが構文的用法の水準のカバーを保証しないことの実証．教材に現れる予約語を用いながら教材コードにほとんど出現せず，各ドメインの 7 割以上のプロジェクトに出現する構文的用法として，条件式と相対インポートを特定した．
3. 教材構成，集約方式，閾値，pycefr の分類，標本変動に対する上記結果の依存性の検証．レベル分布の比較が要約指標の選択に依存することもあわせて示す．

解析スクリプト，pycefr への修正，対象コミットの一覧，除外ファイルの一覧，分析パラメータと乱数シード，主要な中間結果は，再現パッケージとして公開している †1．

2 背景と関連研究

2.1 プログラミング教育とスキルギャップ

プログラミング入門教育では，学生の誤概念と困難 [14] や授業設計 [10] が広くレビューされ，Python と Java が主要言語であることが確認されている [11]．Garousi ら [7] は SE 教育と産業ニーズの乖離を体系的にレビューし，構成管理，設計，テストにおいて最大の乖離があることを見出した．これらの研究は授業内容と実用的ニーズの整合性の重要性を示しているが，乖離の分析は構成管理や設計といったスキル・トピックの水準にとどまる．言語のどの文法要素が使われるかという構成要素の水準の分析は，次節で述べるコード分析の研究群で行われているものの，教育側との比較を目的としたものではない．

2.2 ドメイン固有のコード分析

Peng ら [12] は 8 つのアプリケーションドメインにわたる 35 プロジェクトで 22 の Python 言語機能をスキャンし，機能の使用パターンがドメインによって異なることを見出した．同研究はドメインごとの機能使用の実態把握を主な目的としており，教材との比較や習熟度の観点は対象としていない．このほか，実プロジェクトにおけるプログラミングパラダイムの分類 [4]，複雑な言語機能の使用 [20]，関数型構成要素の認識 [21]，Pythonic イディオム [1] が研究されている．これらの研究はいずれも実プロジェクト側の使用実態の把握を目的としており，本研究の実プロジェ

†1 https://github.com/NAIST-SE/JSSST_EduGapAnalysis

クト側の分析もこの研究群に連なる．本研究の新規性は，同じ構成要素の観点を教材側にも適用し，教材と実プロジェクトの間の隔たりを定量化するという比較の組み合わせにある．

2.3 語彙に基づく実用性評価

我々の先行研究 [6] は，1 節で述べたとおり，教科書・模範解答・実プロジェクトの間で予約語と演算子の集合を比較し，プログラミング入門授業で扱う語彙の実用性を評価する手法を提案した．この枠組みでは，教科書と課題の語彙が実プロジェクトの語彙の上位集合であれば，授業は実用的なプログラミングに必要な構成要素を網羅していると見なす．本稿の解析対象である予約語と演算子は，この語彙の定義を踏襲したものである．同研究は，制御構造やイディオムのような構文の使い方への拡張を今後の課題としていた．本研究はこの比較の枠組みを引き継ぎ，構文的用法とレベル分布の 2 観点でこの課題に取り組むが，実用性の評価ではなく，教材と実プロジェクトの間の隔たりの記述に主張を絞る．

2.4 pycefr によるコードのレベル評価と応用

Robles ら [16] は，ソースコード中の Python の構成要素を検出し，その理解・記述に要する習熟度を CEFR の 6 段階で評価するツール pycefr を提案した．CEFR は言語能力を 6 段階（A1, A2: 基礎段階，B1, B2: 自立段階，C1, C2: 熟達段階）で評価する基準である．pycefr は，Python の構成要素をその複雑さに基づいてこれらのレベルにマッピングする．例えば，単純な代入（`x = 1`）は A1，import 文は A2，`__init__`を持つクラス定義は B1，ジェネレータ式は C1 である．Ragkhitwetsagul ら [15] は CEFR ベースのフレームワークを JavaScript に拡張した jscefr を提案した．なお，本稿で構成要素の「レベル」と呼ぶ値は，pycefr が割り当てた CEFR レベルを指す．

Rojpaisarnkit ら [18] は，12 冊の Python 教科書における構成要素の導入順序を pycefr レベルと比較し，リスト・辞書内包表記 (C1/C2 → B1/B2)，`enumerate` / `map` / `zip` (C2 → B2)，relative import (B1 → B2) などの再分類を提案した．続く研究 [17] では対象を 22 冊へ拡大し，教科書間で導入順序の類似性が高いことを報告した．Febriyanti ら [5] は，3 つの PyPI ライブラリにおける pycefr レベルとコード保守性の関係を研究し，高レベルと判定されたコードの大部分は保守性リスクが低いことを発見した．Charatvaraphan ら [2] はオープンソースソフトウェア（以下 OSS）のプロジェクトにおけるレベルの変遷を追跡する PyGress を開発し，大部分のコードが A1–A2 レベルであり，C1–C2 構成要素は特定の貢献者に集中していることを明らかにした．これらの先行研究は，教科書側または実プロジェクト側のいずれか一方に pycefr を適用して分析している．本研究は両者を同一の観点から分析することで，既存研究では明らかにされていない教材と実プロジェクトの間の文法要素の隔たりを定量化することを目指す．

3 研究設計

本研究の目的は，プログラミング入門教材と実プロジェクトのソースコードの間にどのような文法要素の隔たりがあるかを明らかにすることである．この目的を，1 節で述べた 3 つの観点に対応する研究課題（RQ）に分けて設定する．構文的用法の隔たりが中心的な関心であるため，これを RQ1 に置く．

- **RQ1**: 教材コードにほとんど出現せず，実プロジェクトで広く使われている構文的用法はどれか．その結果は教材構成や分析条件によってどう変わるか？
- **RQ2**: 各教材と 4 ドメインの間で，pycefr が付与するレベルの分布はどのように異なるか？
- **RQ3**: 4 ドメインに共通して高頻度に出現する予約語・演算子は何か．各教材はそれをどの程度カバーするか？

分析では，教材と 4 ドメインの Python ソースコードを収集し，構文的用法とレベル分布は pycefr による構成要素検出で（RQ1・RQ2），予約語・演算子は字句解析で測定する（RQ3）．以下では，データ収集の後，共通の前処理である pycefr による構成要素の検出をまとめて述べ，続いて各 RQ の分析手続きを述べる．

3.1 データ収集

3.1.1 授業データ

先行研究 [6] と同一の NAIST 授業データに加え，比較対象として東京大学の公開授業教材と Python 公式チュートリアル [13] を取り込む．教材の選定は体系的サンプリングではなく，実施文脈と模範解答を利用できる NAIST 授業，日本語で公開され章構成と演習解答を持つ東大教材，言語公式の自習教材である Python 公式チュートリアルという，性格の異なる 3 ソースを意図的に選んだものである．対象授業は，情報系以外の出身の大学院生を主な受講者とする 2023 年度の NAIST の Python プログラミング入門授業である．授業では Python 3.9 を使用していた．本研究では，予約語 35 種類と演算子 32 種類の計 67 種類を字句解析の対象とし，これらをあわせてトークンと呼ぶ．Python 3.10 以降で段階的に導入されたソフトキーワード 4 語（`match`, `case`, `type`, `_`）は，字句解析では識別子としての出現（`re.match` や `type()` 等）と区別できないため解析対象から除外する．

表 1 に各教材データをまとめる．NAIST 教科書は授業で使用した教科書 †2 である．東大教材の内訳は本文 2,570 行，解答 406 行，付録 1,351 行である．公式チュートリアルは CPython リポジトリ main ブランチのコミット d63c994（2026 年 6 月 15 日取得）の Doc/tutorial 以下を対象とし，対話例と code-block 系ディレクティブのコード例を抽出した．本文中の字下げのみで示されるコード例（literal block）は抽出に含まれない．本稿では，NAIST，東大，Python 公式チュートリアルを 3 教材ソースと呼ぶ．解析上は NAIST を教科書・模範解答，東大を本文・解答・付録，公式チュートリアルを 1 層として計 6 教材層に分ける．3 ソースは構造が異なる．NAIST の教科書と模範解答は，市販の教科書の掲載コードと，授業の課題に対して教員とティーチングアシスタントが作成した解答という，独立に作られた 2 つのコレクションである．これに対し東大教材は一体の Jupyter notebook 教材であり，解答と付録は本文に付随する層である．公式チュートリアルは自習用の単層である．この構造の違いは，3.3 節の教材シナリオの構成と，3.4 節の比較対象の選び方に反映される．

3.1.2 ドメイン別データ

Golubev ら [8] が公開したデータセットを採用する．表 2 に概要を示す．原データは各ドメイン 30 の計 120 プロジェクトを収録するが，取得時点で `.py` ファイルを確認できた 118 プロジェクトを分析対象とする．プロジェクトは，(1) 他のリポジトリのクローンではない，(2) コミット数が 1,000–30,000，(3) プロジェクト年齢が 2 年以上，(4) 2020 年 7 月時点でコントリビュータが 10 名以上，という 4 つの基準をすべて満たすものとして選定され，Zenodo で公開されている．各プロジェクトのコードは 2026 年 5–6 月に取得した時点の最新コミットを解析対象とし，取得コミットのハッシュは再現パッケージに記録している．

各プロジェクトから `.py` ファイルを抽出し，テスト，ドキュメント生成，設定に関わるファイルを除外する．抽出した `.py` ファイル数は Web 3,946，Media 6,157，Data 5,808，ML+DL 15,670 の合計 31,581 であった．本稿では，これら 118 プロジェクトを実プロジェクトと呼ぶ．

3.2 pycefr による構成要素の検出

4 ドメインと 6 教材層（NAIST 教科書，NAIST 模範解答，東大教材の本文・解答・付録，Python 公式チュートリアル）のすべてのコードコレクションに対して pycefr [16] v1.0.1 を適用し，検出された構成要素に CEFR レベルを付与する．なお，pycefr は一部ノードで属性参照エラーを発生させるため，失敗したノードを記録して走査を継続するよう修正して用いた．解析対象の pycefr インスタンス数は，教材側は層あたり 418–2,073 件，実プロジェクト側はドメインあたり 640,370–3,040,592 件である．教材側と実プロジェクト側で分母の規模が大きく異なるため，比較は各コレクション内で正規化した頻度に基づく．各コレクションについて，全構成要素インスタンスの CEFR レベル分布と，Class ごとの出現頻度を算出する．分析単位は，実プロジェクトではプロジェクト，教材では章とする．pycefr 構成要素が 0 件の単位は含めず，分析単位数は NAIST 教科書 9 章，

†2 `https://sukkiri.jp/books/sukkiri_python`

表 1　教材データ

教材データ	内容	前処理	規模
NAIST 教科書	授業で使用した教科書の掲載コード	—	17 ファイル・2,034 行
NAIST 模範解答	教員とティーチングアシスタントが作成した課題の解答	—	59 ファイル・866 行
東大教材 [19]	公開授業教材の Jupyter notebook	マークダウン見出しを境界に本文・解答・付録の 3 層へ分離し，構文検証が通らない 62 セルを除外	4,327 行
公式チュートリアル [13]	CPython リポジトリのチュートリアル文書のコード例	章単位に整形し，意図的なエラー例を除外	1,154 行

表 2　ドメイン別データセット [8]

ドメイン	内容	分析数/収録数
Web	Web フレームワーク，ダウンロード・アーカイブツール等	30/30
Media	画像・動画エディタ，音楽ライブラリ管理ツール等	30/30
Data	数学・計算ライブラリ，テーブル処理・グラフ描画・データ処理ツール等	29/30
ML+DL	機械学習，自然言語処理，ニューラルネットワーク関連	29/30

NAIST 模範解答 7 章，東大教材本文 22 章，Python 公式チュートリアル 13 章，実プロジェクト側は各ドメイン 29–30 プロジェクトである．

3.3　構文的用法の隔たりの分析 (RQ1)

RQ1 では構文的用法の隔たりを扱う．生の pycefr Class の頻度差には教材構成や集約方法に依存する候補が多数含まれるため，複数の分析条件で共通して残る差だけを拾う絞り込み手続きを用いる．まず，教材の基準レベル L_{course} を定義する．教材データのインスタンスを CEFR レベル A1, A2, B1, B2, C1, C2 の順に累積し，累積割合が閾値 τ（既定 $\tau = 0.80$）に初めて到達するレベルを L_{course} とする．例えば教材インスタンスの 8 割が A2 までに収まるなら $L_{course} = A2$ である．これは教材コード中のレベル分布の分位点であり，授業で説明された範囲，学生の習得，授業の実到達水準を意味しない．L_{course} の次のレベルを $\mathrm{succ}(L_{course})$ で表し（例えば $L_{course} = A2$ なら $\mathrm{succ} = B1$），各構成要素 $c = (\mathrm{Class}, \mathrm{Level})$ について次の量を定義する．

$$\mathrm{freq}_d(c) = \frac{\text{ドメイン } d \text{ での } c \text{ の件数}}{\text{ドメイン } d \text{ の全構成要素件数}}$$

$$\mathrm{freq}_{course}(c) = \frac{\text{教材での } c \text{ の件数}}{\text{教材の全構成要素件数}}$$

$$\mathrm{gap}_d(c) = \mathrm{freq}_d(c) - \mathrm{freq}_{course}(c)$$

以下では，実プロジェクトと教材の頻度の差を表す $\mathrm{gap}_d(c)$ を頻度差と呼ぶ．ドメイン d について，$\mathrm{Level}(c) = \mathrm{succ}(L_{course})$ かつ $\mathrm{gap}_d(c) > 0$ となる構成要素を頻度差の降順（同値時は Class 名昇順）に並べたリストを教材–実プロジェクト差分候補と呼ぶ．教材の基準レベルのすぐ次の段階で，実プロジェクトに教材より高頻度で現れる Class を示すものである．対象を $\mathrm{succ}(L_{course})$ の 1 レベルに絞るのは，教材コードの現状の分布から最も近い拡張候補に対象を限定するためである．pycefr は文脈に応じて同じ Class へ異なる Level を割り当てるため，順位付けは (Class, Level) ペアで行う．

教材構成による結論の頑健性を見るため，授業データを 6 通りに切り替えて同じ手続きを繰り返す．シナリオは，S1 NAIST 教科書 + 模範解答，S2 Python 公式チュートリアル単独，S3 S1 + S2 の合算，S4 東大教材本文単独，S5 東大教材本文 + 解答，S6 S5 + 東大付録，である．6 つのシナリオは，何を教材とみ

なすかの代表的な取り方を表す．S1 は NAIST の授業で学生が触れるコードの一式，S2 は特定の授業によらない標準教材を単独で用いる場合，S3 は両者を合わせて教材の範囲を最も広く取った場合である．S4 から S6 は，本文・解答・付録からなる東大教材の構成に沿って，教材とみなす範囲を段階的に広げたものである．授業教材に標準教材を加える構成は S3 で代表させるため，東大教材とチュートリアルを組み合わせたシナリオは設けない．S3，S5，S6 は他のシナリオと包含関係を持つため，条件の組は同格な独立反復ではない．集約方式は 2 通りとする．プール集計ではドメイン内の全プロジェクトのカウントを合算してから正規化し，プロジェクト等重み集約ではプロジェクト内で正規化した頻度のドメイン内平均を取る．$\tau = 0.80$ では，教材シナリオ 6 通り，集約方式 2 通り，ドメイン 4 通りからなる 48 分析条件を用いる．既定値 $\tau = 0.80$ は教材インスタンスの大半を覆う分位点として置いた便宜的な設定であり，さらに τ を 0.70–0.95 で振って値の選び方への感度を確認する．これらの条件を網羅するのは，条件同士の優劣を比較するためではなく，教材の範囲や集約方式の選び方によらず残る差だけを主張するためである．ドメインは，隔たりがドメインを問わず現れるかを見る比較の軸を兼ねる．観測された章とプロジェクトの構成に結果が依存しないかを確かめるため，各教材の章と各ドメインのプロジェクトを同数だけ無作為に引き直し，L_{course} の算出から頻度差の順位づけまでの全手続きを再計算する再標本化（ブートストラップ）を 2,000 回行う．同一反復では章・プロジェクト標本を全条件で共有し，条件間の依存構造を保つ．単一プロジェクトへの依存の診断として，1 プロジェクトずつ除外する leave-one-out も併記する．教材側についても，教材ファイルを 1 件ずつ除外して L_{course} を再計算する診断を行う．

$\mathrm{gap}_d(c) > 0$ という条件だけでは，関数シグネチャの引数種の組合せに由来する細粒度な複合 Class を含む多数の構成要素が候補に残る．そこで本研究では，共通上位差分要素を「48 分析条件のすべてで教材–実プロジェクト差分候補の上位 10 件に入る pycefr Class」と定義する．この規則は，どの教材構成・集約方式・ドメインでも一貫して上位に入ることを要求するものであり，見逃しを許す代わりに，条件次第で入れ替わる候補を確実に除く．上位件数 10 は候補数を制限する便宜的な設定であり，件数を変えても結果が変わらないかも確認する（4.1 節）．また，一定割合以上の条件で上位に入る候補を採る緩和規則，48 条件の平均順位による集約，単一条件の上位 10 件などの単純な手続きとも比較する．シナリオに含めない教材構成である NAIST 教科書単独，模範解答単独，および東大教材とチュートリアルの合算にも同じ手続きを適用し，結果が変わらないかを確認する．また，pycefr の分類変更に結果が左右されないかを確かめるため，Rojpaisarnkit ら [18] の再分類提案のうち分類先が一意な `Relative From` 系クラスだけを B1 から B2 へ再分類し，同じ手続きを再実行する．提案には分類先が一意でないものもあるため，推測による対応づけは行わない．Class 粒度の影響を見るため，関数シグネチャの複合 Class と `Relative From` 系 Class をそれぞれ 1 つの Class へまとめる集約でも再実行する．頻度差の解釈を補うため，主要な差分要素については，出現するプロジェクトの割合と出現する教材章の割合の差も算出する．信頼できる実行行数を全入力で一貫して得られないため，行数あたりの頻度は用いない．

3.4 レベル分布の隔たりの分析（RQ2）

RQ2 では，章・プロジェクトごとに A1=1, ..., C2=6 と数値化した構成要素の平均レベルを算出し，教材の章平均と実プロジェクトのプロジェクト平均を，順序データ向けの効果量である Cliff's δ [3] で比較する．これはプロジェクト平均が章平均を上回るペアの割合と下回るペアの割合の差であり，本稿では固定された対象集合内の記述量として扱う．同値ペアは 0 として数える．比較対象の教材は NAIST 教科書，NAIST 模範解答，東大教材本文，Python 公式チュートリアルの 4 つとする．東大解答と付録は，3.1 節で述べたとおり本文に付随する補助層であり，単独の教材を表す標本ではないため，RQ2 の独立比較には含めず，RQ1 の感度条件としてのみ用いる．NAIST の模範解答は独立に作られたコレクションであるため，単独

の教材データとして比較に含める．A1=1, ..., C2=6 の平均はレベル間の等間隔性を仮定するため，各単位の B1 以上割合，B2 以上割合，中央値，および単調な非線形スコア $(0, 1, 2, 4, 8, 16)$ の平均も算出する．各指標について，章とプロジェクトを同数だけ無作為に引き直す再標本化を 10,000 回行い，Cliff's δ の 95%パーセンタイル範囲を求める．反復回数が RQ1 と RQ3 の 2,000 回より多いのは，RQ1 と RQ3 の再標本化が反復ごとに候補抽出の全手続きを再計算するのに対し，本指標の再計算は軽量で反復を増やせるためである．これは観測単位の再重み付けに対する安定性の診断であり，母集団への推論ではない．basic 方式による範囲も区間構成法への感度として算出する．

3.5 予約語・演算子の隔たりの分析 (RQ3)

各ドメインのコードから Python の `tokenize` モジュールを用いて字句解析を行い，各トークンの出現を数える．計数はトークン値の照合で行い，文字列リテラルとコメントの内部は数えない．`is not` と `not in` は構成トークンを個別に数え，デコレータと行列積の`@`は対象演算子に含めない．各プロジェクト内では，対象 67 トークンの出現回数合計を分母としてトークン頻度を正規化する．ドメインの主集計はプロジェクト内頻度の等重み平均とし，全プロジェクトのカウントを合算するプール集計を感度分析に用いる．4 ドメインを以下の 2 つの観点で比較する．

第一の観点は，4 ドメインに共通して高頻度に出現するトークンである．4 ドメインそれぞれについて，67 トークンのうち正規化頻度が上位 X%（X は 50, 25, 10）に入る集合を求め，その積集合を共通高頻度トークンと呼ぶ．上位 X%に相当する件数は 67 トークンの X%とし，端数は切り上げる．境界に同頻度のトークンが並ぶ場合はトークン文字列の昇順で確定する．X の 3 水準は便宜的な区切りであり，単一の閾値に依存しないよう複数の水準で結果を示す．各ドメインのプロジェクトを同数だけ無作為に引き直す再標本化を 2,000 回行い，共通高頻度トークンの選択率を求める．各教材データについて，共通高頻度トークンのうち教材に出現する割合をカバー率として算出する．カバー率が 100%を下回るとき，その差は実プロジェクト側で高頻度でありながら教材に出現しないトークン，すなわち予約語・演算子の隔たりを表す．

第二の観点はプロジェクト出現率である．各トークンについて，出現したプロジェクトの割合（プロジェクト出現率）をドメイン間で比較する．

先行研究 [6] は Kaggle を対象とした単一ドメインの語彙比較であり，本研究の多ドメイン分析を動機づける関連研究と位置づける．取得時点，対象コーパス，分析粒度が異なるため，性能ベースラインとしての直接比較は行わない．

4 結果

4.1 RQ1: 構文的用法の隔たり

6 つの教材シナリオすべてで，$\tau = 0.80$ における L_{course} は A2 となり，次のレベル $\mathrm{succ}(L_{course})$ は B1 となった．この結果は $\tau \in [0.70, 0.85]$ の範囲で全シナリオ不変である．

観測データで共通上位差分要素となったのは，`If statements expression (else)`（AST の `IfExp` に対応する条件式 `x if cond else y` 全体）と `Relative From`（相対インポート `from .module import`）の 2 件である．以下ではそれぞれ条件式，相対インポートと呼ぶ．条件式は三項演算子とも呼ばれ，文である if 文とは別の構文である．if 文は AST では `If` ノードとなり，pycefr は `Simple If statements` などの別 Class に分類する．elif 節も構文木上は入れ子の if 文として同様に扱われる．したがって条件式の件数に if 文と elif 節は含まれない．表 3 に，両要素の実プロジェクト側頻度，教材シナリオ 6 通りにわたる頻度差の範囲，出現プロジェクト数をドメイン別に示す．相対インポートは 6 シナリオすべてで教材頻度が 0 であるため，頻度差は実プロジェクト側頻度に一致する．条件式の教材頻度は東大教材を含む S4–S6 でのみ正となるが，最大でも 0.05%であり，頻度差はどのシナリオでも実プロジェクト側頻度とほとんど変わらない．

両要素は，6 つの教材シナリオ・4 ドメインのすべてで頻度差が正であり，現行分類のもとで共通上位差分要素となった．この隔たりは，頻度差そのものは最大でも 0.66 パーセントポイントと小さいものの，

表 3　B1 の共通上位差分要素の実プロジェクト側頻度と頻度差（プール集計，$\tau = 0.80$，単位%）

構成要素	ドメイン	頻度	頻度差	出現数
条件式	Web	0.41	0.36–0.41	29/30
	Media	0.33	0.28–0.33	28/30
	Data	0.39	0.35–0.39	28/29
	ML+DL	0.66	0.61–0.66	29/29
相対インポート	Web	0.40	0.40	25/30
	Media	0.22	0.22	22/30
	Data	0.30	0.30	25/29
	ML+DL	0.28	0.28	22/29

頻度は実プロジェクト側の構成要素頻度．頻度差は教材シナリオ S1–S6 にわたる範囲．出現数は構成要素が出現したプロジェクト数/分析プロジェクト数を表す．

出現率の対比として明瞭である．条件式は東大教材に 1 回出現するだけで他の 5 教材層のコードには出現しない．一方，if 文の Class は全教材層に多数出現しており，この隔たりは if 文ではなく条件式に固有である．これに対し実プロジェクト側では各ドメイン 28–29 プロジェクト（29–30 プロジェクト中）に出現し，プロジェクト出現率は 93.3–100% である．相対インポートは，抽出したどの教材コードにも一度も出現しない唯一の候補である一方，各ドメイン 22–25 プロジェクトに出現し，出現率は 73.3–86.2% である．教材章での出現率は条件式 0–4.5%，相対インポート 0% であり，教材コードにほとんど出現しない構文的用法が，実プロジェクトの大部分のプロジェクトに出現している．

この隔たりは教材側標本の小ささでは説明できない．実プロジェクト側の頻度が教材コードでも成り立つと仮定すると，教材 6 層の計 6,779 インスタンス中に相対インポートは 15–27 件，条件式は 22–45 件の出現が期待されるところ，観測はそれぞれ 0 件と 1 件である．出現率の対比が指標の飽和による見かけでないことも確認した．B1 に分類される全 Class（関数シグネチャの複合 Class を含み，各ドメイン 1,484–4,679 件）のプロジェクト出現率は中央値 3%であり，70% 以上に達する Class は各ドメイン 23 件に限られる．条件式と相対インポートはこの少数の Class に含まれる．ファイル単位の出現率でも，条件式は各ドメインの 20.4–32.2%，相対インポートは 16.8–20.3%のファイルに出現しており，プロジェクト内の少数ファイルに限られる出現ではない．

この隔たりは特定の教材や少数のプロジェクトによる結果ではない．各プロジェクト内で頻度を正規化しプロジェクトを等重みとしてドメイン集約した場合でも，両構成要素の頻度差は 6 シナリオ・4 ドメインのすべてで正のままである．`Relative From` の頻度差の 4 ドメイン平均は，プール集計の 0.30 パーセントポイントに対して 0.48 パーセントポイントとむしろ大きい．1 プロジェクトずつ除外した再計算（leave-one-out）では，どのプロジェクトを除外しても頻度差の符号は変化せず，両要素の最悪順位は 7 位であった．これは単一プロジェクト依存の診断であり，未観測コーパスへの不確実性評価ではない．

以上の隔たりが分析条件の選び方に依存しないことを確認した．3. 3 節の抽出規則を適用すると，48 分析条件の積集合に残る Class は上記 2 件のみであった．シナリオに含めなかった NAIST 教科書単独，模範解答単独，東大教材とチュートリアルの合算のいずれの構成でも，L_{course} は A2 のままであり，条件式と相対インポートは 2 つの集約方式と 4 ドメインのすべてで教材–実プロジェクト差分候補の上位 10 件に入った．章とプロジェクトを引き直して全手続きを再計算する 2,000 回の再標本化では，全 48 条件で頻度差の 95%パーセンタイル範囲の下端が 0 を上回り，1 反復内で 48 条件すべてに残る割合は条件式 96.6%，相対インポート 96.9%であった．上位件数 K への感度では，$K = 6$–10 の範囲で結果は不変であった．$K = 11$ 以上では関数シグネチャの複合 Class が順に加わり，これは生の pycefr Class を教育単位とみなせないことも示す．閾値 τ への感度では，$\tau = 0.70$–0.85 で積集合は上記 2 件のままであった．$\tau = 0.90$ では S2 のみが $L_{course} = B1$ となり，$\tau = 0.95$ では S1 以外の 5 シナリオが B1 となるため，レベルの異なる候補集合を横断する積集合は空となった．抽出規則への感度では，一定割合以上の条件で残る候補を採る緩和規則や平均順位による集約でも，条件式と相対インポートは常に含まれた．単一条件の上位 10 件などの単純な手続きは最大 4 件の関数シグネチャ変種を候補に含み，教材シナリオやドメインを替えると候補の一部が入れ替わった．出現率の観点でも結果は維持された．

教材章と実プロジェクトの出現率差は，全シナリオ・ドメイン比較で再標本化範囲の下端が正であり，主要 2 要素の差は頻度の分母の取り方に依存しなかった．

一方，絞り込みの出力形は pycefr の分類に依存する．相対インポート系の Class だけを B1 から B2 へ再分類すると，$L_{course} = A2$ は変わらないものの，相対インポートは B1 の候補から外れ，B2 の候補の中で各ドメイン 1 位または 2 位に入る，全教材に出現しない要素となる．このときの積集合は条件式と関数シグネチャ変種 1 件となった．また，類似 Class をまとめる集約では積集合は条件式と相対インポートに対応する 2 件となり，関数シグネチャ変種は残らなかった．もっとも，いずれの再分類・集約でも，両要素が教材にほとんど出現せず実プロジェクトに広く出現するという観察自体は変わらない．分類変更が動かすのは候補リストの構成であり，本手続きの役割は複数条件で安定する要素だけを絞り込むことに限られる．

この相対インポートがどの教材にも出現しないという観察は，Rojpaisarnkit ら [18] が 12 冊の Python 教科書を分析して relative import を「教科書にほぼ未登場で，B1 ではなく B2 に再分類すべき」と提案した結果と対応する．本研究は，relative import が対象実プロジェクトで観測される一方，対象教材の抽出コードには一貫して出現しないことを，実プロジェクト側から 3 教材ソースにわたって示した．

4.2 RQ2: レベル分布の隔たり

表 4 と図 1 に 4 ドメインと主要教材データの CEFR レベル分布を示す．

NAIST 教材（教科書・模範解答）は A1 が約 70%，A2 までで約 95% を占め，B1 以上は 3%–5% に留まる．これに対し 4 ドメインは A1 と A2 がいずれも 40%–50% 台で同程度であり，B1 以上が 6%–9% を占める．Python 公式チュートリアルは A1 が 53%，A2 が 36% と，4 ドメインに近い分布を示す．東大教材は A1 が 67% と NAIST 教材に近い比率だが，B2–C2 の高レベル構成要素を NAIST より広く含む．

章平均とプロジェクト平均に対する Cliff's δ を表 5 にまとめる．表の各セルは固定集合上の記述的 δ と，角括弧内に経験分布再標本化における 95%パーセンタイル範囲を示す．

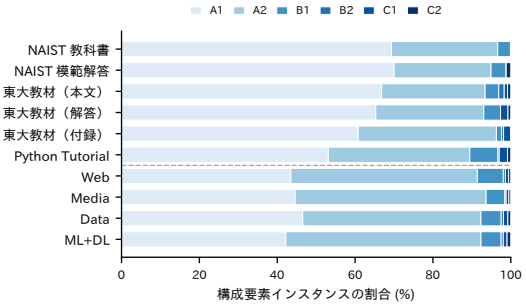

図 1　教材（上 6 段）とドメイン（下 4 段）の CEFR レベル分布．破線より上が教材データ

表 4　データセット別 CEFR レベル分布 (%)

対象	A1	A2	B1	B2	C1	C2
教科書	69.4	27.3	3.3	0.1	0.0	0.0
模範解答	70.0	24.9	3.9	0.0	0.0	1.2
東大本文	66.8	26.6	3.5	1.4	0.9	0.8
東大解答	65.3	27.8	4.3	0.0	1.9	0.7
東大付録	60.8	35.5	1.3	0.5	1.8	0.0
Tutorial	53.1	36.3	7.2	0.4	2.1	0.8
Web	43.5	47.9	6.7	0.5	0.8	0.6
Media	44.6	49.1	4.8	0.3	0.7	0.5
Data	46.6	45.8	5.2	0.6	1.1	0.7
ML+DL	42.2	50.1	5.2	0.6	1.0	1.0

教科書・模範解答は NAIST 教材，東大本文・東大解答・東大付録は東大教材の各層，Tutorial は Python 公式チュートリアルを表す．

線形平均では，NAIST 教科書と 4 ドメインの差は $\delta = 0.72$–0.79，NAIST 模範解答は $\delta = 0.93$–1.00 であり，いずれも再標本化範囲の下端が 0 を上回った．下端が 0 を上回ることは，章とプロジェクトの引き直し方によらず，実プロジェクト側の平均レベルが高いという差の方向が保たれることを表す．一方，東大教材は $\delta = 0.21$–0.32，Python 公式チュートリアルは $\delta = -0.17$–-0.09 で，いずれも再標本化範囲が 0 をまたいだ．範囲が 0 をまたぐことは，章とプロジェクトの引き直し方によって差の符号が入れ替わりうることを表し，この 2 教材については実プロジェクトとの分布差を一方向の隔たりとして主張できない．Python 公式チュートリアルの δ が負に傾くことは，B1 以上の構成要素の割合が 10.5%と 4 ドメイン

表 5　章平均とプロジェクト平均の Cliff's δ と 95%再標本化範囲（正値はドメイン側が高い）

教材	vs Web	vs Media	vs Data	vs ML+DL
NAIST 教科書	0.76 [0.31, 1.00]	0.73 [0.26, 1.00]	0.72 [0.27, 1.00]	0.79 [0.36, 1.00]
NAIST 模範解答	0.99 [0.94, 1.00]	0.94 [0.82, 1.00]	0.93 [0.78, 1.00]	1.00 [1.00, 1.00]
東大本文	0.31 [−0.10, 0.67]	0.23 [−0.15, 0.60]	0.21 [−0.17, 0.57]	0.32 [−0.08, 0.69]
Tutorial	−0.09 [−0.57, 0.48]	−0.14 [−0.63, 0.36]	−0.17 [−0.63, 0.33]	−0.09 [−0.56, 0.38]

東大本文は東大教材の本文層，Tutorial は Python 公式チュートリアルを表す．NAIST 模範解答と ML+DL の 1.00 は，すべてのプロジェクト平均が章平均を上回ったことを表す．

表 6　RQ2 指標感度の要約（正方向維持数／線形平均からの符号変化数，4 ドメイン中）

指標	教科書	模範解答	東大本文	Tutorial
線形平均	4/0	4/0	0/0	0/0
B1 以上割合	3/0	4/0	0/4	0/4
B2 以上割合	4/0	2/0	4/0	0/4
中央値	4/0	4/0	4/0	2/4
非線形平均	4/0	4/0	0/0	0/0

教科書・模範解答は NAIST 教材を表す．各セルの左は再標本化範囲の下端が 0 を上回ったドメイン数，右は δ の符号が線形平均と逆になったドメイン数を表す．

の 6.3–8.6%より高いことと整合し，言語機能の紹介を目的とする自習教材の性格を反映していると考えられる．

指標を替えたときの感度を表 6 に要約する．表の正方向維持は，percentile 方式の 95%再標本化範囲の下端が 0 を上回ることを表す．符号変化は，固定集合上の δ の符号が線形平均と逆になったドメイン数である．NAIST 教材では複数の指標で実プロジェクトとの差が正方向に維持されたが，全閾値で一様ではない．東大教材では指標によって方向が分かれ，B1 以上割合は負，B2 以上割合と中央値は正であった．Python 公式チュートリアルも指標によって符号が変わった．したがって中心的な記述は，NAIST では実プロジェクトとの分布差が複数指標で観測されること，東大教材とチュートリアルでは指標依存であることに限定する．

percentile 方式と basic 方式の方向判定は 80 比較中 79 比較で一致し，中心的な解釈は変わらなかった．

以上を踏まえた RQ2 への回答は次のとおりである．NAIST 教材では，構成要素が A1 に集中し実プロジェクトでは A2 以上へ比重が移るという方向の分布差が，複数の指標で観測された．東大教材と Python 公式チュートリアルの分布は実プロジェクトに比較的近く，差の有無と方向は要約指標の選択に依存した．

表 7　共通高頻度トークンの数と各教材データのカバー率

X	トークン数	教科書	模範解答	東大	Tutorial
50%	31	90.3%	71.0%	100.0%	90.3%
25%	11	90.9%	72.7%	100.0%	100.0%
10%	6	83.3%	83.3%	100.0%	100.0%

教科書・模範解答は NAIST 教材，東大は東大教材を表す．

副次的な観察として，NAIST 模範解答に C2 と判定されたインスタンスが 10 件存在し，それらはすべて `map` / `zip` の使用であった（NAIST 教科書には C2 が 0 件）．同様に，東大教材（本文 + 解答）の C2 判定 20 件のうち 17 件は `map` / `enumerate` の使用であった．pycefr が最高難度 C2 と判定する構成要素を教材作成者が通常の教材・解答内で使用するというこの現象は異なる 2 大学の教材で再観測されており，Rojpaisarnkit ら [18] の再分類提案（C2 → B2）と整合する．ただし教材作成者の難易度認識そのものは測定していないため，提案を間接的に支持する観察に留まる．

4.3　RQ3: 予約語・演算子の隔たり

予約語・演算子の隔たりを，共通高頻度トークンを各教材がどの程度カバーしているかとして測る．表 7 に，プロジェクト等重み集約で閾値 X を変えたときの共通高頻度トークンの数と，各教材データのカバー率を示す．東大教材のカバー率は本文層への出現による．

表 8　$X = 25\%$ の共通高頻度トークンと各教材データでの出現（プロジェクト等重み集約）

順位	トークン	教科書	模範解答	東大	Tutorial
1	`=`	○	○	○	○
2	`if`	○	○	○	○
3	`def`	○	○	○	○
4	`None`	×	×	○	○
5	`return`	○	○	○	○
6	`import`	○	○	○	○
7	`in`	○	○	○	○
8	`from`	○	×	○	○
9	`for`	○	○	○	○
10	`not`	○	×	○	○
11	`else`	○	○	○	○

○は教材データのコードに 1 回以上出現，×は出現なしを表す．順位 1–6 は $X = 10\%$ の共通高頻度トークンでもある．教科書・模範解答は NAIST 教材を表し，東大は本文層への出現による．

表 8 に，$X = 25\%$ の共通高頻度トークン 11 件と各教材データでの出現を示す．頻度順位の上位 6 件が $X = 10\%$ の共通高頻度トークンである．この 11 トークンの再標本化に対する選択率はいずれも 97.8% 以上であった．共通高頻度トークンの同定が，対象プロジェクトの引き直し方にほとんど依存しないことを表す．

教材側の未カバーは特定のトークンに集中していた．NAIST 教科書と NAIST 模範解答が $X = 10\%$ でカバーしないトークンはいずれも `None` のみであり，模範解答は $X = 25\%$ でさらに `from` と `not` をカバーしない．$X = 50\%$ では模範解答の未カバーは `is`，`raise`，`except`，`or` を含む 9 トークンに増え，NAIST 教科書も `None`，`is`，`raise` の 3 トークンをカバーしない．先行研究 [6] が単一の Kaggle コーパスで観察した `from`，`None`，`is` の欠落と同種の隔たりが，4 ドメインの実プロジェクトに対しても観測された．一方，$X = 50\%$ では`|`の選択率が 47.4%，`>` が 52.5%であり，境界付近の構成要素は不安定であった．感度分析のプール集計では $X = 25\%$ の`*`と`==`，$X = 10\%$ の `in` だけが入れ替わり，大規模プロジェクトの影響が確認された．東大教材はすべての閾値でプロジェクト等重みの共通高頻度トークンを 100%カバーし，Python 公式チュートリアルも $X = 10\%$ および $X = 25\%$ で 100%カバーする．ただし，これらはコード中の出現有無であり，教育上の網羅性を意味しない．また，`if` と `import` のようにトークンがカバーされていても，それを用いる構文的用法がカバーされるとは限らない．この点は 4.4 節で RQ1 の結果と突き合わせて述べる．

表 9　プロジェクト出現率のドメイン間範囲が大きいトークン

トークン	最大	出現率	最小	差
`/=`	ML+DL	75.9%	Web	69.2
`//`	ML+DL	100.0%	Web	60.0
`*=`	ML+DL	89.7%	Web	53.0
`&`	ML+DL	93.1%	Web	43.1
`>>`	Media	56.7%	ML+DL	39.4
`<<`	Media	66.7%	Web	36.7
`~`	Media	63.3%	Web	36.7

ドメイン差はプロジェクト出現率で解釈する．表 9 に出現率のドメイン間範囲が大きいトークンを示す．表の最大・最小は出現率が最大・最小のドメインを，差はその出現率の差をパーセントポイントで表す．`/=`は ML+DL の 75.9%に対して Web の 6.7%，`//`は ML+DL の 100%に対して Web の 40.0%で観測された．対象プロジェクト群では計算系ドメインへ広く普及した演算子があり，目標ドメインが明確な学生に向けた教材の補強候補になりうる．ただし，これを一般的なドメイン固有性として確立するには，より広いプロジェクト群での確認を要する．

以上を踏まえた RQ3 への回答は次のとおりである．4 ドメインに共通する高頻度トークンは代入，条件分岐，関数定義，インポートなどの基本要素であり，各教材はその大部分をカバーしていた．未カバーは `None` など少数のトークンに集中しており，予約語・演算子の水準の隔たりは限定的である．

4.4　RQ 横断の統合

3 つの観点の分析を統合すると，本研究の中心的な知見が得られる．すなわち，入門教材と実プロジェクトの間の隔たりは，予約語や演算子の出現の違いにとどまらず，それらを用いる構文の違いに及ぶ．まず，教材の文法要素の選択が実プロジェクトをカバーしているかという 1 節の問いに対しては，RQ3 で示し

たように，予約語・演算子の水準では各教材は共通高頻度トークンをおおむねカバーしていた．ただしカバーは完全ではなく，カバー率は教材と閾値によって71.0–100%であり，$X = 50\%$ の NAIST 模範解答では 31 トークン中 9 トークンが未カバーであった．その上で，教材がトークンをカバーしていても構文的用法のカバーは保証されない．RQ3 の $X = 10\%$ の共通高頻度トークンには `if` と `import` が含まれ，教材層別に確認すると，両トークンは 6 教材層のすべてで 1 回以上出現した．一方，RQ1 の条件式は東大本文に 1 件だけで，他の 5 層では 0 件であった．相対インポートは全 6 層で 0 件であった．条件式と相対インポートは，いずれも教材に現れるこれらの予約語を用いる構文である．それにもかかわらず教材コードにはほとんど出現しておらず，トークンのカバー率だけから構文 Class のカバーを推論できないことが，対象コーパス上で確認された．

レベル分布にも同種の限界がある．RQ2 は教材と実プロジェクトのレベル分布の全体を示し，RQ1 は教材の基準レベルの次のレベルの中で個別 Class の差を見る．例えばレベル分布の差が指標依存で単一方向にずれない東大本文でも，条件式は 1 件，相対インポートは 0 件である．反対に，全体の分布差が複数指標で観測された NAIST でも，すべての Class が一様に不足するとはいえない．レベル分布の隔たりの有無から，個別 Class の隔たりの有無を判定することはできない．

以上から，予約語・演算子，構文的用法，レベル分布の 3 観点は，互いの結果を代用または因果的に説明するものではなく，同じ固定コーパスのトークンの出現，構文的用法，全体構成を異なる分母で確かめる相補的な分析として併用する必要がある．いずれか単独の観点だけで教材を見直すと，本節で示した種類の隔たりを見逃すか，逆に過大に見積もるおそれがある．

5 議論

5.1 観測された隔たりの意味

観測された隔たりの中心は構文的用法にある．予約語・演算子の水準では，各教材は共通高頻度トークンの大部分をカバーしており，未カバーは `None` など少数のトークンに集中していた．一方，構文的用法の水準では，条件式と相対インポートが教材構成とドメインを横断して検出され，特に後者は抽出したどの教材コードにも一度も出現しなかった．また，ドメインへの着目が主にもたらしたのは隔たりのドメイン差の発見ではなく，この構文的用法の隔たりが 4 ドメインに共通して成立するという確認である．ドメイン差が明瞭に現れたのはトークンのプロジェクト出現率（RQ3）であった．

外部教材との記述的な対応として，京都大学の公開教材「プログラミング演習 Python 2023」 [9] は表紙に扱う予約語と扱わない予約語を明示しており，扱うと明示された 28 予約語は $X = 25\%$ の共通高頻度トークンに含まれる予約語 10 件をすべて含む一方，対象外とされた 7 語はいずれも共通高頻度トークンに含まれない．この対応は，本分析の共通高頻度トークンが教材設計の実際の判断と整合しうることを示唆する．他大学の公開教材との照合を広げれば，共通高頻度トークンを教材点検の参考情報として使えるかどうかを評価できると考えられる．

ただし，本研究の隔たりは頻度差として定義したものであり，出現頻度は学習上の重要度，難易度，前提知識を直接表すものではない．たとえば相対インポートはパッケージ構造の理解を前提とするため，入門直後に教えるべきかは自明ではない．条件式についても，可読性の観点から入門教育では意図的に扱わないという判断がありうる．一方で，両要素が実プロジェクトの大部分に出現するという結果からは，入門授業を終えた学生が実プロジェクトのコードを読み始めた時点でこれらの構文に遭遇する可能性が高い，という仮説が立てられる．この仮説は，教員・専門家による差分要素の評価や，入門修了者が実コードを読む際のつまずきの観察によって検証できる．教える項目に加えるのではなく，読めればよい構文として教材に注記するという軽量な扱いも選択肢になる．

5.2 妥当性への脅威

内的妥当性． pycefr のレベル割り当ては専門家の判断に基づいており，学生にとっての実際の難易度を反映しているとは限らない．Rojpaisarnkit ら [18]

は `map` / `zip` (C2 → B2)，リスト・辞書内包表記 (C1/C2 → B1/B2)，relative import (B1 → B2) 等の再分類を提案しており，4.1 節で示したように再分類は共通上位差分要素を変化させる．分類先が一意な Relative From 系のみを B2 へ再分類して RQ2 の指標を再計算すると，表 5 の記述的 δ の変化は最大でも 0.015 にとどまり，4.2 節の解釈は変わらなかった．分類先が一意でない提案を含めた再分類全体に対する感度は未評価である．$L_{course} = A2$ は $\tau \in [0.70, 0.85]$ の全シナリオで安定であり，教材ファイルを 1 件ずつ除外する再計算でも不変であった．また，pycefr の属性参照失敗 366 ノードを記録した．失敗はすべて実プロジェクト側で発生し，主要 2 要素に対応する AST ノードに失敗はない．修正前後の出力を比較すると主要 2 要素の順位変更は 0 件で，表 5 の解釈も変わらなかった．また，プロジェクト数は 4 ドメインでほぼ均衡している一方，コード量には約 4 倍の開きがあるため，全ファイルをプールした頻度集計は規模の大きいプロジェクトの影響を受けやすい．プロジェクト内で正規化するプロジェクト等重みの集約を併用し，主要な結果が集計方式に依存しないことを確認している．

外的妥当性．教材データは，2023 年度の NAIST の単一の Python 入門授業，東京大学の公開授業教材，Python 公式チュートリアルの 3 ソースに基づく．NAIST 授業は単一機関の単一授業であり，異なる目標や機関の授業では結果が異なる可能性がある．また，3 教材は授業用か自習用か，演習の有無など複数の属性で異なるため，観察された隔たりの差を特定の属性に帰属することはできない．一方，教材標本の拡張は本枠組みの延長線上で可能であり，5 節ではその足がかりとして京都大学の公開教材との照合を示した．実プロジェクト側の 118 プロジェクトは目的抽出された固定コーパスであり，各ドメインの母集団を代表すると主張しない．再標本化の範囲も固定コーパス内の安定性を表すものであり，未観測の教材やプロジェクトへの一般化を保証しない．これらは成熟した大規模 OSS プロジェクトであり，入門修了者が最初に取り組むタスクを代表するとは限らない点にも注意が必要である．また，プロジェクトの選定は 2020 年 7 月時点の基準に基づくため，選定時点とコード取得時点・教材年度の間には時間差がある．

構成概念妥当性．L_{course} は教材コード中の pycefr インスタンス分布の分位点であり，教材で「扱われている」ことも構成要素の出現有無・頻度に基づいて定めている．いずれも授業で説明・評価されたことや，学生が実際に習得したことを意味しない．RQ3 は予約語と演算子のみを対象とし，識別子や API 呼び出しは含まない．pycefr 分析（RQ1/RQ2）はツールに定義された 94 構成要素に限定され，Class の粒度も一様ではない．出力 Class と教育上の構成要素の対応は未確立であり，差分候補を教材に加えるべきかどうかの判断には教員による評価を要する．テストファイル・設定ファイルの除外により，テストやパッケージングに関わる構成要素の頻度は過小評価されうる．教材コードの抽出範囲も観察に影響する．公式チュートリアルの抽出は対話例と code-block 系ディレクティブのコード例に限られ，本文中の literal block は含まれない．現行のチュートリアルにはパッケージ内参照の節に相対インポートを示す literal block の例があり，チュートリアル層の相対インポート 0 件はこの抽出範囲のもとでの観察である．また，コードに出現しない構文が散文で説明される場合がある．東大教材は本文の発展項目の節で条件式を説明しており，公式チュートリアルは前述のとおり相対インポートの例を本文に含む．本稿の隔たりは 1 節で定めたとおり教材コード中の出現に基づく記述であり，散文での説明の有無は判定に含めていない．また，RQ1 は教材の基準レベルのすぐ次のレベルに絞っており，それより上のレベルの構成要素や，構成要素間の学習依存関係は扱っていない．RQ2 の比較単位である教材の章と実プロジェクトのプロジェクトは規模も役割も大きく異なり，章平均とプロジェクト平均の比較はこの粒度差の影響を受けうる．RQ2 を固定コーパス上の記述的比較に限定しているのはこのためでもある．

6 おわりに

本研究では，先行研究の語彙比較 [6] を拡張し，プログラミング入門教材と実プロジェクトの間にある文法要素の隔たりを，予約語・演算子，構文的用法，レ

ベル分布の 3 つの観点から実証的に分析した．主要な発見は次の 3 点である．第一に，条件式と相対インポートが，教材コードにほとんど出現しない一方で実プロジェクトの大部分に出現する構文的用法として検出された．この結果は 48 分析条件と再標本化に対して安定していた．pycefr の再分類は候補リストの構成を変えるが，両要素が教材にほとんど出現せず実プロジェクトに広く出現するという観察は保たれた．第二に，レベル分布の隔たりは要約指標の選択に依存し，単一の指標だけで差を主張することはできない．第三に，予約語・演算子では各教材が共通高頻度トークンの大部分をカバーし，未カバーは `None` など少数のトークンに集中した．3 つの観点を統合すると，`if` と `import` は全教材層に出現しても，条件式は 1 層だけ，相対インポートは 0 層であった．条件式と相対インポートはいずれも教材に現れる予約語を用いる構文であり，入門教材と実プロジェクトの間には，予約語や演算子の出現の違いにとどまらず，それらを用いる構文の違いがある．したがって，隔たりの点検には 3 つの観点の併用が必要である．

今後の課題は，5 節で立てた仮説の検証を含む，教員・実務家による差分候補の教育的妥当性の評価と，それを経た授業の見直しの支援への展開である．あわせて，予約語・演算子を超えた API 使用パターンや Pythonic イディオム [1] への拡張や，入門授業後の学生の習熟度の変化を追跡する縦断的研究が考えられる．

謝辞

本研究は一般財団法人テレコム先端技術研究支援センター SCAT 研究費助成，JSPS 科研費 JP26K21197 の助成を受けたものである．

参 考 文 献

[1] Alexandru, C. V., Merchante, J. J., Panichella, S., Proksch, S., Gall, H. C., and Robles, G.: On the Usage of Pythonic Idioms, *Proceedings of the 2018 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, 2018, pp. 1–11.

[2] Charatvaraphan, R., Chatchaiyadech, B., Sukijprasert, T., Ragkhitwetsagul, C., Choetkiertikul, M., Kula, R. G., Sunetnanta, T., and Matsumoto, K.: PyGress: Tool for Analyzing the Progression of Code Proficiency in Python OSS Projects, *2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2025, pp. 3997–4000.

[3] Cliff, N.: Dominance Statistics: Ordinal Analyses to Answer Ordinal Questions, *Psychological Bulletin*, Vol. 114, No. 3(1993), pp. 494–509.

[4] Dyer, R. and Chauhan, J.: An Exploratory Study on the Predominant Programming Paradigms in Python Code, *ESEC/FSE 2022: Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2022, pp. 684–695.

[5] Febriyanti, I., Fan, Y., Shimari, K., Matsumoto, K., and Kula, R. G.: How Maintainable is Proficient Code? A Case Study of Three PyPI Libraries, *2024 IEEE 35th International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2024, pp. 103–104.

[6] Fukushima, K., Ishio, T., Shimari, K., and Matsumoto, K.: Towards Assessment of Practicality of Introductory Programming Course Using Vocabulary of Textbooks, Assignments, and Actual Projects, *2023 IEEE 35th International Conference on Software Engineering Education and Training (CSEET)*, 2023, pp. 199–200.

[7] Garousi, V., Giray, G., Tuzun, E., Catal, C., and Felderer, M.: Closing the Gap Between Software Engineering Education and Industrial Needs, *IEEE Software*, Vol. 37, No. 2(2020), pp. 68–77.

[8] Golubev, Y., Li, J., Bushev, V., Bryksin, T., and Ahmed, I.: Changes from the Trenches: Should We Automate Them?, 2021. arXiv:2105.10157. Dataset available at `https://zenodo.org/records/4004118`.

[9] 喜多一，森村吉貴，岡本雅子: プログラミング演習 Python 2023, `https://repository.kulib.kyoto-u.ac.jp/dspace/handle/2433/285599`, 2023. 京都大学学術情報リポジトリ KURENAI.

[10] Luxton-Reilly, A., Simon, Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., and Szabo, C.: Introductory Programming: A Systematic Literature Review, *ITiCSE 2018 Companion: Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, (2018), pp. 55–106.

[11] Mason, R., Simon, Becker, B. A., Crick, T., and Davenport, J. H.: A Global Survey of Introductory Programming Courses, *SIGCSE 2024: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, 2024, pp. 799–805.

[12] Peng, Y., Zhang, Y., and Hu, M.: An Empirical Study for Common Language Features Used in Python Projects, *2021 IEEE International Conference on Software Analysis, Evolution and Reengi-*

neering (SANER), 2021, pp. 24–35.

[13] Python Software Foundation: The Python Tutorial, `https://docs.python.org/3/tutorial/`, 2024. Source: `https://github.com/python/cpython/tree/main/Doc/tutorial`. Document license: PSF License v2; code examples: Zero-Clause BSD License.

[14] Qian, Y. and Lehman, J.: Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review, *ACM Transactions on Computing Education*, Vol. 18, No. 1(2017), pp. 1–24.

[15] Ragkhitwetsagul, C., Kongwongsupak, K., Maneesawas, T., Puttiwarodom, N., Rojpaisarnkit, R., Choetkiertikul, M., Kula, R. G., and Sunetnanta, T.: jscefr: A Framework to Evaluate the Code Proficiency for JavaScript, *2024 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2024, pp. 863–867.

[16] Robles, G., Kula, R. G., Ragkhitwetsagul, C., Sakulniwat, T., Matsumoto, K., and Gonzalez-Barahona, J. M.: pycefr: Python Competency Level through Code Analysis, *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension*, 2022, pp. 173–177.

[17] Rojpaisarnkit, R., Robles, G., Gonzalez-Barahona, J. M., Matsumoto, K., and Kula, R. G.: Determining Code Proficiency Levels from Python Textbooks, *ACM Transactions on Software Engineering and Methodology*, Vol. 35, No. 7(2026), pp. 1–21.

[18] Rojpaisarnkit, R., Robles, G., Kula, R. G., Wang, D., Ragkhitwetsagul, C., Gonzalez-Barahona, J. M., and Matsumoto, K.: Towards Identifying Code Proficiency through the Analysis of Python Textbooks, *2024 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2024, pp. 288–299.

[19] 東京大学数理・情報教育研究センター: Python プログラミング入門, `https://utokyo-ipp.github.io/`, 2024. CC BY-NC-ND 4.0. Source: `https://github.com/UTokyo-IPP/utokyo-ipp.github.io`.

[20] Yang, Y., Milanova, A., and Hirzel, M.: Complex Python Features in the Wild, *Proceedings of the 19th International Conference on Mining Software Repositories*, 2022, pp. 282–293.

[21] Zid, C., Zampetti, F., Antoniol, G., and Penta, M. D.: A Study on the Pythonic Functional Constructs' Understandability, *ICSE '24: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 2024, pp. 1–13.