

演習ログ分析に基づくプログラミング学習向け 段階的支援エージェントの予備的検討

沼口 学聖[†]吉原 和明[‡]井口 信和[‡]近畿大学大学院総合理工学研究科[†]近畿大学情報学研究所[‡]

1. はじめに

国内におけるソフトウェア開発人材の不足は深刻化しており、実践的なプログラミング能力を有するデジタル人材の育成が急務となっている。この課題に対し、教育現場ではオンラインを活用した様々なプログラミング演習システムが導入されてきた。しかし、既存の Web ベースの演習システムの多くは、エラー発生時に詳細な解説や正解コードを即座に提示する傾向にあり、これがかえって初学者が自らバグの原因を特定し、仮説検証を行う自律的なデバッグ能力の育成を阻害するという問題が指摘されている。オンライン環境においても学習者の主体性を引き出すためには、一方的な正解の提示ではなく、つまづき状況に応じた適切な教育的支援である足場かけが必要不可欠である。

こうした背景から、コミュニケーションツールである Slack をインターフェースとし、Python を対象としたオンライン演習システムを構築してきた。さらに既報においては、従来の自動採点機能に加え、初学者の自律的なデバッグを促す段階的支援エージェントの設計および実装を提案した [2]。本機能は、自動テスト失敗時に生のトレースバックを初期応答として提示し、その後は学習者自身の能動的な要求であるリアクション操作やボタン押下に応じて、抽象的なレビューから具体的な修正方針へと段階的に情報開示のレベルを制御するアプローチを特徴としている。

既報では主にシステムの基本設計と教育効果の検証計画について論じたが、本提案手法が実際の演習環境で有効に機能するためには、導入した支援機能が開発側の想定通りに学習者に利用されているか、また、その際のデバッグプロセスがシステムの収集するログから定量的に追跡可能であるかを精緻に確認する必要がある。そこで本稿では、本システムを用いた少人数の初学者による演習を実施し、教育効果そのものの断定ではなく、演習ログから学習者の行動変容や課題難易度、および分析に必要なログの十分性について評価することを目的とした予備的検討について報告する。

2. プログラミング演習システム

2.1 システム概要

本システムでは、学習者が演習を実施するためのインタフェースに Slack を使用する。本システムは、プログラム提出機能、自動採点機能、および段階的支援機能から構成される。

プログラム提出機能は、学習者が作成した Python プログラムを Slack のチャンネル上で提出するための機能である。

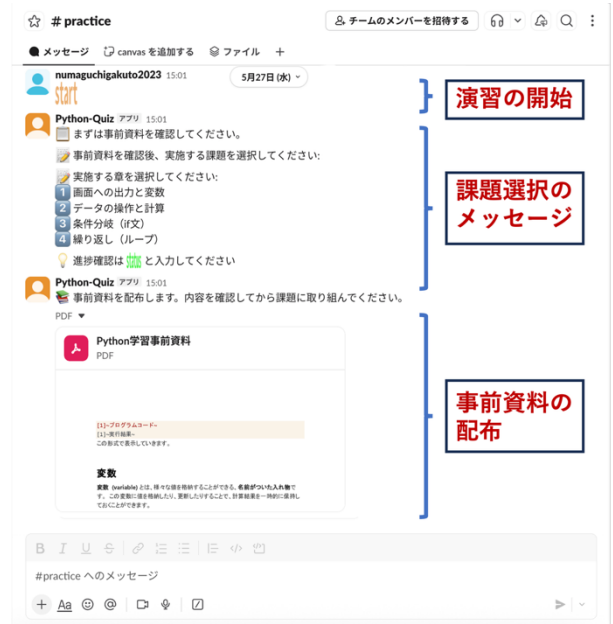


図1 Slack 上でのコマンド発行の様子

自動採点機能は、提出されたプログラムに対してバックエンドで自動テストを実行し、その成否を判定する機能である。また、本システムでは評価実験における行動分析のため、学習者の提出内容や成否の履歴、各種支援の要求イベントなどの演習ログを、バックエンド側で JSON 形式のファイルとして自動的に収集・記録する仕組みを備えている。段階的支援機能は、既報で提案した、テスト失敗時に学習者の能動的なアプローチに応じてヒントを段階的に提示し、自律的なデバッグを促すための機能である。

2.2 Slack 上でのプログラミング演習

学習者が Slack 上でプログラミング演習を実施する様子を図 1 に示す。図中では、student1 が学習者であり、Python-Quiz が本システムのアプリケーションである。

学習者は、対象の Slack チャンネル上で「start」の絵文字リアクションを送信することで演習を開始する。演習が開始されると、Python-Quiz から事前学習のための資料が送信され、学習者はまず基礎知識の学習を行う。その後、実施する課題番号の絵文字をチャンネル上に送信することで、具体的なプログラミング演習へと移行する。

演習に移行すると、Python-Quiz からベースとなる不完全なソースコードが配布される。学習者はこのコードをローカル環境等で修正し、対象のチャンネルへファイルをアップロードすることで提出を行う。提出されたファイルに対しては対応するユニットテストが自動で実行され、採点結果は Slack の API を通じて、提出が行われたメッセージのスレッドへの返信としてリアルタイムに通知される。これにより、学習者は指導者の手を煩わせることなく、自身の

[†]Gakuto NUMAGUCHI, Graduate School of Science and Engineering, Department of Informatics, Kindai University

[‡]Kazuaki YOSHIHARA, Nobukazu IGUCHI, Institute of informatics, Kindai University

コードが要求仕様を満たしているかどうかをリアルタイムに確認しながら演習を進めることが可能となる。

3. 予備評価実験

3.1 予備評価実験の目的

本研究における予備実験の目的は、提案する支援アプローチの教育効果の検証ではなく、システムが設計通りに動作しているかの機能検証、および本実験に向けたデータ分析の実現可能性の確認である。具体的には、少人数の実験を通じて以下の 5 点を確認および検証することを目的とする。

(1) システムの正常動作の確認

学習者がプログラムを提出した際のエラー表示から、SOS リアクションによる支援要求、各種レビューの提示、および修正コードの再提出にいたる一連の機能が正常に動作するかを検証する。

(2) 演習ログからの行動プロセスの再現性

バックエンドに記録された演習ログに基づき、学習者の一連のデバッグ行動や試行錯誤のプロセスを、時系列順に過不足なく復元できるかを確認する。

(3) 支援機能の利用実態の定量的把握

学習者がエラーに直面した際、どの段階まで支援機能を活用したかを定量的に把握し、システムがどのように利用されたかを評価する。

(4) デバッグの推移と継続性の分析

段階的支援の提示を受けた後、同一のエラーが継続したか、あるいはエラーが解消され解決に至ったかというデバッグのプロセスを追跡できるかを検証する。

(5) 演習課題および支援文の改訂

演習ログから読み取れる課題ごとの難易度やエラーの解消状況に基づき、本実験に向けた演習課題の難易度調整、およびエージェントが提示する支援メッセージの修正箇所を抽出する。

3.2 実験設計および演習の手順

本予備実験には、Python の学習経験が浅い初学者 2 名が参加した。実験は Slack 上の専用チャンネルにおいて実施され、学習者は自身のペースで Python プログラムを作成・実行し、課題に取り組んだ。演習時間は 15 時開始の 60 分間とし、基礎的な Python プログラミングに関する計 5 課題（課題 1-1～1-5）を実施した。

本システムにおける支援機能の利用は任意とし、学習者に対して利用を義務付ける指示やタイミングの指定は行わなかった。学習者がエラーの解消に行き詰まった際、自発的に支援を要求した行動ログを収集・分析することを目的としている。

演習の具体的な手順は以下の通りである。まず学習者は、あらかじめ指定された演習用の Slack チャンネルに参加し、開始を意味する start の絵文字リアクションをシステム宛てに送信して演習を開始する。演習が始まると、システムから該当の課題に必要な基礎知識をまとめた事前学習資料が提示されるため、学習者はまずその内容を閲覧して学習を行う。

次に、学習者が自身で実施したい課題番号の絵文字をチャンネル上に送信すると、システムからベースとなる不完全なソースコードが自動で配布される。学習者は自身の開発環境等でコードの修正を行った後、完成したソースコー

ドファイルを演習チャンネル上にアップロードして提出する。

提出されたコードに対してシステムは自動テストを実行し、その成否を判定する。自動テストが不合格であった場合、システムはスレッドへの返信としてエラーログを学習者に提示する。学習者はこのエラーログを確認して自力でのデバッグを試みるが、解決が困難であると判断した場合は、必要に応じて SOS を意味する絵文字リアクションを付与したり、詳細を見るボタンを押下したりすることで、システムから段階的なヒントレビューを取得する。学習者は提示されたヒントを参考にしながら、コードの修正と再提出を繰り返す。最終的にユニットテストに合格することで該当課題が完了となり、学習者は次の課題番号の絵文字を送信して同様の手順で演習を継続する。

3.3 収集する演習ログの項目

学習者の行動プロセスを詳細に分析するため、バックエンドサーバ側で自動的に記録および収集するログ項目を表 1 に定義する。これらのデータは、学習者のアクションが発生したタイミングで JSON 形式のファイルとしてリアルタイムに記録される。

システムが実際に出力する蓄積データの一例として、ソースコード提出時における JSON 形式の行動ログを図 2 に示す。このように学習者のアクションごとに独立したオブジェクトとしてデータが生成され、タイムスタンプを基準としてソートを行うことで、一連の演習の軌跡を辿ることが可能である。

表 1：ログ項目

ログ項目	データ型	内容・説明
event_type	String	行動の種別を示す。 ソースコードの提出を示す SUBMIT や、支援要求を示す各種イベント名を記録する
user_id	String	匿名化された学習者の固有ハッシュ値
task_id	String	実施中の課題番号
timestamp	DateTime	行動が発生した、またはシステムが応答した日時の世界標準時表現
submission_count	Integer	該当課題におけるソースコードの累計提出回数
result	String	自動採点の結果を示す。 正解を示す correct または不正解を示す incorrect を記録する
error_type	String	自動テスト失敗時に発生した Python のエラー種別であり、正解時は None とする
error_log	String	出力されたエラーメッセージや生のトレースバックデータの文字列

図 2：演習ログの JSON 構造例

```
{
  "event_type": "SUBMIT",
  "user_id": "6c76fc5bb9b60ddb",
  "task_id": "3-1",
  "timestamp": "2026-07-18T10:41:15.608Z",
  "submission_count": 1,
  "result": "incorrect",
  "error_type": "SyntaxError",
  "error_log": "Traceback..."
}
```

3.4 分析方法

学習者の学習プロセスを収集した演習ログを用い、以下の手順で提案アプローチの妥当性を検討する。まず、同一のユーザ ID および タスク ID を持つログをタイムスタンプの昇順でソートし、学習者ごとの学習プロセスのタイムラインを復元する。このタイムライン上で、エラータイプが変化したタイミングや、結果が `incorrect` から `correct` に遷移するまでの過程を追跡する。次に、各課題において SOS リアクション や ボタンクリック などのアクションがどのように実行されたかを把握し、学習者の試行錯誤やデバッグプロセスの個別事例を分析する。支援提示の直後の提出において、同一のエラータイプが継続している場合は支援文の表現に改善の余地があると判断し、本実験に向けたテキストのブラッシュアップの指標とする。また、特定の課題において 提出回数 や 支援要求回数が突出して多い場合は、課題自体の難易度や事前資料の構成を見直す材料とする。

4. 結果

4.1 演習ログの統計と各課題における支援の利用状況

本予備実験は参加者が 2 名と小規模であり、統計的な相関分析を行うには標本数が不十分である。そのため、本研究では課題ごとの提出回数、支援機能の利用有無の把握にとどめ、演習ログに基づくデバッグプロセスの復元を中心に分析を行う。

予備実験における学習者の基本統計、および各課題において支援機能がどのように利用され、デバッグに繋がったかの集計結果を表 2、表 3 に示す。「-」は当該課題で支援要求が利用されなかったことを示す。

一人目は課題 1-1 から 1-5 までの全 5 課題を制限時間内に完了した。課題 1-1 を除くすべての課題で支援機能を活用しており、課題 1-3 では抽象レビューのみを閲覧して解決に至った一方、課題 1-2、1-4、および 1-5 では抽象レビューおよび具体レビューの両方を閲覧した上で試行錯誤を重ね、正解に到達していた。

一方、二人目は課題 1-1 から 1-3 までを完了したものの、`str` 関数による明示的な型変換を要する課題 1-4 においてつまづきが発生した。二人目は課題 1-4 において計 4 回の提出を行い、抽象レビューおよび具体レビューを閲覧したものの `TypeError` を解消できないまま制限時間 (60 分) を迎えたため、課題 1-4 は未完了となった。また、課題 1-5 につ

表 2：基本統計および支援機能の利用状況（一人目）

課題番号	1-1	1-2	1-3	1-4	1-5
提出回数	1 回	2 回	2 回	4 回	6 回
支援要求までの提出回数	-	2 回	1 回	1 回	3 回
抽象レビュー	-	○	○	○	○
具体レビュー	-	○	-	○	○
課題完了状況	完了	完了	完了	完了	完了

表 3：基本統計および支援機能の利用状況（二人目）

課題番号	1-1	1-2	1-3	1-4	1-5
提出回数	2 回	3 回	3 回	4 回	0 回
支援要求までの提出回数	-	1 回	1 回	1 回	-
抽象レビュー	-	○	○	○	-
具体レビュー	-	○	○	○	-
課題完了状況	完了	完了	完了	未完	未完

いては、課題 1-4 に時間を費やしたことで着手する前に時間切れとなり、未完了（提出回数 0 回）に終わった。

支援機能（SOS リアクション）は全 10 セッションの取り組み中 7 セッションで利用された。抽象レビューが表示された 7 件のうち、6 件においては学習者が「具体レビュー」ボタンを押下して追加の支援情報を確認していた。

4.2 演習ログで判明した学習プロセスの一例

システムによる一連のエラーハンドリング、支援要求の検知、およびレビュー提示機能が設計通りに正常動作しているかを確認するため、実際の JSON ログから「支援機能を利用したにもかかわらず解決に至らなかった事例」のタイムラインを復元した結果を図 3 に示す。演習時間は 15 時開始の 60 分間であり、当該学習者は 15:28 に課題 1-4 に着手している。

図 3 に示す通り、学習者は初回提出 (15:28) で `TypeError` を発生させた後、2 回目の提出 (15:33) でも不合格となったため、SOS リアクションを実行 (15:35) して抽象レビューを確認した。しかし、7 分後の 3 回目の提出 (15:42) でも同一のエラーが解消されず、「詳細を見る」ボタンを押下 (15:45) して `str()` 関数の利用を促す具体レビューを確認した。

その後、学習者は `str()` 関数を用いた修正を試みたものの、8 分後の 4 回目の提出 (15:53) においても `TypeError` を解消できず、最終的に演習時間の終了時刻 (16:00) を迎えて課題未完了となった。

5. 考察

5.1 システムの正常動作と学習ログの再現性

収集されたログから、ソースコードの提出、自動採点、SOS リアクション、抽象レビュー提示、具体レビューボタンの押下、再提出という主要なイベントを確認できた。また、`timestamp` と `event_type` を用いることで、図 3 に示す事例について、支援要求から再提出までの行動過程と所要時間を時系列に復元できることを確認した。このことから、演習ログを用いて学習者のデバッグ過程を分析できる見通しを得た。

図 3：学習プロセスの一例

【学習者の行動：Slack】	【システムの応答：エージェント】
15:28 ---- 1回目のコード提出 (SUBMIT) ----->	15:28
<---- 自動テスト失敗の通知および生のログ提示 -----	(レベル1支援)
(TypeError: can only concatenate str)	
15:33 ---- 自力修正後の2回目提出 (SUBMIT: 不合格) ----->	
15:35 ---- 支援要求の送信 (SOS絵文字リアクション付与) ----->	
<---- 抽象レビューの提示 (ヒントの提示) -----	15:35
(「データの種類 (型) が合わない計算...」)	(レベル2支援)
15:42 ---- 3回目のコード提出 (SUBMIT: 同一エラーが発生) ----->	
15:45 ---- 詳細を見るボタンの押下 (BUTTON_CLICK) ----->	
<---- 具体レビューの提示 (修正方針の提示) -----	15:45
(「数値を文字にするなら str() を使しましょう」)	(レベル3支援)
15:53 ---- 4回目のコード提出 (SUBMIT: 不合格) ----->	
<---- 自動テスト失敗の通知および生のログ提示 -----	15:53
(TypeError: can only concatenate str)	(レベル1支援)
16:00 x (60分経過: エラー未解消のまま課題1-4完了ならず終了)	

5.2 支援機能の現状課題

収集されたログからシステムの現状の課題として、型変換課題「str()」におけるつまずきの継続と支援メッセージの解像度の不足が考えられる。課題1-4および課題1-5は、数値型変数をstr()関数によって文字列型に変換し、文字列と結合して出力させる課題であった。

図3のログ分析から明らかなように、学習者Bは15:45に具体レビューを要求し、「数値を文字にするならstr()を使いましょう」という具体レビューを確認した。しかし、その後8分間(15:45~15:53)に及ぶ思考・コード修正を行ったものの、4回目の提出(15:53)においても同一のTypeErrorを解消できず、そのまま制限時間の16:00を迎えて時間切れ(未達成)となった。

この経過は、初学者にとって「str()という関数名の知識」と「自身のコードのどの変数に対し、どのような文法・構造でコーディングするか」の間に大きな隔たりがあることを示している。

単に関数名を提示するだけでは、初学者が型変換の構文や文法構造を自身のコードへ正しく適用できない場合があることが示唆された。ただし、本事例における課題未完了には、支援文の表現不足だけでなく、課題全体の難易度設定、参加者の事前理解度、および残された作業時間の少なさなども複合的に影響した可能性がある。

本実験に向けては、「文字列」+str(変数名)のような具体的コードへの適用パターン(構文例)の提示を追加するなど、具体レビューの解像度をより高める改訂が必要である。なお、本予備実験は参加者2名を対象とした単一回の試行であり、得られた結果から支援機能の教育効果や一般的な有効性を判断することはできない。

6. まとめ

本研究では、Slack上で動作するプログラミング演習支援システムにおいて、学習者のデバッグ行動に応じた段階的フィードバック機能が正常に動作するか、および行動ログから試行錯誤プロセスを再現できるかを評価することを目的として予備実験を実施した。

実験の結果、システムは設計通りにリアルタイムでのコード採点、SOSリアクションの検知、および段階的レビュ

ーの提示を実行し、全イベントが時系列ログとして保存されることが確認された。また、復元したデバッグタイムラインの質的分析により、特にstr()を用いた型変換課題において、単に関数名を提示する支援のみでは初学者のつまずきを解消できず、具体レビュー閲覧後も同一のエラーで停滞し、時間切れに至るという本実験に向けた明確な改善課題が抽出された。

今後は、本予備実験で得られた知見に基づき、具体レビューにおける構文適用例の提示や概念解説の追加を行い、支援メッセージの解像度を向上させる。その上で、本実験において本システムの段階的支援が学習者の自律的デバッグ能力向上および学習モチベーションに与える影響を検証する予定である。

7. 参考文献

- [1] 経済産業省:IT 人材需給に関する調査,入手先 https://www.meti.go.jp/policy/it_policy/jinzai/houkokusyo.pdf
- [2] 沼口学聖, 吉原和明, 井口信和, “Slack を用いたプログラミング学習における段階的支援エージェントの設計”, 信学技報, vol.126, no.104, pp.7-11, July 2026.
- [3] 沼口学聖, 吉原和明, 井口信和, “Slack を用いた Python プログラミングのオンライン演習システムの開発”, 情報処理学会第 87 回全国大会講演論文集, 4, no. 2ZJ-05, pp. 487-488, Mar. 2025.
- [4] 宮城勝, 吉原和明, 井口信和:Slack を用いて学習状況の可視化を可能とするネットワーク演習支援システムの提案, インターネットと運用技術シンポジウム論集, 2023(1), 905-906
- [5] Janneke van de Pol, Monique Volman, Jos Beishuizen, “Scaffolding in Teacher–Student Interaction: A Decade of Research,” Educational Psychology Review, vol. 22, no. 3, pp. 271-296, Sep. 2010.
- [6] 奥村光貴, 教育エビデンス抽出のための背景情報の差異調整手法の提案と考察, 教育システム情報学会誌, Vol. 42, No. 2, 2025, pp. 185-193
- [7] 酒井, 長谷川: Sphero を用いた小学校プログラミング学習単元の開発:日本科学教育学会研究会研究報告, vol31, No. 8, pp. 117-122(2017)
- [8] 井垣 宏, 齋藤 俊, 井上亮文, 中村亮太, 楠本真二:プログラミング演習における進捗状況把握のためのコーディング過程可視化システム C3PV の提案, 情報処理学会論文誌, Vol. 54, No. 1, pp. 330-339(2013)
- [9] 長谷川伸, 松田承一, 高野辰之, 宮川治, プログラミング入門教育を対象としたリアルタイム授業支援システム, 情報処理学会論文誌, Vol. 52, No. 12, pp. 3149-3235, 2011 年 12 月
- [10] 新田章太, 小西俊司, 竹内郁雄:複数言語に対応しやすいオンラインプログラミング学習・試験システム track, 情報教育シンポジウム論文集, 2019, pp. 114-121.
- [11] 加藤利康, 石川孝:授業支援システムにおけるプログラミング演習のための学習状況支援機能の設計と評価, 情報処理学会研究報告, Vol. 2012-CE-113, No. 6, pp. 1-8(2012)
- [12] 京都大学: プログラミング演習 Python, <<http://hdl.handle.net/2433/245698>> (参照 2024-11-08)