卒業論文

大規模言語モデルに基づくマルチエージェントによるプログラミング学習支援

指導教官　　村上 陽平 教授

立命館大学　情報理工学部
先端社会デザインコース　4 回生
2600210015-0

有田 隆太朗

2024 年度（秋学期）卒業研究 3（CH）
令和 7 年 1 月 31 日

大規模言語モデルに基づくマルチエージェントによるプログラミング学習支援

有田 隆太朗

内容梗概

近年，大規模言語モデル（Large Language Models，LLM）を活用したプログラミング学習の支援ツールが注目を集めている．GitHub Copilot や ChatGPT のようなツールは，ユーザーが提示するプログラミング問題やコードに対し，即座に解答コードやアドバイスを提供することで，学習効率を大きく向上させる可能性を秘めている．特に，プログラミング初心者や独学者にとって，こうしたツールは迅速なフィードバックを得るための有力な手段である．

しかしながら，既存の LLM を活用したプログラミング支援では，学習者が提出したコードに対して正解コードを直接提示するため，短期的な問題解決には効果的であるものの，学習者自身が試行錯誤する機会を奪い，深い理解や応用力の育成にはつながりにくい．たとえ「正解コードを含まないフィードバック」を求めるプロンプトを用いても，モデルが意図せず正解コードを含む回答を生成してしまう場合がある．また，シングルエージェント型でフィードバックに正解コードが含まれていた場合にフィードバックを再生成するよう求めても，正解コードを含むフィードバックを生成する事しかできなかった．

そこで，本研究では，フィードバックを返すエージェント（FAgent）だけでなく，フィードバックに正解コードが含まれていないかをチェックするエージェント（CAgent）からなるマルチエージェント型のプログラミング学習支援システムを構築する．具体的には，ChatGPT を用いて，各エージェントのプロンプトを作成し，エージェント間のインタラクションを通して，正解コードを含まないフィードバックを生成し，学習者が試行錯誤を重ねながら理解を深められるシステムを構築する．本手法の実現にあたり，取り組むべき課題は以下の二点である．

正解コードの検出

一つ目は正解コードの検出である．FAgent を使って生成された ChatGPT の返答に正解となるコードが含まれているかどうかを CAgent で判定する仕組みが必要だ．しかし，正解コードの基準をどう定義するか，関数名やコードの一部が含まれる場合にどこまで許容するかが課題となる．また，多様な解法が存在する問題では，それら全てを網羅的に検出することが難しい．

学習効果を高めるフィードバックの生成

二つ目は学習効果を高めるフィードバックの生成である．ChatGPT からのフィードバックが学習者の理解を促進するよう，適切なヒントで提示する必要がある．しかし，詳細すぎるヒントは学習者の自主的な思考を阻害する可能性があり，逆に抽象的すぎると理解が進まない場合がある．そのため，学習者のレベルに応じた最適なフィードバックの生成方法が求められる．

提案手法の有用性を示すために，これらの仕組みを組み込んだプログラミング学習支援システムを実装し，動作確認を行った．動作確認では，本システムを活用しプログラミング課題に一定回数取り組む．動作確認を通して，本システムがどの程度有用性があるかを評価する．

1 つ目の正解のコードの検出という課題に関しては，フィードバックに正解コードが含まれていた場合の正解コードの検出数で評価する．

2 つ目の学習効果を高めるフィードバックの生成という課題に関しては，問題点の指摘精度，献身性，具体性，難易度，CAgent の有無による正解コード数で評価する．動作確認の結果，各課題に対して得られた知見は以下の通りである．

正解コードの検出

動作件数 30 件中正解コードが含まれていた件数 25 件に対して，正解コードを検出できたのは 15 件であり，判定率は 60%となった．

学習効果を高めるフィードバックの生成

問題指摘精度について，入力された問題文とコードに必要な修正点が 5 つなのに対して，平均値 3.47 と言う結果が得られた．また，CAgent の有無によるフィードバックに含まれる正解コード数については CAgent 無しが 40 個，CAgent 有りが 37 個となり，7.5%減少していることが確認された．

Multi-Agent Programming Learning Support Based on Large Language Models

Ryutaro Arita

Abstract

In recent years, tools to support programming learning based on Large Language Models (LLM) have attracted a lot of attention. Tools such as GitHub Copilot and ChatGPT have the potential to greatly improve learning efficiency by providing solution code and advice instantly for programming problems and code presented by users. Particularly for beginner and self-taught programmers, these tools are a powerful means of getting quick feedback.

However, exiting programming support based on LLM directly presents the correct code in response to the code submitted by the learner, which is effective for short-term problem solving, but deprives the learner of the opportunity for their own trial-and-error and does not lead to the development of deep understanding and application skills. Even if we use a prompt for feedback that does not contain the correct code, the model may unintentionally generate a response that contain the correct code. In addition, asking to regenerate feedback when the feedback contained the correct code in a single-agent model could only generate feedback that contained it.

In this research, a multi-agent programming learning support system is constructed, consisting not only of an agent that returns feedback (FAgent) but also an agent that checks whether the feedback contains correct code (CAgent). Specifically, we will create prompts for each agent, use a LLM to generate feedback that does not contain the correct code through interactions between agents, building a system that allows learners to deepen their understanding through trial-and-error. In order to realize this method, there are two issues that need to be addressed.

Detection of correct codes

The first is the detection of correct codes. We need a mechanism for CAgent to determine whether the ChatGPT replies generated using FAgent contain codes that are correct. However, the challenge is how to define the criteria for correct codes and how far to allow when function names or parts of code are included. In

addition, it is difficult to detect all of them exhaustively in problems where a variety of solution methods exist.

Generating feedback to enhance learning

The second is the generation of feedback that enhances learning effectiveness. Feedback from the ChatGPT needs to be presented with appropriate hints to facilitate learners' understanding. However, hints that are too detailed may inhibit learners' independent thinking, while those that are too abstract may not be understood. The best may be to generate feedback according to the learner's level is required.

A programming learning support system incorporation these mechanisms was implemented and tested to demonstrate the usefulness of the proposed methods. In the operation check, the system is used to work on programming tasks a certain number of times. We evaluate the usefulness of the system through it.

For the first issue of detecting the correct code, the evaluation is based on the number of correct codes detected when the feedback contains the correct code.

The second issue of generation feedback that enhances learning is evaluated in terms of problem identification accuracy, dedication, concreteness, difficulty, and number of correct codes with and without CAgent. The findings obtained for each task as a result of the operational checks are as follows.

Detection of correct codes

Of the 30 operations, 25 contained the correct code and 15 were able to detect the correct code, resulting in decision rate of 60%.

Generating feedback to enhance learning

For the accuracy of the problem indication, the results showed an average value of 3.47 compared to 5 modifications required to the entered problem statement and code. The number of correct codes included in the feedback with and without CAgent was 40 without CAgent and 37 with CAgent, a decrease of 7.5%.

卒業論文タイトル

目次

第1章 はじめに

近年，大規模言語モデル（Large Language Models，LLM）を活用したプログラミング学習の支援ツールが注目を集めている．GitHub Copilot や ChatGPT のようなツールは，ユーザーが提示するプログラミング課題やコードに対し，即座に正解コードやアドバイスを提供することで，学習効率を大きく向上させる可能性を秘めている．特にプログラミング初学者にとって，こうしたツールは迅速なフィードバックを得るための有力な手段である．

しかしながら，既存の LLM を活用したプログラミング学習支援では，学習者が提出したコードに対して正解コードを直接提案するため，短期的な問題解決には効果的であるものの，学習者自身が試行錯誤する機会を奪い，深い理解や応用力の育成には繋がりにくい．また，たとえ「正解コードを含まないフィードバック」を求めるプロンプトを用いても，モデルが意図せず正解コードを含む回答を生成してしまう場合がある．

そこで本研究では，フィードバックを返すエージェント（FAgent）だけでなくフィードバックに正解コードが含まれていないかをチェックするエージェント（CAgent）からなるマルチエージェント型のプログラミング学習支援システムを構築する．具体的には，大規模言語モデルを用いて，各エージェントのプロンプトを作成し，エージェント間のインタラクションを通して，正解コードを含まないフィードバックを生成し，学習者が試行錯誤を重ねながら理解を深められるシステムを構築する．本手法の実現にあたり，取り組むべき課題は以下の二点である．

正解コードの検出

一つ目は正解コードの検出である．FAgent を使って生成された ChatGPT の返答に正解となるコードが含まれているかどうかを CAgent で判定する仕組みが必要だ．しかし，正解コードの基準をどう定義するか，関数名やコードの一部が含まれる場合にどこまで許容するかが課題となる．また，多様な解法が存在する問題では，それら全てを網羅的に検出することが難しい．

学習効果を高めるフィードバックの生成

二つ目は学習効果を高めるフィードバックの生成である．ChatGPT からのフィードバックが学習者の理解を促進するよう，適切なヒントで提示する必要がある．しかし，詳細すぎるヒントは学習者の自主的な思考を阻害する可能性があ

り，逆に抽象的すぎると理解が進まない場合がある．そのため，学習者のレベルに応じた最適なフィードバックの生成方法が求められる．

以下，本研究では第 2 章において大規模言語モデルを用いたプログラミング学習支援に関する関連研究を紹介し，これに基づいて本研究を行う意義を述べ，第 3 章では，本研究において構築したシステムについて詳細を記述する．第 4 章では，動作確認の結果や評価方法を記述し，結果から課題に対してどの程度精度が向上しているかを記述し，第 5 章で本稿をまとめる．

第2章 大規模言語モデルに基づくプログラミング学習支援

本章では，まず大規模言語モデルに基づくプログラミング学習支援にどのような課題があるのか，LLM を用いた正解コードを含まないフィードバック生成の試行と関連研究での課題から本研究の意義を明記する．

2.1 フィードバック生成の試行

LLM に基づくシングルエージェント型によるプログラミング学習支援の課題を実際に確認するため，GPT-4o を用いて試行を行った．方法として，1 回目の指示として，問題文とコードに対して正解コードを含まないフィードバックを生成するよう求める．2 回目以降は，フィードバックに正解コードが含まれているかの確認と含まれていた場合はフィードバックを再生成するよう求め，再生成後のフィードバックにまだ正解コードが含まれていた場合は，2 回目の指示を繰り返す．各指示後の GPT-4o の出力を表 1 に示す．本研究では，マルチエージェント型のシステム構築を行うため，ここでは，シングルエージェント型でフィードバックに正解コードが含まれていた場合を前提として試行する．表 1 から分かるように，正解コードが含まれるフィードバックを含まないよう再生成するよう求めた時，正解コードを検出出来たとしても同じ正解コードが含まれてしまい結果に変化は見られなかった．

表 1：フィードバック生成の試行

指示内容	GPT-4 の出力（一部）
正解コードを含めずフィードバック生成	「for i in range(a)」のような形で修正できます．
正解コードが含まれていた場合に，再生成	「for i in range(a)」が正解コードとして含まれていますね．修正後のフィードバックは以下です． ...「for i in range(a)」の形で修正すると良いです．

2.2 関連研究

関連研究では，従来のコードレビュー手法が持つ即時性や個別化の限界を克服し，プログラミング教育環境での「学習者に優しいコードレビュー」を提供するシステムの構築を目指した．オンラインジャッジシステムから提出コードと解答例を収集し，データセットを構築し，次に GPT-4 を用いて学習者向けのフィードバックを生成するプロンプトを設計し，複数回の改良を実施した．その後，コード正当性確認モジュールやコードレビューコメント生成モジュールを実装した．最後にシステムを用いて応答時間や API コスト，フィードバックの質を評価する実験を行う．以下では先行研究の結論をもとに，本研究の課題を記述する．

2.2.1 仕様

両モジュールとも GPT-4 を用いて開発し，提出コードに対するフィードバックではコードの修正箇所を強調表示し，修正コードそのものは提供しない設計を採用した．

2.2.2 コード正当性確認モジュール

学習者が提出したコードが正しいかを判断し，エラー箇所を特定するモジュールである．フィードバック生成前に基本的な構文エラーを事前にチェックし，フィードバックの必要のないコードに対してはフィードバックを生成しない．

2.2.3 コードレビューコメント生成モジュール

コード正当性確認モジュールでフィードバックが必要と判断された提出コードに対してフィードバックを生成．

2.2.4 評価

プログラミング教育経験のある 6 名が参加し，実験を行った．実験結果，アンケートより，改良システムの有用性を 4 つの項目（RQ1～RQ4）を通じて検証した．実験結果は表 1 の通りである．

表 1：実験結果

評価項目	結果
コード正当性確認の厳格さ	ハードコーディング検出率 21.3%， 不要コード検出率 17.59%向上
フィードバック応答時間	平均応答時間が 58%短縮（3.6 秒→1.5 秒）
API コールコスト	最大 8.53%削減
フィードバックの質	正確性 4.7/5，実用性 4.6/5，建設的な表現 4.8/5

実験結果から得られた知見は以下の通りである．

コード正当性確認モジュールの精度向上によって，提出されたコードにフィードバックが必要ない場合にプロセスを終了することで，不必要な API コールを削減し，コストと処理時間を効率化することが出来た．

また，レビューコメント生成モジュールにおいて，以下 4 つの内容を設定することで，フィードバックの質向上に繋がったと考えられる．1 つ目が，プロンプトでのロール設定である．「建設的で優しく」や「批判的にならず」といった表現を要求する事で学習者により沿ったレビューコメントを提示する．また，GPT-4 に「コードレビューエージェント」としての役割を設定することで，学習者に対する応答をコードレビューに特化させ，より質の高いフィードバックを返答させる．2 つ目が，マークダウン形式にすることである．コメントを箇条書きにすることでコメントが見やすく表示され，学習意欲向上に繋がる．3 つ目が，正解コードを含まないことである．正解コードや直接的な修正方法は提供せず，学習者自身の思考を促す．4 つ目が，レビューコメントの具体例を提示することである．理想的なレビューコメントの具体例を提示することで，生成結果の質の安定化に繋がる．

これらの要因によって，レビューコメントが一貫性を持ち学習者にとって分かりやすく建設的な表現となったと考えられる．

2.2.5 課題

しかしながら，表 1 では改良による精度向上についてのみ記載されているが，実験結果から，GPT-4 のような LLM を利用したシステムでは，以下のような課

題も残っていると考えられる．誤って学習者に直接正解コードを提示してしまう可能性があることと，学習者のレベルを考慮しないフィードバックの生成である．正解コードを含まないよう設定するだけでは確実性がなく，意図しないフィードバックを提示してしまう可能性があることである．これを防ぐ為のプロンプト設計やシステム設計のさらなる改善が必要である．また，フィードバック内容の対象者レベルについて明記していないため，難解なフィードバックと冗長なフィードバックが混在してしまい，学習者の理解をおくらせてしまうと考えられる．

2.3 本研究の意義

既存のプログラミング学習支援システムやこれまでの先行研究では，直接正解のコードを提示するか，もしくは学習者の自立的思考を促すために提示しないよう設定されていた．しかしながら，LLM を用いた正解コードの判定において，1 度提示しないよう設定するだけでは，意図せず正解コードに近いフィードバックを提示してしまう可能性がある事に加え，シングルエージェント型では再度正解コードを含まないよう指示しても含まれてしまう．また，フィードバック内容を学習者のレベルを考慮しないものとして生成するため，学習効果の低下を招く可能性がある． LLM が 1 度生成したフィードバックに正解コードが含まれていないか判定するエージェントを追加することで，正解コード提示の可能性を低下させることが出来るのではないかと考える．また，システム構築時に学習者のレベルを初学者としてプロンプトを作成することで一貫したフィードバックを提示出来るのではないかと考える．

第3章 提案手法

3.1 マルチエージェント型のプログラミング学習支援システム

正解コードを含まないようにするために，正解コード検出用とフィードバック生成用の 2 つのエージェントを用いたマルチエージェント型のプログラミング学習支援システムを提案する．以下の図 1 がシステム構成図である．システムの流れとしては以下の通りに行う．

・ユーザーが問題文，コードを入力して送信
・ChatGPT がコードに対するフィードバックを作成（FAgent）
・ChatGPT がフィードバックに正解コードが含まれているかチェック（CAgent）
・含まれていた場合, ChatGPT に再送信して, フィードバックを修正（FAgent）
・最終的なフィードバックをブラウザに表示

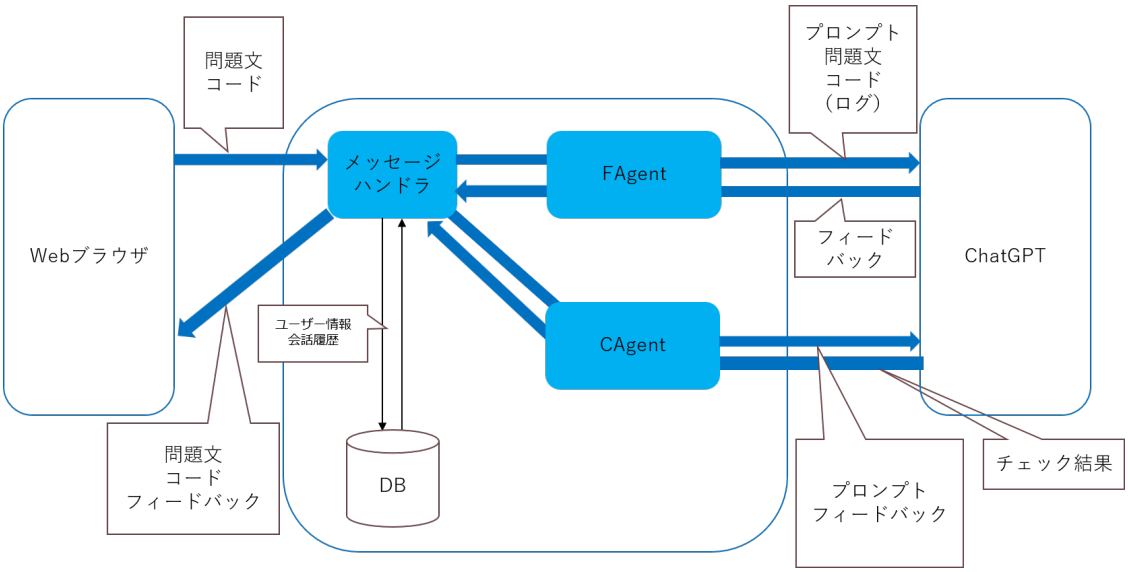

図 1：実験結果

次に，各エージェントについての詳細を記述する．

3.1.1 チェックエージェント

チェックエージェントは，ChatGPT が生成したフィードバックに対して正解コードの検出を行うエージェントである．作成したプロンプトは付録 A.2 の「フィードバックのチェックを行うクラス」に記載している．プロンプトに含んだ主な要件は以下の通りである．

・フィードバックに正解コードが含まれているかどうかを「True」「False」で判定
・含んではいけない正解コードの例
・含んでも良い正解コードの例

3.1.2 フィードバックエージェント

フィードバックエージェントは，学習者がフィードバックの生成を行うエージェントである．作成したプロンプトはチェックエージェント同様付録 A.2 の「フィードバックを生成するクラス」に記載している．プロンプトに含んだ主な要件は以下の通りである．

・問題文，コードに対してフィードバックを生成
・学習の対象者の定義（初学者）
・フィードバック生成における条件
・提示して良いフィードバック，悪いフィードバックの例
・理想的なフィードバックの例

ここで，重要となるのがフィードバック生成における条件である．正解コードを含まないための記述方法，学習者のレベルに合わせたフィードバックの生成等をここで指定している．条件については表 2 に記述している通りである．

表 2：フィードバック生成における条件

条件番号	内容
1	日本語で返答する
2	献身的な表現で返答し，最後に「質問はありますか？」や「頑張ってください」，「又質問してください」等コメントする
3	改行等を活用して，見やすい形式で回答する
4	修正点がある場合は，何行目のコードに問題があるかを明記する．
5	修正点がない場合はないことを最初に伝え，追加で冗長なコードの整理案を提示する
6	学習者が考える余地を残すため，コピーアンドペーストするだけで正解となってしまうような正解コードは，提示しない
7	初学者は関数名や使い方を知らない可能性が高いので，range や None，with open 等の関数名を提示しても良い．その際，問題文に関連のある使い方を提示してはいけない

3.2 機能

本システムでは，学習者毎に個別学習を行い，学習者と本システムとの対話内容を評価に用いるためログイン，会話履歴保存機能を実装する．これらの機能を実現するに当たって，ログイン情報，会話履歴のデータの保存先として，データベース管理システムである MySQL を利用する．また，学習者が新しいプログラミング課題に取り組む際に，会話履歴をリセット出来るよう会話内容リセット機能を追加する．

3.2.1 ログイン，会話履歴保存機能

学習者が自身のアカウントを作成しログインすることで，他学習者の会話内容と干渉することなく，学習者毎の進捗状況を管理し，個別学習を実現する．会話終了後は，ログアウト可能にする．会話履歴は，過去の会話履歴を確認できるようにする等，今後の拡張のためデータベースに保存する．アカウント情報，会話履歴はそれぞれ表 3，4 のテーブル構造でデータベースに保存する．

表 3：アカウント情報テーブル（users）

Field	Type	Null	Key	Default	Extra
Id	Int	NO	PRI	NULL	auto_increment
username	varchar(255)	NO	UNI	NULL	
password	varchar(255)	NO		NULL	
role	enum('user','admin')	NO		user	

表 4：会話履歴テーブル（conversation_log）

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
question	text	NO		NULL	
code	text	NO		NULL	
feedback	text	NO		NULL	
created_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERETED

3.2.2 会話履歴リセット機能

学習者が新しい課題に取り組む他に，学習内容をリセットして再スタートしたい場合に利用できる機能である．元々問題文の入力は会話の最初でしか入力できなかったが，「会話履歴をリセットする」ボタンを押すことで，新しい問題文を入力することが出来る．本機能を追加した理由としては，会話回数が増え会話履歴が長くなると，特定の情報や会話部分を素早く見つける事が難しくなる事に加え，会話履歴を引き継いで ChatGPT に送信するため，次第にフィードバック取得までにかかる時間が増える可能性があるためである．

3.3 動作画面

本章では，前章で説明したプログラミング学習支援システムを実際どのように学習者が利用するのかについて記述する．図 2 は初期のログイン画面であり，ログインを行うと，図 3 の各学習者の問題文，コード入力画面に遷移する．

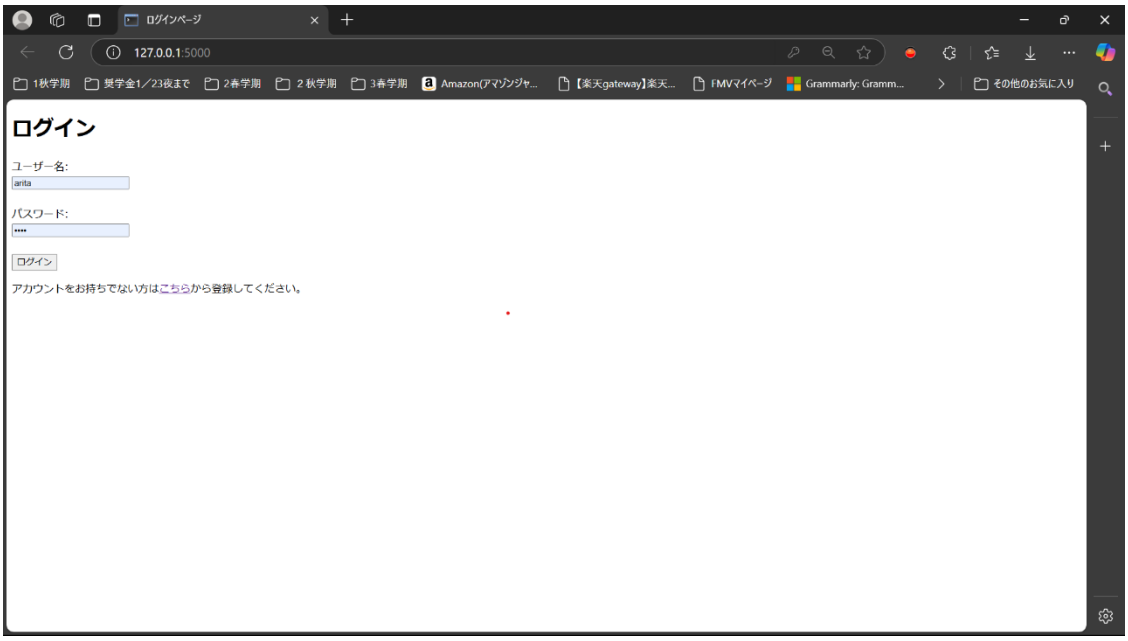

図 2：ログイン（初期）

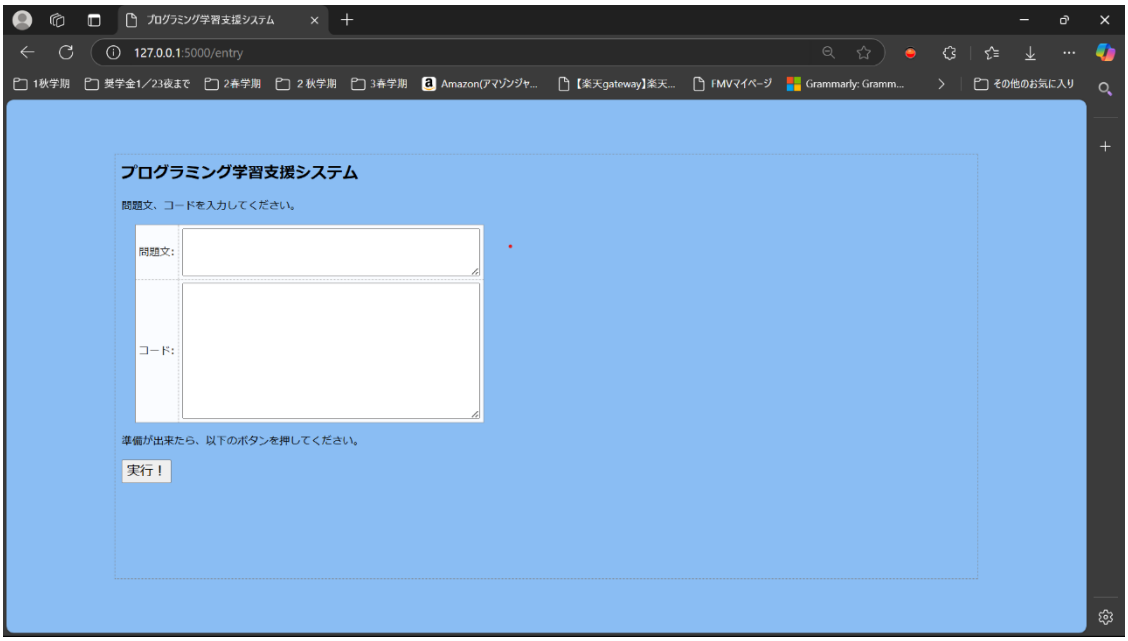

図 3：問題文，コード入力 1

3.3.1 会話

図 4 は図 3 と同様，問題文，コード入力画面である．問題文，コード入力後に実行ボタンを押すことでそれらを ChatGPT に送信し，図 5 に示すようにフィードバックが表示される．送信結果画面（図 5）には，送信内容（問題文，コード）とそれに対するフィードバック，フィードバックを参考にして新しいコードを入力するためのコード入力欄，同問題文に対する会話履歴が表示される．コードを入力し，再度送信ボタンを押すことで 2 度目のフィードバックを受け取ることが出来る．また，3．2 章で記述した会話履歴リセットボタン，ログアウトボタンについては本画面で利用可能である．

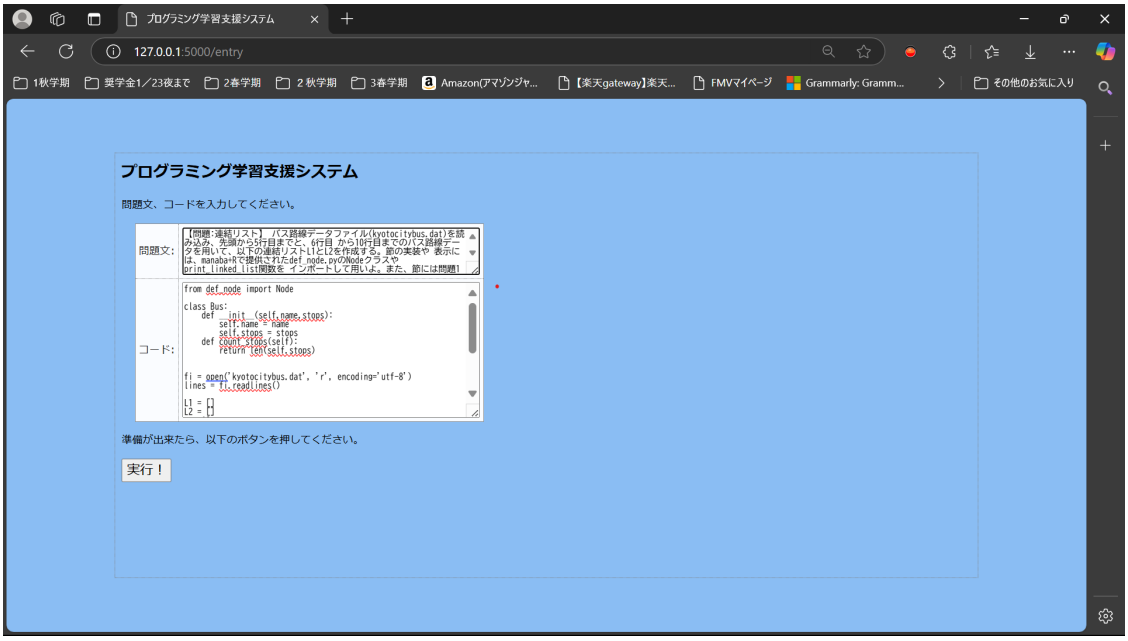

図 4：問題文，コード入力 2

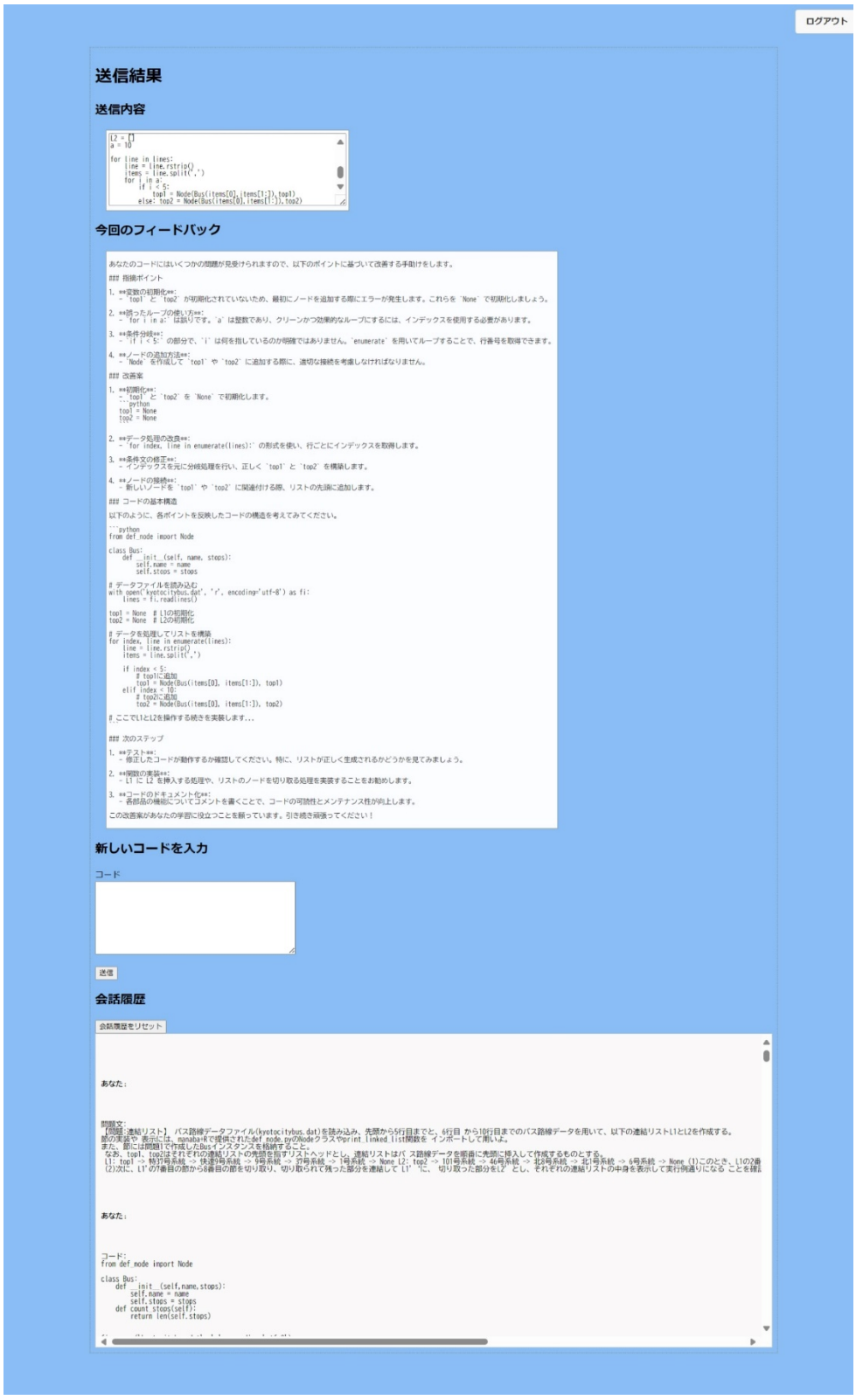
ログアウト
送信結果
送信内容
L2 = []
a = 10
for line in lines:
line = line.rstrip()
items = line.split(",")
for i in a:
if i < 5:
top1 = Node(Bus(items[0],items[1:]),top1)
else: top2 = Node(Bus(items[0],items[1:]),top2)
今回のフィードバック
あなたのコードにはいくつかの問題が見受けられますので、以下のポイントに基づいて改善する手助けをします。
指摘ポイント
1. **変数の初期化**:
- `top1` と `top2` が初期化されていないため、最初にノードを追加する際にエラーが発生します。これらを `None` で初期化しましょう。
2. **誤ったループの使い方**:
- `for i in a:` は誤りです。`a` は整数であり、クリーンかつ効率的なループにするには、インデックスを使用する必要があります。
3. **条件分岐**:
- `if i < 5:` の部分で、`i` は何を指しているのか明確ではありません。`enumerate` を用いてループすることで、行番号を取得できます。
4. **ノードの追加方法**:
- `Node` を作成して `top1` や `top2` に追加する際に、適切な接続を考慮しなければなりません。
改善案
1. **初期化**:
- `top1` と `top2` を `None` で初期化します。
```python
top1 = None
top2 = None
```
2. **データ処理の改良**:
- `for index, line in enumerate(lines):` の形式を使い、行ごとにインデックスを取得します。
3. **条件文の修正**:
- インデックスを元に分岐処理を行い、正しく `top1` と `top2` を構築します。
4. **ノードの接続**:
- 新しいノードを `top1` や `top2` に関連付ける際、リストの先頭に追加します。
コードの基本構造
以下のように、各ポイントを反映したコードの構造を考えてみてください。
```python
from def_node import Node
class Bus:
def __init__(self, name, stops):
self.name = name
self.stops = stops
# データファイルを読み込む
with open('kyotocitybus.dat', 'r', encoding='utf-8') as fi:
lines = fi.readlines()
top1 = None  # L1の初期化
top2 = None  # L2の初期化
# データを処理してリストを構築
for index, line in enumerate(lines):
line = line.rstrip()
items = line.split(",")
if index < 5:
# top1に追加
top1 = Node(Bus(items[0], items[1:]), top1)
elif index < 10:
# top2に追加
top2 = Node(Bus(items[0], items[1:]), top2)
# ここでL1とL2を操作する続きを実装します...
```
次のステップ
1. **テスト**:
- 修正したコードが動作するか確認してください。特に、リストが正しく生成されるかどうかを見てみましょう。
2. **関数の実装**:
- L1 に L2 を挿入する処理や、リストのノードを切り取る処理を実装することをお勧めします。
3. **コードのドキュメント化**:
- 各部品の機能についてコメントを書くことで、コードの可読性とメンテナンス性が向上します。
この改善案があなたの学習に役立つことを願っています。引き続き頑張ってください！
新しいコードを入力
コード
送信
会話履歴
会話履歴をリセット
あなた：
問題文:
【問題：連結リスト】 バス路線データファイル(kyotocitybus.dat)を読み込み、先頭から5行目までと、6行目 から10行目までのバス路線データを用いて、以下の連結リストL1とL2を作成する。
部の実装や 表示には、 nanaba+Rで提供された def node_py のNodeクラスや print_linked_list関数を インポートして用いよ。
また、 部には問題1で作成したBusインスタンスを格納すること。
なお、top1、top2はそれぞれの連結リストの先頭を指すリストヘッドとし、連結リストはバス 路線データを順番に先頭に挿入して作成するものとする。
(1) top1 -> 特万号系統 -> 快速8号系統 -> 6号系統 -> 竹8号系統 -> 1号系統 -> None L2: top2 -> 101号系統 -> 66号系統 -> 北9号系統 -> 北1号系統 -> 6号系統 -> None (1)このとき、L1の2番
(2)次に、L1'の7番目の部から8番目の部を切り取り、切り取られて残った部分を連結して L1' に、 切り取った部分をL2' とし、それぞれの連結リストの中身を表示して実行例通りになる ことを確認
あなた：
コード:
from def_node import Node
class Bus:
def __init__(self,name,stops):
self.name = name
self.stops = stops
def count_stops(self):
return len(self.stops)

図 5：送信結果

第4章 評価/考察

4.1 評価方法

提案手法を用いたプログラミング学習支援システムの評価にはテストケースを用いる．テストケースとして動作確認に使用するものと同じ問題文を用意し，それに対する修正箇所が複数存在する学習者の入力コード，理想のフィードバックを作成する．この問題文を用いて 30 回フィードバックを生成し，正解コードの検出精度，フィードバックとテストケースの比較を行う．また，CAgent の有無による正解コード含有数の比較も行うことで，CAgent が大規模言語モデルに基づくプログラミング学習支援において有用かどうか評価する．テストケースを用いた評価については，第 2 章で説明した各課題に対して，以下に記述する項目で行う．

正解コードの検出

表 5：評価項目（正解コードの検出）

評価項目	詳細
検出精度	正解コードが含まれている場合に，「True」を返答する割合.

学習効果を高めるフィードバックの生成

表 6：評価項目（フィードバックの生成）

評価項目	詳細
問題指摘精度	コード内のエラーを正確に指摘できている割合.
献身性	コメントが学習者にとって批判的な内容となっていないか.
具体性	修正方法のヒントを提示しているか.
難易度	学習者のレベルに応じた理解しやすい内容となっているか.
正解コードの有無	フィードバックに正解コードが含まれている割合

4.2 テストケース

使用するプログラミング課題を図 6，付属ファイルを図 7，付録 A.1，解答サンプルを図 8，入力するコードを図 9，期待されるフィードバックを図 10 に示す.

【問題:連結リスト】 バス路線データファイル(kyotocitybus.dat)を読み込み，先頭から 5 行目までと，6 行目 から 10 行目までのバス路線データを用いて，以下の連結リスト L1 と L2 を作成する．節の実装や 表示には，manaba+R で提供された def_node.py の Node クラスや print_linked_list 関数を インポートして用いよ．また，節には問題 1 で作成した Bus インスタンスを格納すること． なお，top1，top2 はそれぞれの連結リストの先頭を指すリストヘッドとし，連結リストはバ ス路線データを順番に先頭に挿入して作成するものとする． L1: top1 -> 特 37 号系統 -> 快速 9 号系統 -> 9 号系統 -> 37 号系統 -> 1 号系統 -> None L2: top2 -> 101 号系統 -> 46 号系統 -> 北 8 号系統 -> 北 1 号系統 -> 6 号系統 -> None (1)このとき，L1 の 2 番目の節の直前に L2 を挿入して，二つの連結リストを合併し, 合併 した連結リスト L1'の中身を表示して実行例通りになることを確認せよ． (2)次に，L1'の 7 番目の節から 8 番目の節を切り取り，切り取られて残った部分を連結して L1'‘に， 切り取った部分を L2’とし，それぞれの連結リストの中身を表示して実行例通りになる ことを確認せよ.

図 6：テストケース（問題文）

```
    # Node クラスの定義
    class Node:
        def __init__(self, data, next=None):  # next にデフォルト値（何も値が渡されなかった時に設定される値）を設定
            self.data = data  # データを格納するインスタンス変数
            self.next = next  # 連結される次の節を指すインスタンス変数

    def print_linked_list(top):
        message = '{}の停車するバス停数は{}箇所です．'
        while top is not None:  # リストヘッド top が連結リストの終端でない限り以下の処理を繰り返す
            print(message.format(top.data.name, top.data.count_stops()))
            top = top.next  # リストヘッド top を次の節に一つ進める
```

図 7：テストケース（def_node.py）

```
from def_node import Node, print_linked_list

class Bus:
    def __init__(self, name, stops):
        self.name = name
        self.stops = stops

    def count_stops(self):
        return len(self.stops)

# リストから連結リストを作成する関数（返り値は先頭のリストヘッド）
def make_linked_list(lst):
    top = None
    for line in lst:
        line = line.rstrip()
        items = line.split(',') # 1行を半角スペースで区切って items リストに代入
        top = Node(Bus(items[0], items[1:]), top)
        # 新規作成した Node インスタンスを先頭（top）の前に挿入し，その Node インスタンスに先頭リストヘッド(top)を移動
        return top

fi = open('kyotocitybus.dat', 'r', encoding='utf-8')
lines = fi.readlines()
top1 = make_linked_list(lines[:5])
top2 = make_linked_list(lines[5:10])
fi.close()

# (1)
top2.next.next.next.next.next = top1.next
top1.next = top2
top2 = None
print_linked_list(top1)

# (2)
top2 = top1.next.next.next.next.next.next
p = top1.next.next.next.next.next.next.next
top1.next.next.next.next.next.next = p.next
p.next = None
print_linked_list(top1)
print_linked_list(top2)
```

図 8：テストケース（解答サンプル）

```
from def_node import Node

class Bus:
    def __init__(self,name,stops):
        self.name = name
        self.stops = stops
    def count_stops(self):
        return len(self.stops)

fi = open('kyotocitybus.dat', 'r', encoding='utf-8')
lines = fi.readlines()

L1 = []
L2 = []
a = 10

for line in lines:
    line = line.rstrip()
    items = line.split(',')
    for i in a:
        if i < 5:
            top1 = Node(Bus(items[0],items[1:]),top1)
        else: top2 = Node(Bus(items[0],items[1:]),top2)
```

図 9：テストケース（入力コード）

修正点：

学習者のコードにはいくつか修正が必要な部分があります．それぞれについて，問題点と改善方法を解説します．

1. for i in a の部分の問題

問題点：

a = 10 は整数型であり，for i in a のように直接ループで使用することはできません．リストや範囲を明確に指定する必要があります．

改善の考え方：

もし a が回数を意味しているなら，数値を基に範囲を指定することでループが実現できます．これには，数値から反復可能な範囲を作る仕組みを利用します．範囲とは，「開始点から終了点までの数値」を生成するものです．

2. top1 と top2 の初期化がされていない問題

問題点：

top1 や top2 は最初に使用する前に「初期値」を持たせておく必要があります．しかし，コード内では初期化されていないため，エラーが発生します．

改善の考え方：

top1 や top2 は「連結リストの先頭」を示す変数です．連結リストがまだ空の場合，

先頭を示す変数には何も存在しないことを意味する値を代入します．この「何も存在しないこと」を示す値を特別に設定することで，連結リストを初期化できます．

3. 条件分岐の使い方についての問題

問題点：

if i < 5 の条件が適切に動作するか確認が必要です．i の値がどのように決定されるかを明確にし，条件を正しく書く必要があります．

改善の考え方：

i の値はループの中で生成されます．その値が 5 未満かどうかを判定し，正しく top1 と top2 に追加するよう条件を設計することが重要です．

4. リストを使った値の分類の不必要性

問題点：

L1 や L2 を作成していますが，これらを使用せずに直接 Node オブジェクトを構築しています．そのため，これらのリストは不要です．

改善の考え方：

L1 や L2 を削除し，直接連結リストの構築に集中するとコードが簡潔になります．

5. Bus クラスのオブジェクトを連結リストに追加する書き方の問題

問題点：

連結リストは `top1` や `top2` に新しい要素を追加し続ける形式にする必要があります．今のコードでは，毎回新たに `Node` を作成し続けているため，構造が壊れてしまいます．

改善の考え方：

新しい `Node` を追加する際には，現在の `top1` や `top2` の参照を更新する必要があります．

全体的な整理案

修正点を整理すると，以下のように改善できます：

for i in a の使い方を明確にし，範囲を生成する仕組みを利用します．

top1 と top2 を最初に空の連結リストとして適切に初期化します．

条件分岐を，ループ内での処理の流れに従って正しく設定します．

不要なリスト（L1, L2）を削除し，簡潔なコードにします．

連結リストを新しい要素を追加し続ける形式にします．

アドバイス

これらのポイントを基にコードを調整してみてください．また，理解を深めるためには，それぞれの改善点がどのような結果をもたらすかを確認しながら進めることが大切です．

他にも質問があれば気軽に教えてくださいね！

図 10：テストケース（理想のフィードバック）

4.3 評価

第 4.1 章で記述した評価方法に基づき，正解コードの検出と学習効果を高めるフィードバックの評価を行う．学習者が入力したコードに対して 30 回フィードバックを生成し，正解コードが含まれている場合の検出精度を計測する．次に，生成されたフィードバックの質を問題指摘精度，献身性，具体性，難易度，正解コードの有無で評価する．また，チェックエージェントの有無で，フィードバックに正解コードが含まれる数の比較を行う．これにより，提案手法が大規模言語モデルに基づくプログラミング学習支援において，どの程度有益かを明らかにする．以下に各項目における評価結果を記述する．

4.3.1 正解コードの検出

検出精度

動作件数 30 件中，最初に生成されたフィードバックに正解コードが含まれていたのは 25 件で，含まれていなかったのは 5 件である．その中で，正確に「True」と判定されたのは 15 件で，「False」と判定されたのは 10 件であった．このことから，判定率は 60%となった（表 7）．「正解コードが含まれているフィードバック」の判定が 60%であった一方，「正解コードが含まれないフィードバック」，つまり「正解コードの数が 0 であるフィードバック」の判定結果に注目すると，100%で「False」と判定され動作件数 30 件中での精度は完璧と言える．

表 7：評価（正解コードの検出）

動作回数（回目）	正解コードの数	判定結果
1	2	T
2	3	T
3	2	T
4	1	F
5	0	F
6	2	F
7	2	T
8	0	F
9	1	F
10	3	T
11	1	T
12	0	F
13	1	F
14	2	F
15	1	F

16	1	F
17	1	F
18	1	F
19	1	T
20	1	F
21	2	T
22	1	T
23	3	T
24	2	T
25	1	T
26	1	T
27	0	F
28	2	T
29	0	F
30	2	T

4.3.2 学習効果を高めるフィードバックの生成

問題指摘精度

図 10 の期待される理想のフィードバックと提示された修正点の数により評価を行った（表 8）．入力コードに必要な修正点が 5 つであるのに対して，各動作で提示された修正点の数が 3 個または 4 個でおよそ半々となり，全体の平均値として約 3.47 個という結果が得られた．動作ごとに提示された修正箇所自体は概ね安定しており，ばらつきは見られなかった．すべての動作において，修正点の提示数が実際の修正点の数を下回っていることから，特定の問題点が見落とされるケースがあることが分かる．

表 8：評価（問題指摘精度）

動作回数（回目）	実際の修正点（個）	提示された修正点（個）
1	5	3
2	5	3
3	5	4
4	5	3
5	5	4
6	5	4
7	5	3
8	5	4
9	5	3
10	5	4
11	5	3
12	5	4
13	5	4
14	5	3
15	5	3
16	5	3
17	5	4
18	5	3
19	5	3
20	5	4
21	5	3
22	5	3
23	5	3
24	5	4
25	5	4
26	5	4
27	5	4
28	5	3
29	5	3
30	5	4

献身性

学習者が作成したコードを極端に否定することはなく，また，すべてのフィードバックにおいて，「頑張ってください」，「他にも質問があれば気軽に教えてください」等のコメントが含まれており，単なる指摘に終わらず献身的な表現が含まれていた．問題点を指摘する際にも，否定的な表現を避け，前向きな表現で改善を促す構成となっていた．

具体性

問題箇所の説明に加え，修正方法を具体的に提示しているかで評価を行った．各フィードバックには，修正点の指摘に加えて「改善の考え方」という項目が含

まれ，修正方法のヒントが提示された．特に，コードスペニットを用いて短いコードのサンプルや例を提示していた．エラーの原因が記載されているフィードバックが多かった一方で，問題箇所の行数の指定はあるものとないものでばらつきが確認された．また，中級者や上級者向けの挑戦的な表現が使われることもなく，初学者向けのフィードバックを生成していると言える．全体として，フィードバックの献身性は非常に高く，学習者の心理的な負担を軽減する評価出来る．

難易度

基本的にどのフィードバックでも専門用語を最低限に抑えた簡潔な表現が用いられていた．また，修正方法の提示以外にも，初学者向けにエラーの原因を簡潔に説明していた．具体的には「for i in a」のエラーについて，「a=10 は整数型であり，for i in a のように直接ループで使用することはできません.」というシンプルな説明となっていた．

正解コード数

動作回数ごとのフィードバックに含まれる正解コードの数を表 9 に示す．

正解コードの数は動作ごとに異なり，0 個から最大 3 個となっている．フィードバックに正解コードが含まれていないケースは 30 件中 7 件しかなく，また，数に大きなばらつきがある点が特徴であり，提案手法での正解コードを含まないフィードバック生成には確実性がないことが読み取れる．

表 9：評価（正解コードの数）

動作回数（回目）	正解コードの数
1	1
2	3
3	1
4	1
5	0
6	2
7	2
8	0
9	1
10	3
11	2
12	0
13	1
14	2
15	1
16	0
17	1
18	1
19	1
20	1
21	1
22	1
23	3
24	2
25	2
26	2
27	0
28	0
29	0
30	2

4.3.3 CAgent の有無による比較

各動作回数において，CAgent の有無が，提示されたフィードバック内の正解コードの数にあたえる影響の比較を行った（表 10）．30 件中 22 件が正解コード数に変化なし，5 件が減少，3 件が増加という結果となった．また，CAgent 無しの場合の正解コード総数が 40 個，CAgent 有りの場合の正解コード総数が 37 個となり，CAgent 有りの場合の方が 1 割程度減少している．しかし，両条件ともに総数はほぼ同値であり，平均をとるとそれぞれ約 1.3 件，約 1.23 件となり，大きな差異は見られなかった．

表 10：評価（正解コード数の比較）

動作回数（回目）	正解コード数（CAgent 無し）	正解コード数（CAgent 有り）
1	2	1
2	3	3
3	2	1
4	1	1
5	0	0
6	2	2
7	2	2
8	0	0
9	1	1
10	3	3
11	1	2
12	0	0
13	1	1
14	2	2
15	1	1
16	1	0
17	1	1
18	1	1
19	1	1
20	1	1
21	2	1
22	1	1
23	3	3
24	2	2
25	1	2
26	1	2
27	0	0
28	2	0
29	0	0
30	2	2
合計	40	37

4.4 考察

まず，大規模言語モデルに基づくプログラミング学習支援での正解コードの検出において，提案手法での検出は，60%の確率で認められた．含んではいけない正解コードの例を複数用意した上で判定させることで似たようなコードの提示を検出可能になったのではないかと考えられる．一方で，CAgent に含んだデータが生成されたフィードバックとプロンプトのみであったため，正解コードの判定が難しく判定率 60%にとどまったと考えられる．本研究で CAgent 用に

用意した正解コードの例と例外は合わせて 3 つのみでサンプル数が不足していた．また，例の内容についても問題文とその正解コードではなく，「〜に変更することで修正可能.」のように正解コードのみの指定方法であった為，問題文の情報が無く，正確に正解コードを検出出来なかったのではないかと考えられる．

また，学習効果を高めるフィードバックの生成の問題指摘精度において，必要な修正点の数に対して，半数以上修正点を提示可能である事が認められた．これは，ChatGPT 自体が膨大なデータを基に学習している事が起因していると考えられる．ChatGPT は一般的で明確なエラーは指摘可能だが，非効率的なコードやエラーではないが不必要なコードが含まれていたときに修正点として認識しない可能性がある為である．加えて，情報量が多すぎると回答を簡略化してしまう可能性もある為である．献身性，難易度に関しては基本的に今後課題となる様な箇所は見受けられず，ほぼ期待通りのフィードバックを生成出来た．フィードバックの最後のコメントに関しては，「頑張ってください」と表示されることが多くバリエーションが少ないと感じたが，献身性の指定時に例の数が少なかった事で発生してしまったと考えられる．次に，正解コード数，CAgent の有無による比較において，各動作の正解コード数に大きくばらつきがある事が認められた．CAgent と同様，FAgent 用に正解コードの例として用意したものが理想的なフィードバックも含めて 5 つのみで，かつ例の問題文を含めなかったことが原因であると考えられる．また，CAgent 使用後のフィードバック再生成の際に使用するプロンプトが一回目の送信時に使用するプロンプトを共通部分が多かったことから，CAgent の有無による変化が少なくなってしまったと考えられる．

第5章 おわりに

本研究で行った正解コードの検出について，正解コード例の数と情報量が十分でないことで正解コード検出の精度が低くなってしまい，より精度を向上させるには，さらに十分なデータ数と情報量のものを用意し評価を行う必要があると考える．しかし，情報量が極端に増えると，フィードバック生成までに時間がかかってしまい，学習者の学習意欲を低下させてしまう可能性がある．また，ChatGPT のトークン数上限に引っかかってしまう可能性も考えられる．そのため，CAgent に問題文，コード，会話履歴の情報をどれだけ渡すのか，正解コード例の適切な量を試行錯誤し構築を行っていきたいと考える．

また，学習効果を高めるフィードバックの生成について，期待されるフィードバック，提示してはいけない例に関する情報量が十分でないことで ChatGPT が提示してはいけない正解コードを正確に判断出来なかったと考えられる．また，送信時，再送信時のプロンプトに共通点が多かったことから，CAgent で正解コードを検出した後もフィードバックの質向上に繋がらなかったと考える．そのため，例の情報量や種類を増やし再送信時にはフィードバック生成よりも正解コードを含まないことに集中させることで，フィードバックの質向上を行っていきたいと考える．

謝辞

最後に，本研究を行うにあたり，熱心なご指導,ご助言を賜りました，村上陽平教授に深謝申し上げます．また，普段からお世話になっている社会知能研究室の皆様に心より感謝申し上げます．

参考文献

[1] Lee Dong-Kyu: A GPT-based Code Review System for Programming Language Learning, arXiv:2407.04722 [cs.SE], submitted on 21 June 2024

[2] 住田智雄， 初学者を対象としたプログラミング学習支援システムの基本堤機能の実装と評価，情報教育，Vol. 4 pp. 47–54 (2024).

[3] 渋沢良太，大規模言語モデルを利用した非解答型プログラミング学習支援の試行，第一工科大学研究報告，36号，p.23-30(2025-05)

[4] 奥田勝己，Saman Amarasinghe : 大規模言語モデルを用いたプログラミングを支援する言語機構，日本ソフトウェア科学会第40回大会公演連文集，40th，ROMBUNNO.38-R(2023).

[5] 上地泰彰，田丸恵理子，渡邊紀文 : マルチターンで学生の問いを解決するプログラミング学習支援チャットボットの試作，Musashino University Smart Intelligence Center 紀要，第5号，pp.101-119(2024)

付録：ソースコード

A.1 テストケースの付属ファイル

Kyotocitybus.dat

1 号系統,西賀茂車庫前,神光院前,大宮総門口町,山ノ前町,玄琢下,紫野泉堂町,旭ヶ丘,佛教大学前,千本北大路,船岡山,建勲神社前,大徳寺前,北大路堀川,北大路新町,北大路バスターミナル,植物園前,府立大学前,洛北高校前,一本松,下鴨神社前,糺ノ森,新葵橋,葵橋西詰,河原町今出川,出町柳駅前,北大路駅前

37 号系統,西賀茂車庫前,神光院前,大宮総門口町,北大路堀川,北大路新町,北大路バスターミナル,葵橋西詰,河原町今出川,大宮田尻町,上賀茂御薗橋,加茂川中学前,下岸町,上堀川,東高縄町,下鳥田町,下総町,松ノ下町,出雲路橋,出雲路俵町,出雲路神楽町,府立医大病院前,荒神口,河原町丸太町,京都市役所前,河原町三条,三条京阪前,四条河原町,四条京阪前,北大路駅前

9 号系統,西賀茂車庫前,神光院前,大宮総門口町,北大路堀川,大宮田尻町,上賀茂御薗橋,加茂川中学前,下岸町,上堀川,東高縄町,下鳥田町,堀川鞍馬口,天神公園前,堀川寺ノ内,堀川上立売,堀川今出川,一条戻り橋,堀川中立売,堀川下長者町,堀川下立売,堀川丸太町,二条城前,堀川御池,堀川三条,堀川蛸薬師,四条堀川,堀川松原,堀川五条,西本願寺前,七条堀川,下京区総合庁舎前,京都駅前

快速 9 号系統,西賀茂車庫前,神光院前,大宮総門口町,北大路堀川,大宮田尻町,上賀茂御薗橋,加茂川中学前,下岸町,上堀川,東高縄町,下鳥田町,堀川寺ノ内,堀川今出川,堀川中立売,堀川丸太町,堀川御池,四条堀川,堀川五条,七条堀川,京都駅前

特 37 号系統,西賀茂車庫前,神光院前,大宮総門口町,北大路堀川,北大路新町,北大路バスターミナル,下岸町,上堀川,東高縄町,下鳥田町,大宮小野堀町,大宮交通公園前,大宮大門町,下竹殿町

6 号系統,玄琢下,紫野泉堂町,旭ヶ丘,佛教大学前,千本北大路,四条大宮,ライトハウス前,千本鞍馬口,乾隆校前,千本上立売,千本今出川,千本中立売,千本出水,千本丸太町,出世稲荷前,二条駅前,千本三条・朱雀立命館前,みぶ操車場前,北木ノ畑町,鷹峯上ノ町,土天井町,鷹峯源光庵前,釈迦谷口,玄琢

北 1 号系統,玄琢下,紫野泉堂町,旭ヶ丘,佛教大学前,北大路堀川,北大路新町,北大路バスターミナル,上堀川,東高縄町,下鳥田町,北木ノ畑町,鷹峯上ノ町,土天井町,鷹峯源光庵前,釈迦谷口,玄琢,下緑町,常徳寺前

北 8 号系統,紫野泉堂町,旭ヶ丘,佛教大学前,千本北大路,船岡山,建勲神社前,大徳寺前,北大路堀川,北大路新町,北大路バスターミナル,植物園前,府立大学前,洛北高校前,上堀川,北山駅前,野々神町,松ヶ崎駅前,修学院道,一乗寺清水町,一乗寺下り松町,一乗寺木ノ本町,下鴨東本町,高木町,高野橋東詰,高野,北大路駅前,下緑町,常徳寺前,東元町,一乗寺地蔵本町,一乗寺梅ノ木町,松ヶ崎橋,松ヶ崎大黒天,松ヶ崎海尻町,植物園北門前,北山橋東詰,元町

46 号系統,佛教大学前,千本北大路,加茂川中学前,下岸町,四条堀川,大宮交通公園前,大宮大門町,下竹殿町,四条大宮,四条西洞院,四条烏丸,四条高倉,四条河原町,上賀茂神社前,牛若,紫野上野町,今宮神社前,ライトハウス前,千本鞍馬口,乾隆校前,千本上立売,千本今出川,千本中立売,千本出水,千本丸太町,出世稲荷前,二条駅前,千本三条・朱雀立命館前,みぶ操車場前,神宮道,東山三条,四条京阪前,祇園,知恩院前,東山仁王門,京都会館美術館前

101 号系統,千本北大路,大徳寺前,北大路堀川,北大路バスターミナル,堀川今出川,堀川丸太町,二条城前,堀川御池,四条堀川,四条烏丸,千本今出川,烏丸五条,北野白梅町,北野天満宮前,金閣寺道,今出川大宮,わら天神前,京都駅前

102 号系統,千本北大路,大徳寺前,北大路堀川,北大路バスターミナル,河原町今出川,出町柳駅前,堀川今出川,百万遍,銀閣寺道,錦林車庫前,千本今出川,北野白梅町,北野天満宮前,金閣寺道,今出川大宮,烏丸今出川,わら天神前

12 号系統,千本北大路,船岡山,建勲神社前,大徳寺前,北大路堀川,堀川鞍馬口,天神公園前,堀川寺ノ内,堀川上立売,堀川今出川,一条戻り橋,堀川中立売,堀川下長者町,堀川下立売,堀川丸太町,二条城前,堀川御池,堀川三条,堀川蛸薬師,四条堀川,三条京阪前,四条西洞院,四条烏丸,四条高倉,四条河原町,東山三条,立命館大学前,衣笠総門町,金閣寺前,金閣寺道,四条京阪前,祇園,知恩院前

204 号系統,千本北大路,船岡山,建勲神社前,大徳寺前,北大路堀川,北大路新町,北大路バスターミナル,植物園前,府立大学前,洛北高校前,堀川丸太町,河原町丸太町,田中大久保町,高原町,伊織町,上終町京都造形芸大前,北白川別当町,岡崎道,岡崎神社前,東天王町,銀閣寺道,浄土寺,錦林車庫前,真如堂前,千本丸太町,北白川校前,熊野神社前,丸太町京阪前,裁判所前,烏丸丸太町,北野白梅町,丸太町智恵光院,府庁前,金閣寺道,わら天神前,衣笠校前,大将軍,北野中学前,西ノ京円町,丸太町御前通,丸太町七本松,下鴨東本町,高木町,高野橋東詰,高野,北大路駅前

205 号系統,千本北大路,船岡山,建勲神社前,大徳寺前,北大路堀川,北大路新町,北大路バスターミナル,植物園前,府立大学前,洛北高校前,一本松,下鴨神社前,糺ノ

森,新葵橋,葵橋西詰,河原町今出川,下京区総合庁舎前,府立医大病院前,荒神口,河原町丸太町,京都市役所前,河原町三条,西大路四条,四条河原町,西大路松原,西大路五条,西大路三条,西大路御池,太子道,七条大宮,九条車庫前,河原町松原,河原町五条,河原町正面,七条河原町,塩小路高倉,烏丸七条,北野白梅町,金閣寺道,わら天神前,衣笠校前,大将軍,北野中学前,西ノ京円町,西大路花屋町,西大路七条,東寺道,七条西洞院,梅小路公園前,七条千本,七条御前通,北大路駅前,京都駅前

206 号系統,千本北大路,船岡山,建勲神社前,大徳寺前,北大路堀川,北大路新町,北大路バスターミナル,植物園前,府立大学前,洛北高校前,七条堀川,下京区総合庁舎前,四条大宮,百万遍,飛鳥井町,叡電元田中,田中大久保町,大宮五条,大宮松原,東山二条,島原口,七条大宮,七条河原町,ライトハウス前,千本鞍馬口,乾隆校前,千本上立売,千本今出川,千本中立売,千本出水,千本丸太町,出世稲荷前,二条駅前,千本三条・朱雀立命館前,みぶ操車場前,東山三条,烏丸七条,京大正門前,近衛通,熊野神社前,祇園,東山七条,馬町,五条坂,清水道,東山安井,知恩院前,東山仁王門,博物館三十三間堂前,七条京阪前,下鴨東本町,高木町,高野橋東詰,高野,北大路駅前,京都駅前

59 号系統,千本北大路,河原町今出川,堀川今出川,府立医大病院前,荒神口,河原町丸太町,京都市役所前,河原町三条,三条京阪前,四条河原町,ライトハウス前,千本鞍馬口,乾隆校前,千本上立売,千本今出川,福王子,山越中町,山越,ユースホステル前,宇多野病院前,鳴滝本町,御室仁和寺,広沢池・佛大広沢校前,御室,塔ノ下町,竜安寺前,立命館大学前,衣笠総門町,金閣寺前,金閣寺道,今出川浄福寺,今出川大宮,上京区総合庁舎前,烏丸今出川,同志社前,四条京阪前

M1 号系統,千本北大路,船岡山,建勲神社前,大徳寺前,北大路堀川,北大路新町,北大路バスターミナル,立命館大学前,金閣寺道,桜木町,わら天神前,立命館西園寺記念館前,衣笠氷室町,原谷口,原谷農協前,原谷

67 号系統,北大路堀川,加茂川中学前,下岸町,上堀川,東高縄町,下鳥田町,堀川鞍馬口,天神公園前,堀川寺ノ内,堀川上立売,堀川今出川,一条戻り橋,堀川中立売,堀川下長者町,堀川下立売,堀川丸太町,二条城前,堀川御池,堀川三条,堀川蛸薬師,四条堀川,京都外大前,南広町,日新電機前,梅津段町,長福寺道,梅津西浦町,梅ノ宮神社前,松尾橋,四条葛野大路,四条中学前,西院巽町,西大路四条,四条御前通,四条中新道,壬生寺道,四条大宮,上賀茂神社前

北3 号系統,北大路バスターミナル,上賀茂御薗橋,上賀茂橋,北大路駅前,東元町,御薗口町,朝露ケ原町,柊野別れ,ゴルフ場前,京都産大前

直行号系統,北大路バスターミナル,京都産大前

4 号系統,洛北高校前,一本松,下鴨神社前,糺ノ森,新葵橋,河原町今出川,出町柳駅前,加茂川中学前,下岸町,府立医大病院前,荒神口,河原町丸太町,京都市役所前,河原町三条,四条河原町,上賀茂神社前,上賀茂橋,上賀茂菖蒲園町,上賀茂石計町,上賀茂松本町,上賀茂豊田町,深泥池,上賀茂榊田町,北山駅前,野々神町,松ヶ崎駅前,神殿町,北園町,東北園町,松ヶ崎泉川町,洛北高校正門前,河原町松原,河原町五条,河原町正面,七条河原町,塩小路高倉,京都駅前

17 号系統,河原町今出川,出町柳駅前,府立医大病院前,荒神口,河原町丸太町,京都市役所前,河原町三条,四条河原町,百万遍,銀閣寺道,浄土寺,錦林車庫前,河原町松原,河原町五条,河原町正面,七条河原町,塩小路高倉,北白川,京大農学部前,京都駅前

201 号系統,河原町今出川,出町柳駅前,堀川今出川,四条堀川,四条大宮,四条西洞院,四条烏丸,四条高倉,四条河原町,百万遍,東山二条,千本今出川,千本中立売,千本出水,千本丸太町,出世稲荷前,二条駅前,千本三条・朱雀立命館前,みぶ操車場前,東山三条,京大正門前,近衛通,熊野神社前,今出川浄福寺,今出川大宮,上京区総合庁舎前,烏丸今出川,同志社前,四条京阪前,祇園,知恩院前,東山仁王門

203 号系統,河原町今出川,出町柳駅前,堀川今出川,四条堀川,西大路四条,四条御前通,四条中新道,壬生寺道,四条大宮,四条西洞院,四条烏丸,四条高倉,四条河原町,百万遍,西大路三条,西大路御池,太子道,東山二条,岡崎道,岡崎神社前,東天王町,銀閣寺道,浄土寺,錦林車庫前,真如堂前,千本今出川,東山三条,熊野神社前,北野白梅町,北野天満宮前,上七軒,今出川浄福寺,今出川大宮,上京区総合庁舎前,烏丸今出川,同志社前,四条京阪前,祇園,大将軍,北野中学前,西ノ京円町,知恩院前,東山仁王門,北白川,京大農学部前

3 号系統,河原町今出川,出町柳駅前,四条堀川,府立医大病院前,荒神口,河原町丸太町,京都市役所前,河原町三条,京都外大前,南広町,日新電機前,梅津段町,長福寺道,梅津西浦町,梅ノ宮神社前,松尾橋,四条葛野大路,四条中学前,西院巽町,西大路四条,四条御前通,四条中新道,壬生寺道,四条大宮,四条西洞院,四条烏丸,四条高倉,四条河原町,百万遍,飛鳥井町,叡電元田中,田中大久保町,高原町,伊織町,上終町京都造形芸大前,田中樋ノ口町,北白川小倉町,北白川別当町,上池田町,北白川仕伏町

51 号系統,堀川今出川,河原町三条,四条烏丸,四条高倉,四条河原町,千本今出川,烏丸丸太町,烏丸二条,烏丸御池,烏丸三条,北野白梅町,北野天満宮前,上七軒,立命館大学前,今出川浄福寺,今出川大宮,上京区総合庁舎前,烏丸今出川,衣笠校前,堺町御池,小松原児童公園前,烏丸一条,烏丸下長者町,烏丸下立売,京都市役所前

50 号系統,堀川中立売,堀川下長者町,堀川下立売,堀川丸太町,二条城前,堀川御池,堀川三条,堀川蛸薬師,四条堀川,四条西洞院,千本今出川,千本中立売,北野白梅町,北野天満宮前,上七軒,立命館大学前,桜木町,わら天神前,衣笠校前,五条西洞院,七条西洞院,智恵光院中立売,大宮中立売,西洞院仏光寺,西洞院松原,西洞院六条,西洞院正面,京都駅前

10 号系統,堀川丸太町,河原町丸太町,京都市役所前,河原町三条,三条京阪前,四条河原町,千本今出川,千本中立売,千本出水,千本丸太町,裁判所前,烏丸丸太町,福王子,山越中町,山越,ユースホステル前,宇多野病院前,鳴滝本町,御室仁和寺,京福妙心寺駅前,妙心寺北門前,等持院南町,等持院道,府立体育館前,北野白梅町,北野天満宮前,上七軒,丸太町智恵光院,府庁前,四条京阪前

202 号系統,堀川丸太町,河原町丸太町,西大路四条,西大路松原,西大路五条,西大路三条,西大路御池,太子道,東山二条,九条大宮,九条近鉄前,九条車庫前,大石橋,千本丸太町,東山三条,熊野神社前,丸太町京阪前,裁判所前,烏丸丸太町,丸太町智恵光院,府庁前,祇園,西ノ京円町,丸太町御前通,丸太町七本松,西大路花屋町,西大路七条,西大路八条,西大路駅前,西大路九条,東寺南門前,羅城門,九条七本松,洛陽工業高校前,九条河原町,東福寺,泉涌寺道,今熊野,東山七条,馬町,五条坂,清水道,東山安井,知恩院前,東山仁王門

93 号系統,堀川丸太町,河原町丸太町,岡崎道,岡崎神社前,東天王町,錦林車庫前,真如堂前,千本丸太町,熊野神社前,丸太町京阪前,裁判所前,烏丸丸太町,丸太町智恵光院,府庁前,太秦開日町,広沢御所ノ内町,嵯峨中学前,嵯峨嵐山駅前,嵯峨瀬戸川町,嵯峨小学校前,野々宮,嵐山天龍寺前,嵐山,西ノ京円町,丸太町御前通,丸太町七本松,伯楽町,西ノ京馬代町,木辻南町,妙心寺前,花園駅前,花園扇野町,太秦映画村道,常盤・嵯峨野高校前,常磐野小学校前,太秦北路町

15 号系統,堀川御池,河原町三条,三条京阪前,四条河原町,千本丸太町,出世稲荷前,二条駅前,烏丸御池,北野白梅町,立命館大学前,四条京阪前,桜木町,わら天神前,衣笠校前,大将軍,北野中学前,西ノ京円町,丸太町御前通,丸太町七本松,神泉苑前,新町御池,堺町御池,京都市役所前

11 号系統,四条堀川,河原町三条,三条京阪前,西大路四条,四条御前通,四条中新道,壬生寺道,四条大宮,四条西洞院,四条烏丸,四条高倉,四条河原町,西大路三条,庚申前,猿田彦橋,山越中町,山越東町,太秦開日町,広沢御所ノ内町,嵯峨中学前,嵯峨嵐山駅前,嵯峨瀬戸川町,嵯峨小学校前,野々宮,嵐山天龍寺前,嵐山,角倉町,下嵯峨,車折神社前,有栖川,生田口,帷子ノ辻,太秦開町,太秦広隆寺前,太秦東口,蚕ノ社,太秦

天神川駅前,山ノ内,四条京阪前

13 号系統,四条堀川,西大路四条,四条御前通,四条中新道,壬生寺道,四条大宮,四条西洞院,四条烏丸,西大路松原,西大路五条,西大路花屋町,西大路七条,西大路八条,西大路駅前,吉祥院車道町,吉祥院西ノ茶屋町,吉祥院運動公園前,吉祥院長田町,久世橋東詰,久世橋西詰,築山,久世工業団地前,久世大藪町,久世殿城町,下久世,中久世

207 号系統,四条堀川,四条大宮,四条西洞院,四条烏丸,四条高倉,四条河原町,大宮五条,大宮松原,島原口,七条大宮,東寺東門前,九条大宮,九条近鉄前,九条車庫前,大石橋,四条京阪前,祇園,九条河原町,東福寺,泉涌寺道,今熊野,東山七条,馬町,五条坂,清水道,東山安井

26 号系統,四条堀川,西大路四条,四条御前通,四条中新道,壬生寺道,四条大宮,四条西洞院,四条烏丸,西大路三条,西大路御池,太子道,烏丸松原,烏丸五条,烏丸六条,烏丸七条,福王子,山越中町,山越,ユースホステル前,宇多野病院前,鳴滝本町,御室仁和寺,京福妙心寺駅前,妙心寺北門前,等持院南町,等持院道,府立体育館前,北野白梅町,大将軍,北野中学前,西ノ京円町,京都駅前

28 号系統,四条堀川,堀川松原,堀川五条,西本願寺前,七条堀川,下京区総合庁舎前,京都外大前,南広町,日新電機前,梅津段町,長福寺道,梅津西浦町,梅ノ宮神社前,松尾橋,四条葛野大路,四条中学前,西院巽町,西大路四条,四条御前通,四条中新道,壬生寺道,四条大宮,嵯峨小学校前,野々宮,嵐山天龍寺前,嵐山,松尾大社前,内田町,谷ヶ辻町,阪急嵐山駅前,嵐山公園,嵯峨釈迦堂前,小渕町,大覚寺,京都駅前

29 号系統,四条堀川,京都外大前,南広町,日新電機前,梅津段町,長福寺道,梅津西浦町,梅ノ宮神社前,松尾橋,四条葛野大路,四条中学前,西院巽町,西大路四条,四条御前通,四条中新道,壬生寺道,四条大宮,四条西洞院,四条烏丸,松尾大社前,洛西バスターミナル,新林公団住宅前,新林センター前,新林池公園,新林中通,東新林町,境谷大橋,小畑川公園北口,国道中山,国道三ノ宮,御陵町,千代原口,桂中学前,松尾大利町,苔寺道,松室北河原町

32 号系統,四条堀川,京都市役所前,河原町三条,京都外大前,南広町,四条大宮,四条西洞院,四条烏丸,四条高倉,四条河原町,西大路五条,中ノ橋五条,光華女子学園前,西京極午塚町,西京極,西京極運動公園前,市立病院前,京都リサーチパーク前,五条壬生川,大宮五条,大宮松原,川端二条,新間ノ町二条,東山二条,動物園前,岡崎道,岡崎神社前,東天王町,宮ノ前町,上宮ノ前町,法然院町,南田町,銀閣寺前,銀閣寺道,浄土寺,錦林車庫前,真如堂前,京都会館美術館前

55 号系統,四条堀川,四条大宮,四条西洞院,四条烏丸,千本今出川,千本中立売,千本出水,千本丸太町,出世稲荷前,二条駅前,千本三条・朱雀立命館前,みぶ操車場前,北野白梅町,北野天満宮前,上七軒,立命館大学前,桜木町,わら天神前,衣笠校前
8 号系統,四条堀川,京都外大前,四条葛野大路,四条中学前,西院巽町,西大路四条,四条御前通,四条中新道,壬生寺道,四条大宮,四条西洞院,四条烏丸,安井西口,猿田彦橋,黒橋,双ヶ丘,常盤御池町,宇多野御屋敷町,福王子,三宝寺,高鼻町,高雄病院前,平岡八幡前,清水町,広芝町,高雄小学校前,御経坂,御所ノ口,高雄,太秦天神川駅前
91 号系統,四条堀川,西大路四条,四条御前通,四条中新道,壬生寺道,四条大宮,四条西洞院,四条烏丸,西大路三条,西大路御池,太子道,太秦開日町,広沢御所ノ内町,嵯峨中学前,嵯峨嵐山駅前,嵯峨瀬戸川町,西ノ京円町,伯楽町,西ノ京馬代町,木辻南町,妙心寺前,花園駅前,花園扇野町,太秦映画村道,常盤・嵯峨野高校前,常磐野小学校前,太秦北路町,嵯峨釈迦堂前,小渕町,大覚寺
特 13 号系統,四条堀川,西大路四条,四条御前通,四条中新道,壬生寺道,四条大宮,四条西洞院,四条烏丸,西大路松原,西大路五条,西大路花屋町,西大路七条,西大路八条,西大路駅前,吉祥院車道町,吉祥院西ノ茶屋町,吉祥院運動公園前,吉祥院長田町,久世橋東詰,久世橋西詰,久世殿城町,下久世,中久世,国道東土川,東土川橋,久我石原町
臨 13 号系統,四条堀川,西大路四条,四条御前通,四条中新道,壬生寺道,四条大宮,四条西洞院,四条烏丸,西大路松原,西大路五条,西大路花屋町,西大路七条,西大路八条,西大路駅前,吉祥院車道町,吉祥院西ノ茶屋町,吉祥院運動公園前,吉祥院長田町,久世橋東詰,吉祥院池田町,上鳥羽馬廻,吉祥院嶋高町,石原団地前,吉祥院堂ノ後町
43 号系統,堀川五条,四条烏丸,西大路五条,市立病院前,京都リサーチパーク前,五条壬生川,大宮五条,烏丸松原,烏丸五条,西大路花屋町,西大路七条,西大路八条,西大路駅前,久世橋東詰,五条西洞院,西大路九条,船戸町,吉祥院天満宮前,吉祥院高畑町,塔南高校前,吉祥院池田町,上鳥羽馬廻,吉祥院嶋高町,石原団地前,吉祥院堂ノ後町
73 号系統,堀川五条,西大路五条,中ノ橋五条,光華女子学園前,西京極,西京極運動公園前,市立病院前,京都リサーチパーク前,五条壬生川,大宮五条,烏丸五条,烏丸六条,烏丸七条,五条西洞院,洛西バスターミナル,新林公団住宅前,新林センター前,新林池公園,国道中山,国道三ノ宮,御陵町,千代原口,国道沓掛口,国道塚原,平和台町,上桂前田町,上桂御正町,西大橋西詰,京都駅前
75 号系統,堀川五条,西本願寺前,七条堀川,下京区総合庁舎前,西大路四条,西大路

松原,西大路五条,西大路三条,西大路御池,安井西口,葛野大路御池,市立病院前,京都リサーチパーク前,五条壬生川,大宮五条,黒橋,双ヶ丘,山越中町,山越東町,太秦天神川駅前,太秦映画村道,常盤・嵯峨野高校前,常磐野小学校前,太秦北路町,西小路御池,山ノ内御池,京都駅前

80 号系統,堀川五条,京都外大前,南広町,四条葛野大路,四条河原町,西大路五条,中ノ橋五条,光華女子学園前,葛野大路御池,庚申前,西京極午塚町,西京極,西京極運動公園前,市立病院前,京都リサーチパーク前,五条壬生川,大宮五条,河原町松原,河原町五条,烏丸五条,太秦天神川駅前,四条京阪前,祇園,五条西洞院,五条坂,清水道,東山安井,五条高倉,五条京阪前

208号系統,七条堀川,下京区総合庁舎前,七条大宮,九条大宮,九条近鉄前,九条車庫前,大石橋,七条河原町,烏丸七条,西大路七条,西大路八条,西大路駅前,西大路九条,東寺南門前,羅城門,九条七本松,洛陽工業高校前,九条河原町,東福寺,泉涌寺道,今熊野,東山七条,梅小路公園前,七条千本,七条御前通,博物館三十三間堂前,七条京阪前,京都駅前

33 号系統,七条堀川,下京区総合庁舎前,七条大宮,西大路七条,梅小路公園前,七条千本,七条御前通,洛西バスターミナル,新林公団住宅前,境谷大橋,小畑川公園北口,国道中山,西境谷町三丁目,境谷センター前,洛西高校前,三ノ宮,公会堂前,樫原,川島粟田町,川島町,桂駅東口,桂消防署前,下桂,桂離宮前,桂大橋,桂小橋,葛野大路八条,西京極小学校前,大門町,月読橋,京都駅前

特 33 号系統,七条堀川,下京区総合庁舎前,七条大宮,西大路七条,梅小路公園前,七条千本,七条御前通,洛西バスターミナル,新林公団住宅前,境谷大橋,小畑川公園北口,国道中山,西境谷町三丁目,境谷センター前,洛西高校前,三ノ宮,公会堂前,樫原,川島粟田町,川島町,桂駅東口,桂消防署前,下桂,桂離宮前,桂大橋,大門町,月読橋,東側町,川勝寺,京都駅前

16 号系統,下京区総合庁舎前,九条車庫前,京都駅八条口アバンティ前,九条駅前,塩小路高倉,烏丸札ノ辻,十条駅前,十条新町,十条油小路,十条大宮,市民防災センター前,南区総合庁舎前,東寺南門前,羅城門,九条七本松,洛陽工業高校前,西寺前,八条中学前,御土居,東寺西門前,六孫王神社前,八条大宮,八条油小路,京都駅前,京都駅八条口

19 号系統,下京区総合庁舎前,九条大宮,九条近鉄前,九条車庫前,大石橋,京都駅八条口アバンティ前,市民防災センター前,南区総合庁舎前,東寺南門前,国道赤池,横大路車庫前,下三栖,中書島,京橋,西大手筋,三栖大黒町,三栖公園前,国道大手筋,国

道下鳥羽,城南宮,鳥羽大橋北詰,下鳥羽城ノ越町,上鳥羽,京都駅前,京都駅八条口

42 号系統,下京区総合庁舎前,東寺東門前,九条大宮,久世橋東詰,久世橋西詰,中久世,吉祥院天満宮前,吉祥院高畑町,塔南高校前,吉祥院池田町,吉祥院堂ノ後町,市民防災センター前,南区総合庁舎前,東寺南門前,東寺道,千本十条,落合町,中久世一丁目,久世七本松,JR 桂川駅前,高田町,洛西口駅前,京都駅前

78 号系統,下京区総合庁舎前,九条大宮,九条近鉄前,九条車庫前,大石橋,京都駅八条口アバンティ前,吉祥院車道町,吉祥院西ノ茶屋町,吉祥院運動公園前,吉祥院長田町,久世橋東詰,久世橋西詰,築山,久世工業団地前,久世大藪町,久世殿城町,下久世,中久世,西大路九条,東寺南門前,羅城門,九条七本松,洛陽工業高校前,京都駅前,京都駅八条口

65 号系統,河原町丸太町,四条烏丸,四条高倉,百万遍,飛鳥井町,叡電元田中,田中大久保町,高原町,伊織町,上終町京都造形芸大前,岩倉操車場前,国際会館駅前,岩倉大鷺町,上高野,花園橋,宝ヶ池,修学院離宮道,修学院道,一乗寺清水町,一乗寺下り松町,一乗寺木ノ本町,京大正門前,近衛通,熊野神社前,丸太町京阪前,裁判所前,烏丸丸太町,烏丸二条,烏丸御池,烏丸三条

100 円循環,河原町三条,四条烏丸,四条高倉,四条河原町,烏丸御池,烏丸三条,堺町御池,京都市役所前

5 号系統,河原町三条,三条京阪前,四条烏丸,四条高倉,四条河原町,上終町京都造形芸大前,北白川別当町,京都会館美術館前,動物園前,東天王町,銀閣寺道,浄土寺,錦林車庫前,真如堂前,岩倉操車場前,国際会館駅前,岩倉大鷺町,上高野,花園橋,宝ヶ池,修学院離宮道,修学院道,一乗寺清水町,一乗寺下り松町,一乗寺木ノ本町,北白川校前,南禅寺・永観堂道,法勝寺町,神宮道,東山三条,烏丸松原,烏丸五条,烏丸六条,烏丸七条,京都駅前

27 号系統,京都外大前,四条葛野大路,四条中学前,西院巽町,西大路四条,西大路松原,西大路五条,中ノ橋五条,光華女子学園前,葛野大路高辻,西大路三条,西大路御池,太子道,西ノ京塚本町,西ノ京藤ノ木町,馬塚町,右京ふれあい文化会館前,安井西口,葛野大路御池,庚申前,太秦天神川駅前

69 号系統,京都外大前,南広町,日新電機前,梅津段町,四条葛野大路,四条中学前,西院巽町,西大路四条,四条御前通,四条中新道,壬生寺道,四条大宮,二条駅前,千本三条・朱雀立命館前,みぶ操車場前,千代原口,桂中学前,松尾大利町,樫原,桂駅東口,桂消防署前,下桂,平田町,物集女,二条駅西口,梅津石灘町,上野橋,上桂西居町,上桂駅前,月見ヶ丘,川島六ノ坪町,自衛隊前,莚田町,桂高校前,桂滝川町,下津林大般若町,

下津林六反田,史跡公園前
71 号系統,京都外大前,南広町,日新電機前,梅津段町,長福寺道,梅津西浦町,梅ノ宮神社前,松尾橋,四条葛野大路,四条中学前,西院巽町,西大路四条,四条御前通,四条中新道,壬生寺道,四条大宮,大宮五条,大宮松原,島原口,七条大宮,東寺東門前,九条大宮,九条近鉄前,九条車庫前,大石橋,京都駅八条口アバンティ前,九条駅前
70 号系統,梅津段町,太秦東口,蚕ノ社,太秦天神川駅前,JR 桂川駅前,桂駅東口,桂消防署前,下桂,梅津石灘町,上野橋,桂高校前,下津林大般若町,下津林六反田,北上久世,下津林中島,牛ヶ瀬,中桂,桂徳大寺,上桂東ノ口,南太秦,太秦小学校前
84 号系統,四条葛野大路,光華女子学園前,葛野大路高辻,葛野大路御池,庚申前,大石橋,京都駅八条口アバンティ前,九条駅前,太秦天神川駅前,吉祥院西ノ茶屋町,吉祥院運動公園前,吉祥院長田町,久世橋東詰,吉祥院池田町,吉祥院堂ノ後町,十条駅前,九条河原町,上鳥羽村山町,吉祥院蒔絵町,上鳥羽,十条竹田街道,葛野大路八条,西京極小学校前,南松ノ木町,河原町十条,久世橋通新町,久世橋通油小路,近鉄上鳥羽口駅前,久世橋通大宮,阪急西京極,吉祥院宮ノ西町,吉祥院壱ノ段町,葛野大路九条
快速 202 号系統,西大路四条,西大路五条,西大路三条,西大路御池,九条近鉄前,九条車庫前,北野白梅町,立命館大学前,衣笠校前,西ノ京円町,小松原児童公園前,西大路七条,西大路駅前,羅城門
快速 205 号系統,西大路四条,西大路五条,西大路三条,西大路御池,七条大宮,九条車庫前,烏丸七条,北野白梅町,立命館大学前,衣笠校前,西ノ京円町,小松原児童公園前,西大路七条,七条千本,京都駅前
18 号系統,四条大宮,大宮五条,大宮松原,島原口,七条大宮,東寺東門前,九条大宮,みぶ操車場前,久我石原町,東寺南門前,羅城門,唐戸町,千本十条,五丁橋,上ノ町,上鳥羽村山町,上鳥羽小学校前,城ケ前町,岩ノ本町,地蔵前,奈須野,小枝橋,城南宮道,赤池,上鳥羽塔ノ森,久我,菱妻神社前,国道赤池,パルスプラザ前,城南宮東口,竹田内畑町,竹田駅西口
特18 号系統,四条大宮,大宮五条,大宮松原,島原口,七条大宮,東寺東門前,九条大宮,みぶ操車場前,久世橋東詰,吉祥院池田町,上鳥羽馬廻,吉祥院嶋高町,石原団地前,吉祥院堂ノ後町,東寺南門前,羅城門,唐戸町,千本十条,五丁橋,上ノ町,上鳥羽村山町,吉祥院蒔絵町
31 号系統,四条烏丸,四条高倉,四条河原町,百万遍,飛鳥井町,叡電元田中,田中大久保町,東山二条,岩倉操車場前,国際会館駅前,岩倉大鷺町,上高野,花園橋,宝ヶ池,修

学院離宮道,修学院道,一乗寺清水町,東山三条,京大正門前,近衛通,熊野神社前,烏丸三条,四条京阪前,祇園,知恩院前,東山仁王門,高野,一乗寺青城町,一乗寺北大丸町,一乗寺高槻町,一乗寺赤ノ宮町

特 8 号系統,安井西口,黒橋,双ヶ丘,常盤御池町,宇多野御屋敷町,福王子,三宝寺,高鼻町,山越中町,太秦天神川駅前,やまごえ温水プール前

特 93 号系統,安井西口,黒橋,双ヶ丘,太秦開日町,広沢御所ノ内町,嵯峨中学前,嵯峨嵐山駅前,嵯峨瀬戸川町,嵯峨小学校前,野々宮,嵐山天龍寺前,嵐山,太秦天神川駅前,太秦映画村道,常盤・嵯峨野高校前,常磐野小学校前,太秦北路町

100 号系統,動物園前,岡崎道,東天王町,宮ノ前町,銀閣寺前,銀閣寺道,錦林車庫前,神宮道,東山三条,祇園,東山七条,五条坂,清水道,博物館三十三間堂前,七条京阪前,京都会館美術館前,京都駅前

81 号系統,大石橋,京都駅八条口アバンティ前,塩小路高倉,横大路車庫前,下三栖,中書島,京橋,西大手筋,京阪中書島・伏見港公園,肥後町,西板橋,西丹波橋,住吉,棒鼻,西墨染通,竹田城南宮道,七瀬川町,竹田出橋,竹田久保町,深草下川原町,勧進橋,十条竹田街道,札ノ辻,京都駅前

特 81 号系統,大石橋,京都駅八条口アバンティ前,塩小路高倉,横大路車庫前,下三栖,中書島,京橋,西大手筋,肥後町,西板橋,西丹波橋,住吉,棒鼻,西墨染通,竹田城南宮道,七瀬川町,竹田出橋,竹田久保町,深草下川原町,勧進橋,十条竹田街道,札ノ辻,竹田駅東口,京都駅前

南 5 号系統,塩小路高倉,竹田出橋,青少年科学センター前,竹田駅東口,深草西浦町,龍谷大学前,警察学校前,稲荷大社前,十条相深町,月輪,東福寺道,塩小路橋,京都駅前

特南 1 号系統,久世工業団地前,久世大藪町,久世殿城町,下久世,中久世,中久世一丁目,久世七本松,JR 桂川駅前,高田町,桂駅東口,桂消防署前,下桂,川島六ノ坪町,自衛隊前,莚田町,桂高校前,桂滝川町,下津林大般若町,下津林六反田

南 1 号系統,久世殿城町,下久世,中久世,国道東土川,東土川橋,久我石原町,赤池,上鳥羽塔ノ森,久我,菱妻神社前,国道赤池,パルスプラザ前,城南宮東口,竹田内畑町,竹田駅西口,中久世一丁目,久世七本松,桂駅東口,桂消防署前,下桂,桂高校前,桂滝川町,下津林大般若町,下津林六反田,北上久世,下津林中島,牛ヶ瀬

22 号系統,久我石原町,久我,菱妻神社前,横大路車庫前,下三栖,中書島,京橋,西大手筋,三栖大黒町,三栖公園前,国道大手筋,国道下鳥羽,京阪中書島・伏見港公園,八丁畷,府道横大路,菱川,羽束師志水町,下鳥羽,三町,横大路,久我の社,下久我,神川小学

校前,神川出張所前,南工業団地前
南 2 号系統,赤池,上鳥羽塔ノ森,久我,国道赤池,パルスプラザ前,城南宮東口,竹田内畑町,竹田駅西口,免許試験場前,樋爪口,菱川,久我の社,下久我,神川小学校前,神川出張所前,工業団地前,馬場,芝本,JR 長岡京東口
特南 2 号系統,赤池,上鳥羽塔ノ森,久我,国道赤池,パルスプラザ前,城南宮東口,竹田内畑町,竹田駅西口,樋爪口,菱川,久我の社,下久我,神川小学校前,神川出張所前,工業団地前,馬場,芝本,JR 長岡京東口
南 3 号系統,パルスプラザ前,城南宮東口,竹田駅西口,横大路車庫前,下三栖,中書島,京橋,西大手筋,三栖大黒町,三栖公園前,国道大手筋,国道下鳥羽,下鳥羽城ノ越町,京阪中書島・伏見港公園,竹田浄菩堤院町,油小路丹波橋,伏見警察署前
20 号系統,横大路車庫前,下三栖,中書島,京橋,西大手筋,三栖大黒町,三栖公園前,国道大手筋,京阪中書島・伏見港公園,八丁畷,府道横大路,洛水高校前,南横大路,富ノ森,納所岸ノ下,納所北城堀,納所町,淀,宮前橋西詰,水垂町,樋爪町,上樋爪,免許試験場前,樋爪口,菱川,羽束師志水町
南 8 号系統,横大路車庫前,中書島,京橋,西大手筋,三栖大黒町,三栖公園前,国道大手筋,京阪中書島・伏見港公園,竹田出橋,桃陵団地前,御香宮前,板橋,丹波橋,桃山中学前,最上町,墨染,藤森神社前,直違橋一丁目,藤ノ森,青少年科学センター前,竹田駅東口,深草北蓮池町,下町,伏見インクライン前
臨南 5 号系統,竹田出橋,藤森神社前,直違橋一丁目,藤ノ森,青少年科学センター前,竹田駅東口,聖母女学院前,坊町,僧坊町,西久宝寺町,西寺町,東寺町,JR 藤森駅前,教育大学前,藤森神社,深草西浦町
西 4 号系統,JR 桂川駅前,高田町,洛西口駅前,洛西バスターミナル,新林公団住宅前,境谷大橋,境谷センター前,洛西高校前,竹の里小学校前,東竹の里町,西竹の里町,北福西町一丁目,第二回生病院前,物集女
特西 3 号系統,洛西バスターミナル,新林公団住宅前,新林センター前,新林池公園,新林中通,東新林町,境谷センター前,洛西高校前,東竹の里町,西竹の里町,京都明徳高校前,桂駅西口,三ノ宮街道,樫原水築町,樫原小学校前,野田町,福西竹の里,南福西町三丁目,南福西町二丁目,南福西町,北福西町
臨西 2 号系統,洛西バスターミナル,新林公団住宅前,境谷大橋,境谷センター前,洛西高校前,竹の里小学校前,東竹の里町,西竹の里町,大原野小学校前,南春日町,北福西町一丁目,京都明徳高校前,桂駅西口,三ノ宮街道,樫原水築町,樫原小学校前,野田町

西 1 号系統,洛西バスターミナル,新林公団住宅前,新林センター前,新林池公園,新林中通,東新林町,境谷大橋,北福西町一丁目,京都明徳高校前,桂駅西口,三ノ宮街道,樫原水築町,樫原小学校前,野田町

西 2 号系統,洛西バスターミナル,新林公団住宅前,境谷大橋,境谷センター前,洛西高校前,竹の里小学校前,東竹の里町,西竹の里町,北福西町一丁目,京都明徳高校前,桂駅西口,三ノ宮街道,樫原水築町,樫原小学校前,野田町,福西竹の里,南福西町三丁目,大原野上里北ノ町,勝山町,紅葉町,右京の里

西 3 号系統,洛西バスターミナル,境谷大橋,京都明徳高校前,桂駅西口,三ノ宮街道,樫原水築町,樫原小学校前,野田町,福西竹の里,南福西町三丁目,福西遺跡公園前,洛西大橋,中央公園南口,南福西町二丁目,南福西町,北福西町

西 8 号系統,洛西バスターミナル,新林公団住宅前,新林センター前,新林池公園,新林中通,東新林町,境谷大橋,京都明徳高校前,桂駅西口,三ノ宮街道,樫原水築町,樫原小学校前,野田町,福西竹の里,南福西町三丁目,福西遺跡公園前,洛西大橋,中央公園南口,南福西町二丁目,南福西町,北福西町

西 5 号系統,新林公団住宅前,新林センター前,新林池公園,境谷大橋,国道沓掛口,北福西町一丁目,京都明徳高校前,桂駅西口,三ノ宮街道,樫原水築町,樫原小学校前,野田町,桂坂中央,桂坂口,峰ヶ堂町一丁目,大枝山町東,花の舞公園前,桂坂小学校前,大枝山町西,ふれあいの里,西桂坂,天蓋公園前,桂坂センター前,峰ヶ堂町三丁目

西 6 号系統,桂イノベーションパーク前,桂駅西口,三ノ宮街道,樫原水築町,樫原小学校前,野田町,桂坂中央,桂坂小学校前,大枝山町西,ふれあいの里,西桂坂,天蓋公園前,桂坂センター前,峰ヶ堂町三丁目,東桂坂,桂御陵坂,京大桂キャンパス前

A.2 プログラミング学習支援システムのソースコード

ソースコード 1：フィードバックを生成するクラス

```
class FAgent:
    def __init__(self, client):
        self.client = client

    def add_message(self, role, content):
        session['conversation_history'].append({"role": role, "content": content})
```

```
        session.modified = True  # セッションの変更を明示的に通知

    def clear_history(self):
        session['conversation_history'] = []

    def get_feedback(self, question, code):
        # メッセージ履歴に基づいてリクエストを生成
        messages = [
            {
                "role": "system",
                "content": (
                    "あなたはプログラミング教育者です．\n"
                    "学習の対象者は，プログラミング初学者とします．以下にプログラミング初学者の定義を 3 つ記述します．\n"
                    "定義 1：プログラミングの基本的な概念（変数，条件分岐，ループなど）を学習中．n"
                    "定義 2：プログラミングのエラーメッセージやコードの意味を理解することが容易でない．\n"
                    "定義 3：対象者の例：学校や大学でプログラミングを学び始めた人\n"
                    "\n"
                    "以下の問題文とコードに対して問題点を指摘しつつ，修正方法を学習者が理解できるように説明してください．\n"
                    "ただし，次の条件を必ず守ってください．特に条件５～７は絶対に守ってください．\n"
                    "条件１：日本語で返答する．\n"
                    "条件２:献身的な表現で返答し,最後に「質問はありますか？」や「頑張ってください」,「又質問してください」等コメントする．\n"
                    "条件３：改行等を活用して，見やすい形式で回答する．\n"
                    "条件４：修正点がある場合は，何行目のコードに問題があるかを明記する．「例１：for i in a ・・・ 21 行目」「例２：for i in a ・・・ 21～25 行目」"
```

"条件５：修正点がない場合はないことを最初に伝え，追加で冗長なコードの整理案を提示する．"

"条件６：学習者が考える余地を残すため，コピーアンドペーストするだけで正解となってしまうような正解コードは，提示しない．\n"

"条件７:初学者は関数名や使い方を知らない可能性が高いので，range や None，with open 等の関数名を提示しても良い．その際，問題文に関連のある使い方を提示してはいけない．\n"

"以下に提示して良いフィードバック，悪いフィードバックの例を示す．"

"提示してはいけない例１：修正したコードの流れは以下のようになります．「(修正案のコード)」\n"

"提示してはいけない例２：「for i in a」を修正する必要がある場合に「for i in range(a)」と変更することで修正可能．具体的には，`range(a)`を試してみてください．\n"

"提示してはいけない例３：コードの中では，`top1`と`top2`が初期化されていないため，初めて`Node`を作成する際にエラーが発生するでしょう．これらの変数を最初に`None`で初期化する必要があります．以下のように修正してみてください：...python top1 = None　top2 = None\n"

"提示すべき例：「for i in a」を修正する必要がある場合に for 文で整数を直接使用することは出来ないため，range 関数を使うと良いです．"

"理想的なフィードバックの例は以下のようになります．\n"

"修正点：\n"

"学習者のコードにはいくつか修正が必要な部分があります．それぞれについて，問題点と改善方法を解説します．\n"

"1. for i in a の部分の問題\n"

"問題点：\n"

"a = 10 は整数型であり，for i in a のように直接ループで使用することはできません．リストや範囲を明確に指定する必要があります．\n"

"改善の考え方：\n"

"もし a が回数を意味しているなら，数値を基に範囲を指定することでループが実現できます．これには，数値から反復可能な範囲を作る仕組みを利用します．範囲とは，「開始点から終了点までの数値」を生成するもので

す．\n"

"2. top1 と top2 の初期化がされていない問題\n"

"問題点：\n"

"top1 や top2 は最初に使用する前に「初期値」を持たせておく必要があります．しかし，コード内では初期化されていないため，エラーが発生します．\n"

"改善の考え方：\n"

"top1 や top2 は「連結リストの先頭」を示す変数です．連結リストがまだ空の場合，先頭を示す変数には何も存在しないことを意味する値を代入します．この「何も存在しないこと」を示す値を特別に設定することで，連結リストを初期化できます．\n"

"3. 条件分岐の使い方についての問題\n"

"問題点：\n"

"if i < 5 の条件が適切に動作するか確認が必要です．i の値がどのように決定されるかを明確にし，条件を正しく書く必要があります．\n"

"改善の考え方：\n"

"i の値はループの中で生成されます．その値が 5 未満かどうかを判定し，正しく top1 と top2 に追加するよう条件を設計することが重要です．\n"

"4. リストを使った値の分類の不必要性\n"

"問題点：\n"

"L1 や L2 を作成していますが，これらを使用せずに直接 Node オブジェクトを構築しています．そのため，これらのリストは不要です．\n"

"改善の考え方：\n"

"L1 や L2 を削除し，直接連結リストの構築に集中するとコードが簡潔になります．\n"

"全体的な整理案\n"

"修正点を整理すると，以下のように改善できます：\n"

"for i in a の使い方を明確にし，範囲を生成する仕組みを利用します．\n"

"top1 と top2 を最初に空の連結リストとして適切に初期化し

```
ます．\n"
                    "条件分岐を，ループ内での処理の流れに従って正しく設定します．\n"
                    "不要なリスト（L1, L2）を削除し，簡潔なコードにします．\n"
                    "アドバイス\n"
                    "これらのポイントを基にコードを調整してみてください．また，理解を深めるためには，それぞれの改善点がどのような結果をもたらすかを確認しながら進めることが大切です．\n"
                    "他にも質問があれば気軽に教えてくださいね！\n"
                    "\n"
                    "以下が学習者が入力したコードに関する説明です．\n"
                    "説明：コードにはメインコード，追加ファイルがある場合のコード，学習者からの質問が含まれています．それぞれ以下の形式で前後を囲まれています．\n"
                    "メインコード：特になし\n"
                    "追加ファイルのコード：「///追加ファイル名」，「///」\n"
                    "学習者からの質問：「///質問」，「///」\n"
                    "\n"
                    "以下が問題文，コードです．\n"
                )
            },
            {"role": "user", "content": question},
            {"role": "user", "content": code}
        ]
        try:
            response = self.client.chat.completions.create(
                model="gpt-4o-mini",
                messages=messages
            )
            chatgpt_res = response.choices[0].message.content
            return chatgpt_res
        except Exception as e:
```

```
            logging.error(f"ChatGPT エラー: {e}")
            raise e

    def re_get_feedback(self, question, code, response):
        # ChatGPT に修正コードを含めず，ヒントのみの応答を再リクエスト
        messages=[
            {
                "role": "system",
                "content": (
                    "あなたはプログラミング教育者です．\n"
                    "学習の対象者は，プログラミング初学者とします．以下にプログラミング初学者の定義を 3 つ記述します．\n"
                    "定義 1：プログラミングの基本的な概念（変数，条件分岐，ループなど）を学習中．n"
                    "定義 2：プログラミングのエラーメッセージやコードの意味を理解することが容易でない．\n"
                    "定義 3：対象者の例：学校や大学でプログラミングを学び始めた人\n"
                    "\n"
                    "ただし，次の条件を必ず守ってください．特に条件５～７は絶対に守ってください．\n"
                    "条件１：日本語で返答する．\n"
                    "条件２:献身的な表現で返答し,最後に「質問はありますか？」や「頑張ってください」,「又質問してください」等コメントする．\n"
                    "条件３：改行等を活用して，見やすい形式で回答する．\n"
                    "条件４：修正点がある場合は，何行目のコードに問題があるかを明記する．「例１：for i in a ・・・ 21 行目」「例２：for i in a ・・・ 21～25 行目」"
                    "条件５：修正点がない場合はないことを最初に伝え，追加で冗長なコードの整理案を提示する．"
                    "条件６：学習者が考える余地を残すため，コピーアンドペーストするだけで正解となってしまうような正解コードは，提示しない．\n"
```

"条件 7 :初学者は関数名や使い方を知らない可能性が高いので, range や None，with open 等の関数名を提示しても良い．その際，問題文に関連のある使い方を提示してはいけない．\n"

"以下に提示して良いフィードバック，悪いフィードバックの例を示す．"

"提示してはいけない例１：修正したコードの流れは以下のようになります．「(修正案のコード)」\n"

"提示してはいけない例２：「for i in a」を修正する必要がある場合に「for i in range(a)」と変更することで修正可能．\n"

"提示してはいけない例３：コードの中では，`top1`と`top2`が初期化されていないため，初めて`Node`を作成する際にエラーが発生するでしょう．これらの変数を最初に`None`で初期化する必要があります．以下のように修正してみてください：...python top1 = None　top2 = None\n"

"提示すべき例：「for i in a」を修正する必要がある場合に for 文で整数を直接使用することは出来ないため，range 関数を使うと良いです．"

"理想的なフィードバックの例は以下のようになります．\n"

"修正点：\n"

"学習者のコードにはいくつか修正が必要な部分があります．それぞれについて，問題点と改善方法を解説します．\n"

"1. for i in a の部分の問題\n"

"問題点：\n"

"a = 10 は整数型であり，for i in a のように直接ループで使用することはできません．リストや範囲を明確に指定する必要があります．\n"

"改善の考え方：\n"

"もし a が回数を意味しているなら，数値を基に範囲を指定することでループが実現できます．これには，数値から反復可能な範囲を作る仕組みを利用します．範囲とは，「開始点から終了点までの数値」を生成するものです．\n"

"2. top1 と top2 の初期化がされていない問題\n"

"問題点：\n"

"top1 や top2 は最初に使用する前に「初期値」を持たせておく必要があります．しかし，コード内では初期化されていないため，エラーが発生

します．\n"

"改善の考え方：\n"

"top1 や top2 は「連結リストの先頭」を示す変数です．連結リストがまだ空の場合，先頭を示す変数には何も存在しないことを意味する値を代入します．この「何も存在しないこと」を示す値を特別に設定することで，連結リストを初期化できます．\n"

"3. 条件分岐の使い方についての問題\n"

"問題点：\n"

"if i < 5 の条件が適切に動作するか確認が必要です．i の値がどのように決定されるかを明確にし，条件を正しく書く必要があります．\n"

"改善の考え方：\n"

"i の値はループの中で生成されます．その値が 5 未満かどうかを判定し，正しく top1 と top2 に追加するよう条件を設計することが重要です．\n"

"4. リストを使った値の分類の不必要性\n"

"問題点：\n"

"L1 や L2 を作成していますが，これらを使用せずに直接 Node オブジェクトを構築しています．そのため，これらのリストは不要です．\n"

"改善の考え方：\n"

"L1 や L2 を削除し，直接連結リストの構築に集中するとコードが簡潔になります．\n"

"全体的な整理案\n"

"修正点を整理すると，以下のように改善できます：\n"

"for i in a の使い方を明確にし，範囲を生成する仕組みを利用します．\n"

"top1 と top2 を最初に空の連結リストとして適切に初期化します．\n"

"条件分岐を，ループ内での処理の流れに従って正しく設定します．\n"

"不要なリスト（L1, L2）を削除し，簡潔なコードにします．\n"

"アドバイス\n"

"これらのポイントを基にコードを調整してみてください．また，理解を深めるためには，それぞれの改善点がどのような結果をもたらすかを確認しながら進めることが大切です．\n"
"他にも質問があれば気軽に教えてくださいね！\n"
"\n"
"以下が学習者が入力したコードに関する説明です．\n"
"説明：コードにはメインコード，追加ファイルがある場合のコード，学習者からの質問が含まれています．それぞれ以下の形式で前後を囲まれています．\n"
"メインコード：特になし\n"
"追加ファイルのコード：「///追加ファイル名」，「///」\n"
"学習者からの質問：「///質問」，「///」\n"
"\n"
"以下が問題文，コードです．\n"
)
},
{"role": "user", "content": question},
{"role": "user", "content": code},
{
"role": "user",
"content": (
"問題文，コードに対して生成されたフィードバックには，含まれてはいけない正解コード，表現が含まれています．\n"
"上記定義や条件を満たしつつ，生成されたフィードバックを修正して再生成してください．また，「フィードバックを修正しました」等の説明はしないでください．\n"
"以下が生成されたフィードバックです．\n"
)
},
{"role": "assistant", "content": response}
]
try:

```
            response = self.client.chat.completions.create(
                model="gpt-4o-mini",
                messages=messages
            )
            re_chatgpt_res = response.choices[0].message.content
            return re_chatgpt_res
        except Exception as e:
            logging.error(f"ChatGPT エラー: {e}")
            return "再リクエスト時にエラーが発生しました．"
```

ソースコード 2：フィードバックをチェックするクラス

```
class CAgent:
    def __init__(self, client):
        self.client = client

    def check_feedback(self, feedback):
        # ChatGPT にフィードバック内のコードや関数名が含まれているか確認させる
        try:
            check_res = self.client.chat.completions.create(
                model="gpt-4",
                messages=[
                    {
                        "role": "system",
                        "content": (
                            "あなたは判定者です．日本語で返答してください．以下のフィードバックに正解コードが含まれているかどうかを判定してください．以下に正解コードの例と例外を示します．\n"
                            "例１：以下が修正案です．「(修正案のコード)」\n"
                            "例２：「for i in a」を修正する必要がある場合に「for i in range(a)」と変更することで修正可能．\n"
                            "例外：「for i in a」を修正する必要がある場合に for 文で
```

```
整数を直接使用することは出来ないため，range 関数を使うと良いです．"
                        "含まれていれば「True」，含まれていなければ「False」と返答してください．"
                    ),
                },
                    {"role": "user", "content": f"\n\n フィードバック:\n{feedback}"},
                ],
            )
            return check_res.choices[0].message.content.strip()
        except Exception as e:
            logging.error(f"ChatGPT エラー: {e}")
            raise e
```

ソースコード 3：アプリケーションのメイン処理部分

```
# メイン処理
@app.route('/search4', methods=['POST'])
def do_search() -> str:
    try:
        print("try 開始")
        # セッションに会話履歴がなければ初期化
        if 'conversation_history' not in session:
            session['conversation_history'] = []
        session.modified = True

        print("問題文成形開始")
        #  問題文を成形し、初回時のみ会話履歴(session)に追加
        if len(session['conversation_history']) == 0:
            question = request.form.get('question', '').strip()
            if not isinstance(question, str):
                question = ''.join(question)  # 必要に応じてリストを文字列に変換
```

```
        session['question'] = question
        fixed_question = question.replace('。', '。\n')
        feedback_agent.add_message("user", f"問題文:\n{fixed_question}")
    else:
        question = session['question']  # セッションから問題文を取得
        if not isinstance(question, str):
            question = ''.join(question)  # 必要に応じてリストを文字列に変
換
        fixed_question = question.replace('。', '。\n')
    print("問題文成形完了")
    #   コードを取得し会話履歴に追加
    code = request.form.get('code', '').strip()
    #code = request.form['code']
    print("コード取得成功")
    feedback_agent.add_message("user", f"コード:\n{code}")
    print("会話履歴にコード追加")
    #history = session['conversation_history']

    if not fixed_question:
        print("問題文を入力してください。")
        return redirect(url_for('entry_page'))
    if not code:
        print("コードを入力してください。")
        return redirect(url_for('entry_page'))

    print("start 1")
    # 1. ChatGPT から最初のフィードバックを取得
    chatgpt_res = feedback_agent.get_feedback(fixed_question, code)
    logging.info(f"生成されたフィードバック:{chatgpt_res}")
    print("start 2")
    # 2. フィードバックに具体的なコードが含まれているか確認
    check_result = check_agent.check_feedback(chatgpt_res)
```

```
        print(f"コード含有判定結果: {check_result}")
        logging.info(f"コード含有判定結果:{check_result}")
        print("start 3")
        # 3. コードが含まれている場合、再リクエストで修正版を取得
        if check_result == "True":
            print("具体的なコードが含まれているため、再リクエストを送信します。")
                chatgpt_res = feedback_agent.re_get_feedback(question, code, chatgpt_res)
            logging.info(f"最終的なフィードバック:{chatgpt_res}")
        else:
            logging.info("再リクエストなし")
        print("start 処理完了")

        # 会話履歴に最終的なフィードバックを保存
        feedback_agent.add_message("assistant", f"返答:\n{chatgpt_res}")
        print("会話履歴にフィードバック保存")

    except Exception as e:
        logging.error(f"エラーが発生しました: {e}")
        logging.error(traceback.format_exc())
        flash("処理中にエラーが発生しました。もう一度お試しください。")
        return redirect(url_for('entry_page'))

    # データベースに履歴を保存
    with UseDatabase(app.config['dbconfig']) as cursor:
            _SQL = "INSERT INTO conversation_log(question, code, feedback)VALUES (%s, %s, %s)"
        cursor.execute(_SQL, (question, code, chatgpt_res))
```