論文（2023年度産業理工学部プロジェクト研究成果報告）

初学者のプログラミング学習を支援するAIティーチングアシスタントの開発と評価

Development and evaluation of an AI teaching assistant to assist novice programmers in learning

勝瀬　郁代[1)]
Ikuyo Masuda-Katsuse

杉岡　慎之介[1)]
Shinnosuke Sugioka

■Abstract

In this study, we developed a web application that serves as a teaching assistant for programming courses by integrating ChatGPT. The application aims to provide dynamic and interactive responses to novice programmers in learning, accommodating the multi-layered skills required for programming. Utilizing System Prompting, Chain-of-Thought, and Few-Shot Learning technologies, the application generates sample program codes restricted to previously learned grammatical items and provides detailed explanations. The results of user questionnaires and the end-of-term test demonstrate that the application is user-friendly and enhances understanding of programming.

Key Words; プログラミング教育支援、プログラミング初学者、生成言語モデル、チャットシステム、ティーチングアシスタント

1．はじめに

1-1．プログラミング学習の困難さとICTによる支援

コンピュータプログラミング教育における初期教育の難しさはよく知られた問題である[1] [2]。プログラミングは多層のスキルを一度に求められる複雑なタスクであるため、初学者は様々な理由で躓きに直面する[1]。例えば、プログラミング学習を必須とする日本の大学の課程では、大学入学後１年目にプログラミング入門科目を履修することが多い。プログラミングで用いられる命令は英語であり、キーボードでタイピングする必要がある。そのため、英単語のスペリングやタイピングのスキルが不足していても、躓きの原因となる。

初学者はまずプログラムの文法について学ぶことになるが、データ構造や基本的なアルゴリズムとその記述方法を理解しなければならない。変数の型や配列の扱い方、条件分岐や繰り返し、関数やクラスの基本的な概念の理解に苦労する学習者は少なくない。

そして、プログラミングをするということは、具体的な問題解決を図るわけであるが、そもそも解くべき問題を理解する能力が必要である。問題自体を理解できなければ、プログラミングに限らずどのような手段であっても解くことができないのだが、この段階で躓いている場合も少なくない。

解くべき問題を理解できたら、次は問題を論理的に分析し、適切な変数やアルゴリズムを選択して、段階的に組み立てる論理的思考力が必要とされる。この段階で、初学者の多くは、実行時エラーや正しくない実行結果に直面する。どこに誤りがあってどのように解決すればよいのかが思いつかず、ここでも多くの初学者が挫折を味わう。

このように、プログラミングの初学時には多くの難関が待ち受けている。初学者のプログラミングにはこれらの躓きが個別に出現するわけでないし、初学者自身がどの段階で躓いているのかを分析することは難しい。

このような様々な難関を乗り越えるために、これまでICTを活用した様々なツールが提案されてきた。例えば、三浦[3]は、タイピングスキルの影響を軽減するために、コード補完機能を導入している。時田ら[4]は、多重ループの理解を助けるために、実行時の処理の流れを可視化することで、理解を促進しようとしている。古池ら[5]は、プログラミングの構造的理解を支援するために、処理のブロックを組み合わせるインタフェースを開発している。山本ら[6]は、初学者が体系的なデバッグ手順を身に着けるための学習支援システムを構築している。このように、初学者がプログラミング学習時に直面する個々の課題に個別に対応したツールは多く提案されてきたが、いまだ根本的な解決には至っていない。それはやはり、プログラミングは多層のスキルが一度に求められる複雑なタスクであるがゆえであろう。多層のスキルに総合的に対応する必要があると思われる。

一方で、システムエンジニア（SE）職の大卒求人案内

1）近畿大学産業理工学部情報学科

を見ると、出身学部を不問としているものが多く、プログラミング教育を受けたことのない人が多数採用されている。そして、立派にSEとして活躍しているのである。我々は以前、元SEで、新人のプログラミング教育に携わり、その後、大学にてプログラミング教育に従事している教員から話を聞く機会があった。その方によると、先輩SEがマンツーマンで指導すれば、誰でもプログラミングができるようになるとのことであった。初学者に対して、“学習者が必要としたその瞬間に、インストラクターがプログラムの概念を説明し、フィードバックを与え、詳細な説明を行うといった、動的で双方向の支援”を行うことが必要なのである[1]。それゆえ、ICTを活用した教育・学習支援システムについても、これまでのような段階別の支援ではなく、プログラミング時に必要とされる各種スキルに柔軟に対応できることが望ましい。

1-2．大規模言語モデルとプログラミング支援

近年、大規模言語モデル（LLM）を搭載したチャットアプリケーション、ChatGPT[7] の利用が急速に広まっている。LLMは大量のテキストデータによって事前学習されるが、その学習データには、webページ、Wikipedia、会話テキスト、書籍やニュース、科学データ、プログラムコードといった多様なデータソースが含まれている[8] ため、プログラムコードの生成ができるLLMは多い。また、プログラムコードの生成だけでなく、チャットを通じて、エラーの対処法やデバッグの方法について、ユーザにアドバイスを与えることもできる。つまり、LLM基盤モデルは、前節で述べた多層のスキルに対応するだけの能力を潜在的に有しており、LLMに対して“適切な問い合わせ（プロンプティング）”ができさえすれば、学習者が抱えている問題に対して、動的で双方向の対応が可能になると考えられる。

ChatGPTが広く普及するにつれ、プログラミング科目の演習課題でも、明らかにChatGPTにより生成されたとみられる解答がいくつも提出されるようになった。ChatGPTの利用で学習効果が上がるのなら、大いに利用してもらって構わないと思う。しかし、提出されたプログラムコードには、問題の意図とは異なっているものや、未学習の文法が使用されているものが散見され、必ずしも学習効果があるようには思えない。

そこで本研究では、学習者とOpenAIのChatGPTとの間に独自のアプリケーションを挿入することで、ChatGPTが有する多層のスキルへの対応能力及びチャットによる動的かつ双方向性を活かしつつ、学習効果のある利用方法を検討する。そして、産業理工学部情報学科1年後期に開講されているプログラミングⅡにおいてその効果を検証する。

2．プログラミングⅡの学習単元と授業の組み立て

情報学科では、1年前期にプログラミングⅠ、1年後期にプログラミングⅡが開講されている。プログラミングⅠでは、変数とデータ型、コレクション（リスト・ディクショナリ・タプル）、条件分岐、繰り返し、関数の導入までを学習している。プログラミングⅡの学習到達目標に含まれる文法事項は、関数の基本、例外処理の基本、モジュールの活用、クラスの基本である。ただし、グログラミングⅡでは、最初の4週を使って、前期に学習したコレクション、条件分岐、繰り返しについて復習した後、関数、例外処理、外部モジュールの利用、クラスの学習を行っている。表1に、プログラミングⅡの授業計画を示す。

表1：プログラミングⅡ授業計画

週	学習する単元
1〜4	関数を除く前期範囲の演習
5〜6	関数の基本
7	例外処理
8〜9	モジュールの活用
10〜11	Classの基本
12〜15	総合演習
16	定期試験

図1：プログラミングⅡの各回の学習の流れと本研究のアプリケーションの位置づけ

図1に、プログラミングⅡの各回の学習の流れを示す。各回の授業では、最初に、その日の単元の文法について解説する。受講生はその文法を使用したサンプルコードを実際に作成して実行することで、動作の確認を行う。このハンズオン学習により作成したプログラムコードは「授業課題」として授業時間内にGoogle Classroomに提出される。授業課題は3〜4題であることが多く、課題に取り組む間、教員とティーチングアシスタント（TA）が巡回し、必要に応じて個別対応を行う。授業課題は授業内に解答例を示す。その後、その日の学習単元の文法とこれまで学習した文法を活用する応用問題に取り組む。これを「演習課題」として2問程度出題している。授業時間内にも演習課題に取り組む時間を確保しており、教員及びTAが演習室内を

巡回して適宜質問に対応する。このように、授業時間内は教員が一方的に話す“講義”の時間を極力短くしてハンズオン形式を取り入れ、かつ、教員2名とTA2名が常に演習室内を巡回して個別対応することで、“動的で双方向の対応”を心掛けている。

演習課題については、一部の学生は授業時間内に全問提出まで至るが、授業時間内に終えられない場合は宿題となる。演習課題の提出期限は出題日から4日後の正午である。提出期限後、ただちに教員による採点が行われる。誤りが見られた場合は、何がどのように誤っているのかについてフィードバックを返す。軽微な誤りの場合はわずかな減点をして受理するが、プログラム構造に問題があるなど理解が十分ではないことが明らかな場合は「解説動画」を閲覧の上、再提出するように促す。再提出の期限は翌授業の直前である。

解説動画は出題課題ごとに作成されており、最初の提出期限後にGoogle Classroom上で公開される。「解答例のプログラムコード」ではなく「解説動画」としているのは、多くの学生が躓いていると思われる、変数、アルゴリズムの適切な選択や、それらを組み立てる手順の解説を聞きながら、学生自身がプログラムを作成することで、少しでも理解を深めてもらうためである。しかし、一方的な解説動画では、受講生それぞれの躓きに個別に対応することはできない。

そのため、授業終了後も随時、Slackやメール、訪問による質問を受け付けている。以前は頻繁に学生からの質問を受けたが、昨今はオンライン・対面に関わらず、授業時間外の質問はほとんどない。授業時間外の質問が激減している原因の一つとして、個人PCの利用が考えられる。以前は、大学の演習室のPCでしかプログラミングができなかったため、受講生は平日日中の空き時間に演習室で宿題をしていた。演習室の近くに教員の居室があるため、わからなければすぐに教員に声をかけ、解決することができた。現在は、自宅で学生自身のPCを使って課題に取り組むことが多い。Classroomへの課題提出時刻が夜間に多いことから、オンライン・対面に関わらず、教員に質問してすぐに回答が得られる時間帯に課題に取り組んでいないことが推察される。

AIなら24時間いつでも質問に対応でき、しかも、どんな基本的な質問でも遠慮なく訊くことができる。学生が自宅で夜間に演習課題の宿題に取り組んでいる時でも、AIなら動的で双方向の対応が可能となる。そこで本研究では、ChatGPTを活用したAIティーチングアシスタントシステム（AI-TA）を開発し、図1のように、授業時間外に演習課題に取り組む際に利用してもらうことにした。授業時間内はAI-TAを使用しないで教員やTAに質問するように促した。

3．開発環境とシステム構成

本研究では、AI-TAの開発環境として、Pythonウェブアプリケーションのフレームワークに Flaskを、アプリケーションのデプロイ先にAzure Web Apps[9] を使用した。

図2にアプリケーション利用時の概念図を示す。図2(a)は、OpenAIのChatGPTを直接利用する時の概念図である。利用者はチャット形式で直接ChatGPTに質問をし、ChatGPTから直接回答を得る。図2(b)は、我々が開発するAI-TAの概念図である。独自に開発するwebアプリケーションは、ChatGPTのAPI[10] を利用して、利用者とChatGPTの間を仲介する。まず、アプリケーションが利用者からの質問を受け付け、その質問に独自のプロンプトを付け加えてChatGPTに送信する。ChatGPTから得られた回答のフォーマットを変更して、webアプリケーションを通じて利用者に提示する。

図3にAI-TAの画面遷移を示す。AI-TAはユーザ管理のためのログイン機能を有する。初めて利用するユーザは、ユーザ登録ページに遷移してユーザ登録を行い、アカウントを作成する。アカウントを有するユーザは、ログインし、トップページに遷移する。トップページには各種注意事項とページリンク先が記されている。リンク先は、チャットルームとアンケートがある。利用者は、チャットルームに遷移して質問を行うことができる。また、トップページから、意見を回収できるアンケートページにも遷移できる。

（a） ChatGPT単体利用の概念図

（b）開発したアプリケーションAI-TA利用の概念図

図2：ChatGPT単体利用とAI-TA利用の概念図

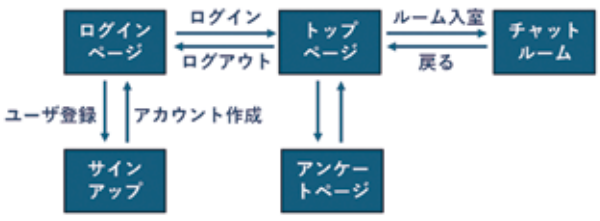

図3：AI-TAの画面遷移

チャットルームでは、利用者からの質問にシステムプロンプトが付与されてChatGPTに送信される。システムプロンプトとは、ChatGPTに役割を与えるために設定する命令文のことである。本研究では、①Pythonプログラミングを教える、②正解のコードのみを回答しないで、解答の手順を段階的に教える、③学習進度に合わせる、といったことを実現するために、次のようにシステムプロンプトを設定した。

解答の手順を段階的に提示させるためには、Chain of Thought[11] の一つであるZero-shot CoT Prompting[12] を参考に、「ステップバイステップで回答」するように指示した。また、既学習の文法事項のみを取り扱うように、既学習の単元を記載し、授業週ごとに更新した。さらに、特にChatGPTからの回答を授業で示したプログラム記述例に沿った表現で出力させたい場合は、システムプロンプトにFew-Shot Learning[13] を併用し、質問とそれに対する回答の例を与えた。本研究で作成したシステムプロンプトの例を付録に示す。

ChatGPTから取得した回答は、そのままAI-TAのチャット画面上に表示するのではなく、敢えて、プログラムのインデント（字下げ）を削除した。解説と共に解答例のプログラムコードが示されていても、学習者は、解説とコードを読んで考えながら、インデントを付与しなければならないので、一定の学習効果を期待できる。

ところで、ChatGPTのAPIライセンス契約形態の都合により、同時に複数のユーザがAPIを通じてChatGPTと対話した時、ChatGPTはユーザごとに対応するのではなく、複数ユーザを一人のユーザとして対応してしまうため、チャットは一回の質問―回答ごとに閉じ、これまでのチャット履歴を参照して回答しないように設定した。

4．実験

1年後期の専門科目であるプログラミングⅡを受講している学生のうちの希望者に対し、7週目以降からAI-TAを利用してもらった。そして、最後に、プログラミングⅡ受講生を対象にアンケート調査を行い、アンケート実施日の授業出席者63名のうち48名から回答を得た。

4-1．アンケート項目とその結果

以下に、アンケートの質問項目の一部とその回答、及びそれに対する考察を述べる。まず、アンケート回答者がAI-TAを利用したかどうかを尋ねた。

(1)「演習問題を解く際にこのアプリを使用したことがありますか」

使用したことがない	*75.0 %*
使用したことがある	*25.0 %*

利用していない36名に対して、使わなかった理由を尋ねた。

(2)「アプリを使おうとしなかった理由を教えてください（重複回答可）」

自力で解けるから	*47.2 %*
ChatGPTと大して変わらないから	*36.1 %*
友達に教えてもらえるから	*27.8 %*
アプリの使い方がわからないから	*22.2 %*
なるべく自分で解きたいと思ったから	*2.8 %*
自分で解かなければ身につかないと思ったから	*2.8 %*
ChatGPTの方が使いやすかったから	*2.8 %*

この回答結果からは、利用しなかった人の少なくとも約半数は、そもそも自力で問題を解けるので、AI-TAを必要としていなかったことになる。また、ChatGPTと大して変わらない、アプリの使い方がわからないといった回答も多かった。実際にアプリケーションを利用した人からは、ChatGPTとの違いが利点として挙げられていたので、これは残念な結果であった。AI-TAを授業に導入するにあたり、授業の中で、このアプリケーションの存在を告知しただけで、ChatGPTとの違いやAI-TAの使い方を説明する時間を取らなかったことが、利用につながらなかった原因であると思われる。

さらに、利用しなかった人に対して、実際に課題をどうやって解いていたかを尋ねた。

(3)「どうやって課題を解いていましたか（重複回答可）」

自力で解いた	*72.2 %*
友達と協力して解いた	*36.1 %*
ChatGPTを利用した解いた	*33.3 %*
解説動画を見た	*31.0 %*
自力で解けない時は解くこと自体を諦めた	*2.8 %*

自力で解いたと回答した割合が一見多く見えるが、他の選択肢と重複回答していない人の割合は30.1%であり、質問(2)の回答「自力で解けるから」の47.2%よりも下回っていた。自力で解いたと回答した人の約17%は、解けない問題にはChatGPTを利用するなど何らかの手段で解決していたことになる。また、ChatGPTを利用している人が3分の1程度いることがわかった。

これ以降は、AI-TAを利用した人を対象とした質問である。

(4)「このアプリは使いやすかったですか」

とても使いやすい	*16.7 %*
使いやすい	*75.0 %*
使いづらい	*8.3 %*
とても使いづらい	*0.0 %*

「使いづらい」と回答した人は1名のみで、ほとんどの利用者が使いやすいと回答した。以下の(5)～(8)は、「とても使いやすい」「使いやすい」と回答した人への質問である。

(5)「とても使いやすい」「使いやすい」と回答した理由を選択してください。(重複回答可。自由記述可)」

ChatGPTより詳しい解説が出力されたから	*63.6 %*
解説がわかりやすいから	*45.5 %*
解答となるプログラムが出力されたから	*45.5 %*
わからない関数などの意味も詳細に解説してくれたので、お金を払えるくらい良かったです	*自由記述*

(6)「回答の解説は分かりやすかったと思いますか」

とても分かりやすかった	*36.4 %*
少しわかりやすかった	*63.6 %*
あまりわかりやすくなかった	*0.0 %*
まったくわからなかった	*0.0 %*
解説は読まなかった	*0.0 %*

(7)「このシステムを利用することで、プログラミングの授業の理解度は向上したと思いますか」

非常にそう思う	*18.2 %*
そう思う	*63.6 %*
どちらとも言えない	*18.2 %*
そう思わない	*0.0 %*
全くそう思わない	*0.0 %*

質問(5)の回答より、ChatGPTとは異なり、コードだけはなく解説が出力される仕組みが評価されていること、質問(6)の回答より、その解説がわかりやすいと評価されたこと、質問(7)の回答より、その結果、8割を超える人が、理解度が向上したと回答していることがわかった。また、質問の仕方についても尋ねた。

(8)「チャット開始時の質問の仕方についてお伺いします」

わからないところだけを質問する文を考えて入力した	*54.5 %*
問題の中の分からない関数の意味を訊いた	*9.1 %*
問題文のコピーをそのまま入力した	*36.4 %*

「使いやすい」と答えた利用者の多くは、まずは自分自身で解けるところまで取り組み、躓いたところでAI-TAを利用していることがわかった。一方で、回答の3分の1を占めている「問題文のコピーをそのまま入力」する場合は、恐らく、AI-TAが解説と共に出力するプログラムコードを最初から期待している可能性が高い。

なお、「使いづらい」と回答した1名の利用者は、同様の質問に対して、AI-TA使用時には、課題の問題文をそのまま入力する方法で使用しており、出力された解説があまりわかりやすくなかったため、問題の解決ができず、ChatGPTの方が使いやすいと回答していた。解答のみを知りたいのであれば、ChatGPTではインデントされたコードが出力されるので、そのままコピーして提出可能である。実際に使用ログを見ると、AI-TAが表示したインデントなしのコードをそのまま再度入力して、インデント付与を求めていると思われるものがあり、解説よりも解答そのものを求めていたと思われる。

4-2. 利用者の理解度の向上

アンケートの結果から、利用者の多くは内観報告として理解度が向上したと回答していることがわかった。そこで、実際に理解度が向上したかどうかを確認するために、アンケート回答者のうち、AI-TA利用者と非利用者間で、演習課題及び定期試験の結果を比較した。

表2に、利用者、非利用者ごとの、授業の出席率、演習課題点（100点満点に換算)、定期試験（100点満点に換算）の平均を示す。まず、利用者と非利用者間で出席率にほとんど違いがないことから、授業への取り組み姿勢については大差がないことが推察される。演習課題については、3.2点の差がみられるが、演習課題は、AI-TAやChatGPTを活用する、または、友人同士で教えあうなどの様々な方法で解けるため、個々の受講生の理解度を直接測れているとは言えない。一方で、定期試験は、ノートや資料等の持ちこみは一切なしで、PCを使わず机上でプログラミングを行う筆記試験であるため、個々の受講生の理解度を測ることができる。表から、定期試験の平均点は、AI-TA利用者群が非利用者群に比べて8点近く高いことがわかる。定期試験における8点の違いはかなり大きい。このことから、アンケートの回答結果の通り、AI-TAの利用によって、受講者の理解度が向上したといえる。

表2：AI-TA利用者/非利用者の出席率・演習課題平均点(100点満点に換算)・定期試験平均点(100点満点に換算)

	出席率（%）	演習課題	定期試験
利用者	90.0	84.0	68.2
非利用者	90.8	80.8	60.3

5．今後の展開

AI-TAは、現在、Microsoft Azure Web App上に、OpenAIのChatGPT3.5 APIを利用して実装されてい

る。どちらのサービスも現時点では有料である。特にChatGPT APIは利用時に送信されるトークン数に応じて課金される。これが足枷となり、現時点では、チャットの履歴を反映できず、また、利用者のプログラムコードを送信することができなかった。また、OpenAI社はChatGPTのモデルを随時更新しており、モデルが変更されれば、同じシステムプロンプトでも振る舞いが違ってしまうことがあり、安定したサービスの提供が難しい。

一方で、昨今のLLMの開発競争の激化に伴い、高品質なLLM基盤モデルが次々と公開されるようになってきた[8]。日本国内でも日本語に強いLLM基盤モデル開発の大型プロジェクトが進行しており[14]、今年度中にLLM基盤モデルが複数公開される見込みである。これらのオープンモデルは、商用利用でなければ無料で利用できるだけでなく、下流タスクに合わせてファインチューニングすることが可能である。例えば、Instruction Tuning[15]によって、我々の求めるような出力様式になるようLLMを制御することができる。さらに、Augmented Language Model[16]によって、授業で使用した講義資料や、利用者自身が作成しているプログラムコードを踏まえた回答を出力させることができるようになる。

また、近畿大学全学で、仮想デスクトップサービスの導入が始まっており、2024年度後期からは、Microsoft Azureサービスの開始も計画されていると聞く。そうなれば、webサービスを利用した教育支援システムを無料で提供できるようになる。

このように、AIの教育利用を取り巻く環境はよい方向へと向かいつつある。今後もこういった環境を積極的に活用し、AIを活用した教育支援・学習支援を推進していきたい。

6．まとめ

本研究では、ChatGPTのチャット機能を組み込んで、プログラミング科目のティーチングアシスタントとしての機能を有するwebアプリケーションAI-TAを開発した。AI-TAは、LLMモデルが多層のプログラミングスキルに対応できる能力を潜在的に有するという利点を生かしつつ、ChatGPTの出力が初学者の理解を促進するよう、既学習の文法事項のみに限定したプログラムコードと、その詳しい解説を出力するようにプロンプティングした。そして、AI-TAをプログラミング授業で実際に使用してもらい評価を行った。利用者アンケート及び期末試験の結果から、AI-TAは初学者にとって使いやすく、プログラミング理解の向上につながることが分かった。

謝辞

本研究は、2023年度産業理工学部プロジェクトB「生成AIを用いたパーソナライズドラーニング事業」の一部として実施された。

参考文献

[1] C. S. Cheah,“Factors Contributing to the Difficulties in Teaching and Learning of Computer Programming: A Literature Review,”Contemporary Educational Technology, 12(2), ep272, pp. 1-14, 2020.
[2] E. Lahtinen, K. Ala-Mutka, and H-M. Jarvinen,“A Study of the Difficulties of Novice Programmers,”Proceedings of the 10th Annual SIGCSE, 37(3), pp. 14-18, 2005.
[3] 三浦元喜,“初学者向けProcessingプログラミング環境におけるコード補完機能の導入と効果”, 教育システム情報学会誌, 37(2), pp. 167-172, 2020.
[4] 時田真実乃, 不破泰,“初学者向けプログラミング基礎教育における可視化を用いた多重ループの効果的な学習方法”, 教育システム情報学会誌, 38(1), pp. 49-54, 2021.
[5] 古池謙人, 他,“プログラミングの構造的理解を指向した部品の段階的拡張手法の提案と支援システムの開発・評価”, 教育システム情報学会誌, 36(3), pp. 190-202, 2019.
[6] 山本頼弥, 他,“場当たり的なデバッグを行ってしまう学習者に体系的デバッグ手順を指導する授業パッケージと学習支援システムの構築”, 教育システム情報学会誌, 35(1), pp. 21-37, 2018.
[7] OpenAI,“ChatGPT,”URL（https://chat.openai.com/chat）
[8] W. X, Zhao, et al.,“A Survey of Large Language Models,”arXiv: 2303.18223v13, 2023.
[9] Microsoft,“Azure Web Apps,”URL（https://azure.microsoft.com/ja-jp）
[10] OpenAI,“API references,”URL（https://platform.openai.com/docs/api-reference/introduction）
[11] J. Wei, et al.,“Chain-of-Thought Prompting Elicits Reasoning in Large Language Models,”arXiv: 2201.11903, 2023.
[12] T. Kojima, et al.“Large Language Models are Zero-Shot Reasoners,”arXiv: 2205.11916, 2022.
[13] T. B. Brown, et al.,“Language Models are Few-Shot Learners,”arXiv: 2005.14165, 2020.
[14] 経済産業省,“Generative AI Accelerator Challenge（GENIAC）,”URL（https://www.meti.go.jp/policy/mono_info_service/geniac/index.html）
[15] J. Wei, et al.“Finetuned Language Models Are Zero-Shot Learners,”arXiv: 2109.01652, 2019.
[16] O. Ram, et al.“In-Context Retrieval-Augmented Language Models,”arXiv: 2302.00083, 2023.

付録

作成したシステムプロンプトの例である。

messages = [{“role”: “system”, “content”: ”# 命令書￥n
あなたは、[pythonプログラミングの先生] です。￥n
以下の制約条件をもとに、[授業課題の質問者の遂行] をサポートしてください。￥n# 制約条件￥n・入力内容に対し［ソクラテスメソッド］で教えてください。￥n・pythonの［リスト］、［ディクショナリ］、［タプル］、［if文］、［while文とfor文］、［関数］の知識まで利用可能です。￥n・［クラス］の知識は使わずに教えて下さい。￥n・入力内容に対し［プログラムコード］は記載しないで下さい。￥n・入力内容に対し学生が自分で考えるように［ヒント］として［例］や［説明］を提供してください。”},
{“role”: “user”, “content”: question},
{“role”: “user”, “content”: ”ステップバイステップで教えて下さい。”},
{“role”: “user”, “content”: “［答えをすべて教えず］、［プログラムコード］は［部分的に］使って下さい。”}],