研究ノート

ChatGPT を用いたプログラミング教育
――AI チュータリングの試み――

河　村　一　樹

東京国際大学論叢　人間科学・複合領域研究　第 11 号　抜刷
2026 年（令和 8 年）3 月 20 日

研究ノート

ChatGPT を用いたプログラミング教育
――AI チュータリングの試み――

河　村　一　樹

Programming Education Using ChatGPT
— Trial AI Tutoring —

KAWAMURA, Kazuki

Abstract

The author has been conducting programming education since the 2013 academic year. Initially, the classes were conducted in a lecture-plus-practice format, but issues related to copy-and-paste plagiarism emerged. As a result, from the 2014 academic year onward, lectures were discontinued and replaced with self-directed learning and individual instruction using the LMS (Moodle). Furthermore, in the 2025 academic year, a pilot study will be conducted to replace individual instruction by instructors with tutoring by generative AI (ChatGPT)

Keywords: Programming Education, Debugging, ChatGPT, AI Tutoring, AI Usage Guidelines

要　旨

筆者は，2013年度から現在まで，プログラミング教育（科目「プログラミング基礎」）を担当している。この間に，講義・実習から講義レスの個別指導へ，テキストエディタからプログラム開発環境へ，教科書ベースからLMS（Moodle）ベースへ，自作プログラミングから写経プログラミングへ，そして，教師によるチュータリングからAIによるチュータリングへと，それぞれ改変を行ってきた。この中のAIチュータリングに関しては，2025年春学期に仮実証実験を行うとともに，秋学期には本実証実験を行っている。本稿では，これらの内容について報告する。

キーワード：プログラミング教育，デバッグ，ChatGPT，AIチュータリング，AI活用ガイドライン

目　次

はじめに

筆者は，2013年度から商学部においてプログラミング教育（科目名は「プログラミング実習」から「プログラミング基礎」に改称）を担当してきた。当初は旧来の授業パターン（1コマ目講義で2コマ目実習，実習課題の提出は印刷）を踏襲したが，学生のコピペ問題が発覚した。

そこで，2014年度以降，いくつかの観点からプログラミング教育の改変を行ってきた。具体的には，授業形態については連続コマからペアコマへ，講義から自学自習と個別指導へ，教材については市販の紙媒体の教科書からMoodleにアップロードしたディジタル教科書へ，プログラミングについては自作プログラミング対写経プログラミング，デバッグについてはブラウザのデバッガーからAIチュータリングへ，実習課題の提出については印刷物からMoodleで実習課題のアップロードへ，実習支援についてはMoodleの電子掲示板であるフォーラムからAIチュータリングへ，実習環境については，テキストエディタとブラウザからIDE（統合開発環境）であるVisual Studio Codeへ，などがあげられる。

その中で，新たな問題として，Moodleのフォーラムが使われていないこと，デバッガーの限界，Zoomでの質疑応答の制約などが顕著になってきた。これらは，いずれもプログラミングにおけるデバッグ工程に起因していることが明らかになった。そこで，デバッグにおいて，AIチュータリングを試行することにした。

生成AIの技術的な発展は著しく，テキスト生成AIだけでなく，画像生成AI，音声生成AI，動画生成AI，そして，マルチモーダルAIと適用範囲が広がっている。テキスト生成AIについてはChatGPT（OpenAI），Gemini（Google），LLaMA 3（Meta）など，画像生成AIについてはDALL-E 3（OpenAI），Stable Diffusion（Stability AI）など，音声AIについてはElevenLabs（ElevenLabs）など，

動画生成AIについてはPika Labs（Pika）など，マルチモーダルAIについてはGPT-4o（OpenAI），Gemini 1.5（Google）などがそれぞれあげられる。

ChatGPTについては，2018年のGPT-1，2019年のGPT-2，2020年のGPT-3，2022年のGPT-3.5，2023年のGPT-4，2024年のGPT-4.5 & GPT-4oと進展してきている。GPT-3.5は，始めて大規模に一般ユーザに無料公開されたことで，一般層にも生成AIが浸透するきっかけになった。GPT-4は，テキストだけなく画像も対応したマルチモーダルとなり，ChatGPT Plusプランで利用可能になった。GPT-4o（オムニモデル）は，テキスト・画像・音声がすべて一つのモデルで処理できるマルチモーダル完全統合モデルとなった。

プログラミング教育では，プログラムのソースコードというテキストデータを扱うので，テキスト生成AIとの相性がよいといえる。このため，最近では大学のプログラミング教育においても生成AIを用いた事例が発表されるようになった。村田らは，GitHub Copilotを用いてプログラム作成の手順を検証し，これを前提とした教育システムを提案した[1]。佐藤らは，生成AIを活用したプログラミングを行う上で求められるスキルとして，プロンプト力がコードの品質に最も影響を与える可能性があることを示唆した[2]。佐々木らは，生成AIを活用しバーチャルTAとなるシステムの開発を行った[3]。原田らは，生成AIを用いたペアプログラミングによるプログラミング自己学習法を開発してその効果の検証を行った[4]。

以上の関連研究に対して，筆者はプログラミングのデバッグ工程に着目し，教員による個別指導ではなく，生成AI（ChatGPT3.5）によるチュータリングを試みることにした。そのために，科目「プログラミング基礎」において，2025年度春学期には（仮）実証実験を実施し，秋学期は実証実験を行う予定である。

1. プログラミング教育の変遷

ここでは，2013年度から2025年度までに至るプログラミング教育において，どのように改変を進めてきたのかについて**表1**にまとめる。

1.1　2013年度までのプログラミング教育

週2コマ連続開講の授業形態については，多くの大学で行われている授業パターンを踏襲し，1コマ目は講義･2コマ目は実習という形で進めた。教材は，教科書として自著(旧版)[5]を採用した。

実習においては，自作プログラミングとし，デバッグを終えたソースコードとプログラムの実行結果を印刷して提出させた。実習室では，教員とTA（Teaching Assistant）が机間巡回をしながら，学生からの質問に個別に答えた。実習環境としては，テキストエディタ（Tera Pad）とブラウザ（Internet Explorer）だけとした。

授業を進めていた中で，学生同士のコピペ問題が発覚した。プログラミング初学者にとって，わからないことが生じても何も対応できないことから，他の学生のプログラムをコピペして実行結果を印刷してそのまま提出していることが明らかになった。これには，そもそもプログラミングのスキル習得には学生毎に個人差があり，講義を中心とした一斉授業は適していないことが要因として考えられた。

1.2　2014年度以降のプログラミング教育

2013年度の授業状況を踏まえて，2014年度からは，講義を止めて，自学自習と個別指導に切

表1　プログラミング教育の改変経過

年度	学期	履修者数	授業形態	教材	プログラミング	デバッグ	課題提出	実習支援	実習環境	評価方法
2013	半期		週2コマ連続，講義＋実習	教科書(旧版)	自作	ブラウザ起動	実行結果をプリントアウト	なし	TeraPad，IE	課題提出数＋個別指導状況
2014	半期		週2コマ連続，**自学自習＋個別指導(対面)**	同上	同上	同上	同上	フリーの掲示板	TeraPad，IE	同上
2015	半期		同上	同上	同上	同上	**ソースコード**と**実行結果**をサイボーズにアップロード	サイボーズの掲示板	TeraPad，IE，**グループウェア(サイボーズLive)**	同上
2016	春学期	23	**ペアコマ**，個別指導(対面)	同上	同上	Chromeのデバッガー(F12)	**htmlファイル**をMoodleにアップロード	**Moodle[フォーラム]全員閲覧**	TeraPad，**Chrome**，**Moodle[課題]**	同上
2017	春学期	39	同上	**MoodlePDF**	同上	同上	同上	同上	TeraPad，Chrome，**Moodle[小テスト]**	同上
	秋学期	62	同上	同上	同上	同上	同上	同上	同上	同上
2018	春学期	19	同上	同上	同上	同上	同上	Moodle[フォーラム]**学生毎に限定**	同上	同上
	秋学期	50	同上	同上	同上	同上	同上	同上	同上	同上
2019	春学期	44	同上	同上	同上	同上	同上	同上	同上	**＋迅速性**
	秋学期	50	同上	同上	同上	同上	同上	同上	同上	課題提出数
2020	春学期	COVID-19で開講せず								
	秋学期	30+31	ペアコマ，**個別指導(Zoom)**	同上	同上	同上	同上	同上	同上	同上
2021	春学期	24	同上	同上	同上	同上	同上	同上	同上	同上
	秋学期	27	同上	同上	同上	同上	同上	同上	同上	同上
2022	春学期	13	同上	同上	同上	同上	同上	同上	**VS Code**，Chrome，Moodle[小テスト]	同上
	秋学期	24	同上	同上	同上	同上	同上	同上	同上	同上
2023	春学期	27	同上	同上	同上	同上	同上	同上	同上	同上
	秋学期	11	同上	同上	**写経**	同上	ソースコード＋実行結果＋説明文	同上	同上	課題提出数＋**確認テスト**
2024	春学期	20	同上	同上	自作	同上	htmlファイルをMoodleにアップロード	同上	同上	同上
	秋学期	18	同上	同上	写経	同上	ソースコード＋実行結果＋説明文	同上	同上	同上
2025	春学期	19	同上	**教科書(新版)**	自作	**AIチュータリング**	htmlファイルをMoodleにアップロード	**生成AI(仮実証実験)**	同上	同上
	秋学期	9	同上	同上	同上	同上	同上	**生成AI(実証実験)**	同上	同上

り替えることにした。それに合わせて，BBS（フリーソフトのYY-BOARD）やLMS（サイボーズLive，Moodle）を導入するとともに，以下のようにいくつかの改変を行ってきた。

1.2.1　授業形態と教材の改変

（1）授業形態について

2014年度からは，講義は一切せずに，自学自習と個別指導を実施した。自学自習については学生が教科書を講読しながらプログラミングの実習を繰り返し，個別指導については実習室の教卓において教員と学生1対1の対面形式で質疑応答を行うことにした。自学自習にすることで，授業中だけでなく授業外（放課後や自宅など）でもプログラミングができるようになった。個別指導では，学生がわからないことをそのままにしないように，理解できるまで丁寧に説明を行った[6]。

2016年度からは，全学レベルで授業運用が変更となり，ペアコマ（月曜日1コマ＋木曜日1コマあるいは火曜日1コマ＋金曜日1コマ）が導入された。これは，実習科目などにおいて，1週間の間をあけずに週2回授業を繰り返すことによってスキルを身につけさせるという目論見があると思われる[7]。

2020年度春学期はCOVID-19のため授業はすべて休講になったが，秋学期はZoomによるオンラインでの開講となった。その際に，Zoomのブレイクアウトルーム（学生1名のみ入室，教員が巡回）を利用することで，個別指導を実施した[8]。2021年からは，対面授業に戻ったが，個別指導は実習室でZoomを併用して実施している。これは，ブレイクアウトルームにおいて学生が自分の画面を共有することで，操作の手順や表示したJavaScriptのソースコードを見ながら教員がアドバイスできるからである。

（2）教材について

2013年度からは，上述したように自著を教科書として使用し，教科書を参照しながら講義を行っていた。

2017年度からは，出版社から承諾を得た上で，教科書の内容（章・節毎の解説，実習課題，ヒントと実行結果の画面イメージ）をすべてMoodleにアップロードした。これによって，Moodleにアクセスするだけで，自学自習ができるようにした。

2025年度からは，旧版となる自著の改訂版[9]を出版し，それを教科書に採用した。新版では，JavaScriptの言語仕様の改訂に合わせて推奨する代替（var→let/const，with文は使用不可，暗黙のグローバル変数の使用も不可など）にするとともに，第7章をゲームプログラミングに全面的に書き替えた。これにより，最終的にはJavaScriptを用いたゲームプログラムの作成を目標とするようにした。

1.2.2　プログラミングとデバッグの改変

（1）プログラミングについて

2013年度からは，自作プログラミングによる実習を進めた。自作プログラミングとは，日本語のプログラム仕様をもとに，JavaScriptを用いてプログラミングとデバッグを行うことを意味している。

2023年度と2024年度には，自作プログラミングと写経プログラミングの比較実験を行った。それぞれ春学期には自作プログラミングを，秋学期には写経プログラミングを実施し，Moodleで学習データを収集して比較した[10]。

写経プログラミングとは，プログラムのソースコードを画像ファイルに変換してMoodleにアップロードし，学生はそれを見てコード入力してデバッグを行うというものである。画像ファイルにしたのは，学生がそのままコピペできないようにするためである。また，単にプログラミングするだけではなく，入力したソースコードの意味をコメント行として挿入させて提出させるようにした。さらに，学期末にはJavaScriptの構文と基本的なロジックを問う確認テストを実施した。以上の形で実証実験を行った結果，プログラミング初学者にとっては自作プログラミングよりも写経プログラミングの方が，学習効果が高くなることが示唆された[11]。

（2）デバッグについて

2013年度からは，ブラウザであるInternet Explorerを起動することで動作確認を行った。ただし，プログラムが仕様通りの結果を画面に表示しなかった場合は，スペルミスか構文エラーかロジックエラーかを自分で確認するしか方策がなかった。テキストエディタであるTera Padを使うと全角のスペース（構文エラーとなる）を判別することができたり，JavaScriptの構文箇所にはそれぞれ色がつく。しかし，それだけではプログラミング初学者にとっては，エラーを見つけ出す

のが難しいといえる。

そこで，2016年度からは，ブラウザをGoogle Chromeに変えることにした。Google ChromeにはJavaScriptのデバッガーが同梱されており，ファンクション12を押すとデベロッパーツールが起動し，HTML/CSS/JavaScriptの確認・編集，コンソールログの確認，ネットワーク通信の監視，モバイル表示の確認，エラーの調査ができる。これにより，プログラムのエラーメッセ―ジとエラー箇所（行番号）が表示され，デバッグがしやすくなるといえる。

1.2.3　課題提出の改変

2013年度からは，プログラムの実行結果だけを印刷して提出させた。しかし，印刷だけだと名前だけを変えるだけで済むので，コピペがしやすいといえる。

そこで，2015年度からは，無料のグループウェアを使い，プログラムのソースコードと実行結果をアップロードするように変更した。グループウェアとしてはサイボーズLiveを採択し，学生毎にアカウントを用意した。学生は自分のアカウント領域内でファイルのアップロードを行うため，他の学生との連携はしにくい環境になっている。また，アップロードされたソースコードにより，教員の方でもプログラムの動作確認ができるようになった。

2016年度は，全学レベルでMoodleが導入された。そこで，scriptタグによりJavaScriptを埋め込んだhtmlファイルを，Moodleの[課題]（以降，Moodleの活動やリソース，あるいは，Zoomに関する用語については，鍵括弧で囲む）にアップロードさせるようにした。2017年度からは，提出先をMoodleの[小テスト]に変更した。これは，日本語仕様を表示する（HTMLのコメント行として，<!-- コメント行 -->）とともに，htmlファイルのテンプレートを学生に提供するためである[12]。

2023年度と2024年度の秋学期には，htmlのソースコード（説明文となるコメント行を挿入）と実行結果（画面イメージ）をMS-Wordにコピペしたdocxファイルをアップロードするようにした。これは，写経プログラミングの実証実験を行ったからである。

1.2.4　実習支援と実習環境の改変

（1）実習支援

以前から使用していたフリーの掲示板であるYY-BOARDを用いて，学生の質疑応答に対応していた[13]が，2015年度は，上述したサイボーズLiveを導入したので，サイボーズLiveの掲示板を利用するように変更した。さらに，2016年度には，Moodleが導入されたことで，Moodleの[フォーラム]に変更した。[フォーラム]では，テキスト文だけでなく，添付ファイルをアップロードすることもできるので，学生の中にはhtmlファイルを直接アップして質問することも多くなってきた。その結果，他の学生が勝手にコピーするという状況が散見された。

そこで，2018年度から，[フォーラム]の[利用制限]を設定し，学生毎に[フォーラム]を開設するように変更した（**図1**）。

図 1　Moodle の [フォーラム] における [利用制限]

	姓 / 名	メールアドレス(POTI)	ステータス	開始日時	受験完了	継続時間	評点/10.00	Q. 1 /10.00
2	受験をレビューする		終了	2025年 04月 3日 15:30	2025年 04月 7日 16:27	4 日	10.00	10.00
2	受験をレビューする		終了	2025年 04月 3日 16:32	2025年 04月 7日 16:20	3 日 23 時間	0.00	0.00
2	受験をレビューする		終了	2025年 04月 7日 10:54	2025年 04月 7日 16:18	5 時間 24 分	5.00	5.00

図２　Moodle の [小テスト] モジュールの表示

（2）実習環境

2013年度からテキストエディタとブラウザだけで実習を続けていたが，2022年度からは，統合開発環境である Visual Studio Code（以降，VS Code と略す）を利用することにした。VS Code は，マイクロソフト社が提供するコードエディタであり，オープンソースに基づき構築されているので無償で利用できる。特徴的なこととしては，クロスプラットホーム対応であること，拡張機能が豊富であること，多くの言語（HTML/CSS/JavaScript，TypeScript，JSON，拡張機能により Python, PHP, Ruby, Node.js など）をサポートしていること，シンタックスハイライト（ソースコードの構造を視覚的に区別できる表示機能）があることなどがあげられる。

さらに，VS Code では，コード補完（補完候補のコードが挿入），エラー表示（拡張機能），クイックフィックス（修正の候補が表示），デバッグアクション（プログラムの実行制御）などの機能があり，プログラミング初学者にとってプログラムのデバッグにおいて非常に有効な支援となり得る。

1.2.5　評価方法の改変

2013年度からは，課題提出数と個別指導情報（進捗度合や質問内容など）によって評価を行ってきた。ただし，Moodle を [課題] から [小テスト] に変更したことにより，実習課題毎に [開始日時] と [受験完了] および [継続時間] のデータを取得できるようになった（図2）。

そこで，2019年度には，学生が個々の実習課題に対して，どれくらいの時間をかけてプログラミングとデバッグを行ったかを，「迅速性」という評価基準として加えることにした。この結果，「正確性」（プログラムの仕様通り）だけでなく，「迅速性」を加えた方がクラス全体の課題達成率が向上することが明らかになった [14]。

2. 現状の科目「プログラミング基礎」の進め方

1で取り上げた内容をもとに，現在実施している科目「プログラミング基礎」において，どのように授業を実施しているのかについて，受講者である学生側と教授者である教員側それぞれの立場から取り上げる。

2.1　学生側（授業中）

学生は，Zoom の [ミーティングルーム] に入室後，指定された [ブレイクアウトルーム] に入る（図3）。

Moodle を立ち上げ，[小テスト] をクリックし，実習課題の日本語仕様（HTML のコメント行として挿入）を確認する。次に，HTML のソースコード（テンプレートとなる）をコピーして（図4の①），VS Code にペーストする。

VS Code において，script タグの中に，実習課題の仕様に応じた JavaScript のソースコードを入力する。ファイル名を toi〇-〇-〇.html として保存してから，[デバッグなしで実行] をクリック

図３　Zoom のブレイクアウトルーム

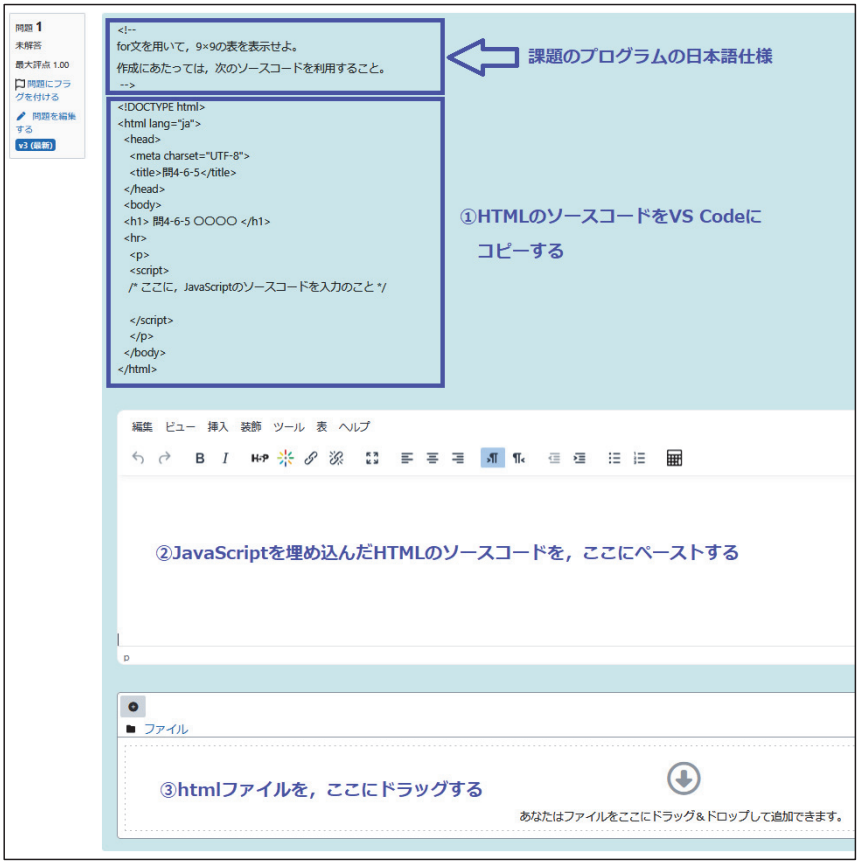

図４　実習の進め方

する。ブラウザはGoogle Chromeを選び，プログラムを実行する。エラーがあれば修正しながら，正しい実行結果の画面が得られるまで繰り返す。

この間に，教員が[ブレイクアウトルーム]に入室してきたら，わからないことなどについて質問をする。その中で，教員にソースコードを見せたい場合は，Zoomの[画面共有]ボタンを押す。これによって，より詳細に助言を受けることができる。

正しい実行結果が得られてから，Moodleの[小テスト]に戻り，出来上がったHTMLのソースコードを中段の欄にペーストする（図4の②）とともに，htmlファイルを下段の欄にドラッグする（図4の③）。[小テストを終了する]ボタンをクリック後，さらに[すべての解答を送信して終了する]ボタンをクリックすることで，Moodleに実習課題がアップロードされたことになる。

なお，学生は，提出した実習課題についてMoodleの[評定]で確認することができる。10点になっていない場合は，教員が記述したフィードバックコメントをみながらソースコードを修正して，再度アップロードし直すことになる。

2.2　教員側（授業前・授業中・授業外）

（1）授業前

授業前に，Zoomの[ミーティングルーム]を予約する必要がある。科目「プログラミング基礎」はペアコマなので，秋学期の場合，月曜日3コマ目については初回（2025年9月8日午後1時20分）から最終回（2025年12月8日午後3時）まで，木曜日3コマ目については初回（2025年9月4日午後1時20分）から最終回（2025年12月4日午後3時）まで，を[ミーティングをスケジュールする]で設定する（図5）。

図５　ミーティングルームのスケジュール

以上によって，予約された[ミーティングルーム]のURL及び[ミーティングID]とパスワードが付与されるので，これをMoodleのコース「プログラミング基礎」のトップの[トピックス]に掲載する（図6）。これによって，学生は該当日時に指定された[ミーティングルーム]に入室できる。

（2）授業中

実習室で授業開始後に，当日のZoom[ミーティングルーム]を[ホスト]として開設する。次に，出席者（遅刻者分も含める）の人数分の[ブレイクアウトルーム]を[自動で割り当てる]で作成する（図7）。これによって，学生一人だけの[ブレイクアウトルーム]になる。

図 6　Moodle のトピックス「連絡事項」の表示

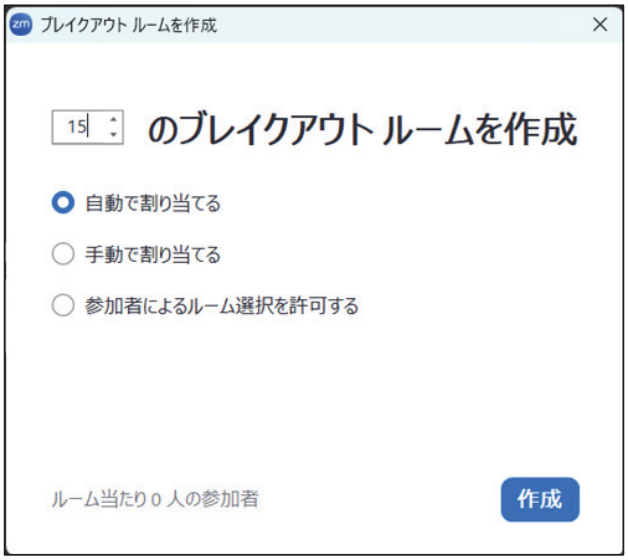

図 7　Zoom ブレイクアウトルームの作成

[自動で割り当てる]にしたのは，教員はルーム1から順番に[ブレイクアウトルーム]を巡回するので，学生にとっては毎回ランダムにルームが割り当てられるように配慮したからである。遅刻者については，[ミーティングルーム]に入室すると画面に表示されるので，教員が遅刻者用のルームを割り当てることができる。なお，一通り巡回が終わってから質問したい場合は，[ブレイクアウトルーム]で[ヘルプを求める]ボタンをクリックすることで，教員側にメッセージが届くようになっている。

[ブレイクアウトルーム]に入室した教員は，学生と1対1で個別指導（進捗の確認，質疑応答など）を行う。

（3）授業後

学生は，放課後や自宅などの授業外でもプログラミング実習ができる。その際に，わからないことが生じると実習が中断してしまうことになる。そこで，学生毎のMoodleの[フォーラム]で質疑応答を行えるようにした。[ディスカッショントピックを追加する]をクリックすることによって，質問（テキスト文だけでなく，htmlファイルの添付も可能）をアップすることができる。

一方，教員は，Moodleの[最近の活動]において，学生から[フォーラム]にアップされたディスカッショントピックが表示されるので，それによって質問に対する回答をアップすることができる。ただし，最大数日間のタイムラグが生じる場合がある。

次回の授業までに，教員はMoodleの各[小テスト]にアップロードされたhtmlファイルの動作確認を行う。プログラムの仕様通りの実行結果が得られた場合は評点を10点，軽微なエラーがある場合は5点，重度なエラーがある場合は0点，と評点を与える。エラーについては，[フィードバックコメント]にその原因と対応策のヒントを戻す。

すべての[小テスト]について評点をつけてから，Moodleの[評定][ユーザレポート]でExcelスプレッドシートにエクスポートする。それを，Moodleに公開することで，学生に進捗状況を確認させることができる。（図8）

	A	B	C	D	E
1	名	姓	小テスト:問4-1-1 (5	小テスト:問4-2-1 (5	小テスト:問4-2-2 (5
2	22	太	10	-	-
3	22		10	-	-
4	22	来	0	-	-
5	22		10	10	10
6	22	貴	10	10	10
7	22	菜	10	-	-
8	22	梨	10	-	-
9	22	みか	10	0	-
10	22	空	10	-	-
11	23	哉	10	5	0
12	23	A KRISTINA	0	-	-
13	23	悠貴	0	-	-
14	23		-	-	-
15	23	太	0	-	-
16	23		5	10	-
17	23	音	-	-	-
18	24	正	5	-	-
19	24	斗	5	-	-
20	24	雅	10	10	-

図８　実習の進捗状況

2.3　問題の発覚

2024年度まで実施してきた「プログラミング基礎」において，再びいくつかの問題が顕著化してきた。

(1) Moodle[フォーラム]の利用が一部の学生に限定

Moodleの[フォーラム]を使い，授業外における学生からの質問にいつでも対応できるような支援体制を続けてきたが，利用する学生が限られるという事態が起きている。学生へのアンケート調査によると，「授業外で自学自習することがほとんどないので必要ない，授業中に先生に聞けばよい，Moodleのアクセスが面倒くさい，人に聞くのではなく自分で解決したいから，すぐに回答を得ることができない」などの理由があげられた。

MoodleとVS Codeが使える環境であれば，どこでもいつでも自学自習ができるようにしているが，「すぐに回答を得ることができない」ことが利用しない要因になっていると考えられる。

(2) Google Chromeのデバッガーの限界

学生が悩んだり，わからなくて先に進めなくなる場面は，プログラミングよりもデバッグ工程にあるといえる。

VS Codeの操作において，インテリセンス機能（コード補完候補，シンタックスハイライトなど）を用いるように指導しているので，単純な入力ミスを防ぐことはできる。その後，Google Chromeのデバッガーを起動してプログラムをテストすると，スペルミスや構文エラーについては検出できるが，論理エラーまでは対応できない。学生が最も悩む状況は，プログラムの実行はできるのだが結果が異なったり，途中でABEND（abnormal end）した場合である。これらは，いずれも論理エラーに起因していることが多い。これらに対する助言が，Google Chromeでは得られないという問題が生じた。

(3) Zoomでの質疑応答の制約

現在，実習室における個別指導を，Zoomを用いて行っている。通常，[ブレイクアウトルーム]であれば，音声は他のルームに届くことはない。しかし，実習室内の音響設備の状況により，教卓のパソコンのマイクを使うことができない。このため，教卓の教員と席に座っている学生との間で直接会話をすることになるので，他の学生にも会話が聞こえてしまうことになる。

その結果，他の学生に聞かれたくないので「質問はありません」ということで個別指導を終えてしまうという学生が見受けられるようになった。また，中には，何もわからず質問しようがないという学生も同じ応対をするような様子が見られた。

3. AIチュータリングの試み

2.3の問題に対処するために，2025年度から，Zoom[ブレイクアウトルーム]での教員による個別指導ではなく，生成AI（ChatGPT-3.5）によるチュータリングに切り替えることにした。そのために，2025年度の春学期に（仮）実証実験を実施した上で，秋学期に本格的な実証実験を行うことにした[15]。

3.1　生成AIの使用について

(1) ChatGPTの採択理由

無償となる生成AIとしては，ChatGPT（OpenAI），Gemini（Google），Meta AI（Llama 3），Claude（Anthropic）などがある。その中で，今回はChatGPT（-3.5）採択するに至った。

その理由としては，プログラミング初学者にも丁寧な説明が可能である，デバッグの手順や考え方を自然な日本語で提示できる，日本語の質問に高精度に応答できる，エラーメッセージおよびロジックの説明やアルゴリズムの指導に向いている，などがあげられる。一方，留意点として，処理速度が遅くなる場合があったり，生成されるコードはあくまで補助であり動作検証が必須であることも考慮する必要がある。

（2）ChatGPTのアカウントについて

ChatGPTを利用するにあたり，アカウントを取得するか否かのケースがある。

①取得した場合

会話履歴がすべて残るので，過去の会話が保存され後から見返したり，続けて会話ができる。また，テキスト文だけでなくファイルをアップロードして質問したり，画像を使って質問もできる。

②取得しない場合

使用できる版は，ChatGPT-3.5のみに限られる。会話履歴は保存されず，その場限りの会話で画面を閉じると内容が消去される。また，ファイルのアップロードや画像認識，カスタム指示などの機能が使えず，アクセス回数や時間帯にも利用制限がかかる場合がある。

（3）ChatGPTの会話履歴について

会話履歴が一画面に収まらないとき，PrintScreenキーではすべてをコピーすることができない。そこで，Google Chromeの拡張機能であるGoFullPageをインストールして起動すると，全画面がコピーできてPDFファイルに変換できる。しかし，ここで画面の下部が表示されないという問題が生じた。このため，別法を試す中で，会話履歴の全画面をコピーしてMS-Wordにペーストすると整形されて表示できることが判明し，docxファイルとして保存することにした。

なお，ある時点までの会話履歴をすべて取得したい場合は，ChatGPT画面の左下にあるアカウント名をクリックし，[設定]→[データコントロール]→[エクスポートする]→[エクスポートを確認する]と続ける。すると，OpenAIからメールが届く。[データエクスポートのダウンロード]をクリックし，圧縮ファイルを解凍し，chat.htmlファイルを開くと，それまでのすべての会話履歴が表示できる。

3.2 （仮）実証実験

春学期の科目「プログラミング基礎」において，仮の実証実験を実施した。実施にあたっては，1）アカウント取得は自由，2）使い方に制限を与えず，3）会話履歴の添付は自由，4）ChatGPTの利用は評価対象外，5）最終回にアンケートを実施，とした。

1）については，アカウントを取得せずに使う学生が散見された。2）については，難しいプログラムになるほど最初からソースコードの生成を求めたり，ChatGPTが提示したソースコードをそのままコピペしてプログラムの動作確認をせずにアップロードする学生がいた。3）については，ChatGPTを使ったのにも関わらず，会話履歴のdocxファイルをアップロードしない学生もいた。4）については，実習課題に対する評点にはChatGPTの利用状況を加味せずとしたので，ChatGPTを全く使わずに自力で実習を続ける学生もいた。

5）については，**図9**のようになった。

アンケート結果によると，ChatGPTの使用は自由としたが，多くの学生が使っていることが明らかになった。これは，教員による個別指導では時間が限られたり，質問したいときにできなかったりする一方で，ChatGPTならばいつでも質問できて即回答がわかることが考えられる。ChatGPTの使い方については，最初から実習課題のソースコードを生成させている場合もあり得

(1) ChatGPTを使って質問をしたか？	
毎回	2
ほぼ毎回	7
ときどき	6
一度もない	0

(2) ChatGPTを利用した目的は？(複数回答)	
エラーメッセージの意味を調べる	10
コードの修正方法を尋ねる	9
プログラムの仕様の理解を深める	1
他の方法での実装アイデアを得る	0
その他	0

(3) ChatGPTの回答は問題解決に役立ったか？	
とても役立った	4
ある程度役立った	8
あまり役立たなかった	2
まったく役立たなかった	0

(4) 回答に対して，誤りや不正確さを感じたことは？	
頻繁にあった	1
ときどきあった	7
まれにあった	6
まったくなかった	0

(5) ChatGPTの説明は理解しやすかったか？	
非常にわかりやすかった	1
ある程度わかりやすかった	10
やや難しかった	2
まったくわからなかった	1

(6) ChatGPT利用でエラー原因を考える力がついたか？	
非常にそう思う	3
ややそう思う	8
あまりそう思わない	3
まったくそう思わない	0

(7) ChatGPTの利用に抵抗感や不安感を感じたか？	
とても感じた	1
少し感じた	6
ほとんど感じなかった	6
まったく感じなかった	1

(8) ChatGPTと教員のどちらを多く利用したか？	
ChatGPTの方が多かった	5
教員の方が多かった	0
同じくらい	8
どちらもほとんど利用しなかった	0

(9) ChatGPTのようなAIツールを活用したいか？	
ぜひ活用したい	7
どちらかといえば活用したい	5
あまり活用したくない	1
まったく活用したくない	0

(10) ChatGPT利用により，授業の学びが深まったか？	
非常にそう思う	5
ややそう思う	8
あまりそう思わない	0
まったくそう思わない	0

図９　アンケート結果

るので，何らかの制約を与える必要がある。また，ChatGPTの利用については全体的に肯定的であり，授業での利用についても賛成する学生が多いようである。

3.3　実証実験

春学期の（仮）実証実験の実施結果を経て，秋学期には同じ科目「プログラミング基礎」においてAIチュータリングの実証実験を予定している。具体的には，次のような方策を考えている。

科目ガイダンス時に，「AI活用ガイドライン」を配布する。また，実習室に毎回持参して常時参照するように指示する。「AI活用ガイドライン」には，次のような事項を記載する。

（1）アカウント取得の義務づけ

https://chatgpt.com にアクセスして，ChatGPTのアカウントを取得する。その際のIDは，大学が付与しているsxxxxxxxx@al.tiu.ac.jp（xxxxxxxxは学籍番号）とする。

(2) 実習の進め方

実習課題の日本語仕様を読んだ上で，例題を参考にしながら，自分でJavaScriptのソースコードを入力すること。最初からソースコードを生成させるような質問はしないことを実習における前提条件とする。会話履歴により発覚した場合は，減点対象にする。

デバッグ工程に入ってから，ChatGPTと会話しながらエラー原因を突き止めプログラムを完成させる。完成したかどうかは，Moodleにアップロードした章・節毎の[ヒントと実行結果]にある実行結果の画面イメージと同一か否かで判断すること。同一が確認できてから，実習課題をMoodleにアップロードする。確認せずにアップロードした場合，評点は0点とする。

(3) プロンプトの入力方法

質問のコツとしては，具体的に書く（どのような状況で，どんなエラーが起きているかを明示），コードとエラーメッセージを添付する（該当するコードと表示されたエラーや警告を一緒に提示），期待した動作と実際の動作を書く（何をしたかったのか，結果がどう違ったのかを伝える）などがあげられる。

(4) 質問のテンプレート

次のような質問のテンプレートを用意して，学生に使わせる。

①パターン1

```
【質問内容】
 このコードを実行すると，〇…〇というエラーが出ます。
【やりたいこと】
 本当は，〇…〇をしたいのです。
【コード】
 該当するコードをここにペーストする。
【エラーメッセージ】
 エラー文をここにペーストする。
この原因と解決方法を教えてください。
```

②パターン2

```
【問題の説明】
 〇〇という現象が起きています。(例：変数が undefined)
【該当コード】
 該当するコードをここにペーストする。
【知りたいこと】
 - なぜこのような現象が起きるのか
 - どういうミスが考えられるか
 - 仕様上の注意点があれば教えてください
 - 修正コードは不要で，分析と考察のみをお願いします
```

(5) ハルシネーション

ChatGPTの回答には誤情報が含まれる場合があることを説明する。このため，必ずプログラムを実行して，動作確認を義務づける。

（6）会話履歴の保存

ChatGPTとの会話履歴は，ChatGPTを終了するまでが一つのチャットとなり，それにチャット名が自動的につく。また，画面の左側には，過去のチャット名が列挙される。このため，そのチャット名をクリックすることで，再度質問を続けることもできる。

ある実習課題において，ChatGPTとの会話を終了した時点で，そのチャットすべてをコピーしてMS-Wordにペーストし，docxファイルとして保存する。

（7）実習課題の提出

Moodleの[小テスト]の中段欄にソースコードをペーストし，下段の添付欄にhtmlファイル（実習課題のプログラム）とdocxファイル（ChatGPTとの会話履歴）をアップロードすることを義務づける。

（8）評価について

実習課題におけるプログラムの正確性とChatGPTの使い方の妥当性を評価基準とする。

プログラムの正確性については，日本語仕様通りに動作するかどうかを判定する。具体的には，次のようにする。

①5点

・実行結果の画面イメージと同一である

・[フィードバックコメント]に，「OKです。」と表示する

②0点

・きちんとデバッグを終えていないため，実行しても何も表示されなかったり，実行結果の画面イメージと異なる

・ChatGPTとの会話履歴のファイル（toi〇-〇-〇.docx）がアップロードされていない

・ファイル名がtoi〇-〇-〇.htmlあるいは，toi〇-〇-〇.docxになっていない

・h1タグに，実習課題番号（問〇-〇-〇）と自分の名前，水平線が入力されていない

・[フィードバックコメント]に，「再提出」と表示する。

学生は，評点が5点になるまでデバッグを繰返しながら，再提出を続ける。

評点が5点になったものに対して，ChatGPTの使い方の妥当性を評価する。評価の観点としては，ChatGPTを主体的・効果的に活用しているかどうか，単に答えを得るのではなく自らの考えを持ち対話を通じて理解を深めているか，などがあげられる。具体的には，次のようにする。

①5点

・問題の原因やエラーの内容を自分なりに分析し，その上でChatGPTに相談している

・回答に対してさらに質問や試行錯誤を行い，複数回のやりとりで理解を深めている

・ChatGPTの提案を鵜呑みにせず，自分の判断で修正や改善を行っている

②3点

・問題をChatGPTに伝えているが，曖昧な質問や丸投げになっている部分がある

・ChatGPTの回答をそのまま試すだけで，自分の考察があまり見られない

・やりとりはあるが，理解の深まりや発展的な質問は少ない

③1点

・ChatGPTとのやり取りがない，またはほとんどない

・「このコードを書いて」と依頼して，提出したプログラムがChatGPTの生成したものと酷似している

・ChatGPTから得たコードを理解せずにコピペしている形跡がある

以上より，一つの実習課題に対して，最終的に評点は10点，8点，6点のどれかとなる。

おわりに

筆者は，商学部において，2013年度からプログラミング教育を担当してきた。その中で，いくつかの観点から授業内容の改変を進めてきた。その上で，今回はプログラムのデバック工程において，教員による個別指導ではなくAIによるチュータリングを試行することにした。そして，2025年度の秋学期において，そのための本格的な実証実験を予定している。Moodleにおいて取得した学習データを分析することで，AIチュータリングの有効性について評価する予定である。

なお，本原稿は，ソフトウェア技術者協会教育分科会において講演した内容をもとに加筆している[15]。

参考文献

[1] 村田美友紀，嘉藤直子，大月美佳，掛下哲郎：生成AIによるプログラミング教育のパラダイム転換と教育支援ツールに関する研究構想，情報処理学会「情報教育シンポジウム2024」，2024年
[2] 佐藤美唯，野口結衣，伊東和香，梶浦照乃，高野志歩，倉光君郎：生成AIを活用したプログラミングに重要なスキルの予備調査，人工知能学会第38回年次大会，2024年
[3] 佐々木虎太郎，伊藤　恵：生成AIを活用したプログラミング演習支援のためのバーチャルTAの提案，日本ソフトウェア科学会第41回大会講演論文集，2024年
[4] 原田紗季，山口大成，丸山浩平，森本康彦：生成AIを用いたペアプログラミングによるプログラミング自己学習方法の開発，日本教育工学会論文誌 48(Suppl.)，pp. 197–200，2024年
[5] 河村一樹：JavaScriptによる情報教育入門，大学教育出版，2011年
[6] 河村一樹：講義レスによるプログラミング実習教育の試み，情報処理学会研究報告，Vol. 2015-CE-128(21)，2015年
[7] 河村一樹：開講コマの違いによる学習進捗の相違について――自学自習ベースのプログラミング教育の場合，情報処理学会研究報告，Vol. 2017-CE-139(8)，2017年
[8] 河村一樹：科目「プログラミング基礎」における対面授業とオンライン授業の比較，情報処理学会論文誌教育とコンピュータ，Vol.8，No.1，pp. 100–107，2022年
[9] 河村一樹：JavaScriptによるプログラミング講座，近代科学社，2024年
[10] 河村一樹：プログラミング実習における学習効果について――自作プログラミング対写経プログラミングの比較，東京国際大学論叢 人間科学・複合領域研究，第9号，2024年
[11] 河村一樹：Moodleを用いたプログラミング教育における学習効果の比較－自作プログラミング対写経プログラミング，e-Learning教育研究，第19巻，pp. 1–11，2025年
[12] 河村一樹：Moodleを用いた自学自習ベースのプログラミング教育，東京国際大学論叢 人間科学・複合領域研究，第6号，2021年
[13] 河村一樹：電子掲示板を用いたチュータリングの試み――プログラミング教育での適用――，e-Learning教育研究，第7巻，pp. 33–42，2012年
[14] 河村一樹：Moodleを用いたプログラミング教育における評価方法について――「正確性」と「迅速性」を加味した結果――，e-Learning教育研究，第14巻，pp. 34–42，2020年
[15] 河村一樹：生成AIを用いたプログラミング教育――AIチュータリングの試み――，ソフトウェア技術者協会教育分科会第27回教育事例研究会2025，2025年