Comparative Analysis of Generative AI Performance in University Programming Courses

Eero Suomalainen, Erno Vanhala, and Jouni Ikonen

LUT University, Yliopistonkatu 34, 53850 Lappeenranta, Finland
eero.suomalainen@lut.fi
erno.vanhala@lut.fi
jouni.ikonen@lut.fi

Abstract. Generative artificial intelligence (GenAI) has had an impact on university education and also to the software industry. This can be seen in many different courses where GenAIs can complete exercises, exams, and even whole courses by themselves. This study aims to find out how effectively different GenAI tools can complete exercises in programming courses. Our objective is to find how GenAI's performance varies across courses and what differences there are in results between multiple GenAI tools. During the study, five programming courses were evaluated using five different GenAI tools. These tools were able to solve 43–90% of programming assignments and 80–100% of quizzes. In practice, the result is that any of the selected GenAIs could have been used to pass the majority of all the tested course parts. Results indicate that we have to rethink how courses' tasks are designed and that courses need to have an element where learning is verified in a controlled environment to verify that we are educating software professionals for the industry.

Key words: Artificial Intelligence, cheating, education, programming

1 Introduction

The release of ChatGPT, chatbot built on generative pre-trained transformers (GPT) models, and the generative artificial intelligence (GenAI) boom following it has garnered significant interest, attracting a large user base to AI tools that can create texts, images and videos from user input. GenAIs can also provide answers to user queries, although their responses are not always accurate. As these tools can produce artefacts resembling human-created work, there is a chance that students could skip manual work in education by using GenAI tools.

Previous studies have demonstrated that GenAI can pass university-level programming courses, sometimes achieving the highest grades [47, 9, 11, 15, 16, 31, 4, 34, 43, 22, 56]. Different types of GenAI tools and models, such as ChatGPT and GitHub Copilot, have been tested in these studies, often focusing on programming languages commonly taught in universities. In addition, research has examined how the use of GenAI affects students' learning on courses [36, 42], what role GenAI could have in course curriculum, what benefits it has for student use [9, 14] and how to combat likely misuse of GenAI on courses [37].

However, previous studies on how GenAI's ability to solve coursework tend to focus only on one course per research and on a single GenAI tool. These studies often favour the most common and freely available GenAIs intended for code generation. In this study, we use multiple different GenAI tools and models to complete course exercises, aiming to explore what other GenAI's besides ChatGPT can be used for this purpose. This study also extends the findings of Suomalainen [46]. Our research addresses the following research questions: **RQ1**: What previous research has been conducted on using GenAI to complete university-level programming courses?, **RQ2**: What different GenAI tools can be used to solve exercises in programming courses? and **RQ3**: How well do the selected GenAIs perform on university programming course exercises?

Literature review addresses RQ1, while RQ2 involves inspecting and testing GenAIs. A subset of these GenAIs is then selected to complete course exercises and afterwards their results are analysed to form the artefact for RQ3. The aim is to mimic students' behaviour and see how much they could get points from the course tasks if they are just using GenAIs to try to pass the courses.

2 State of the art of using AI in programming courses

A literature review was conducted on the topic of the use of GenAI tools (often called *chatbots*) in programming courses. The literature review follows snowballing guidelines by Wohlin [51], beginning with a core set of articles gathered from academic databases and subsequently expanded through both backward and forward snowballing. Institute of Electrical and Electronics Engineers (IEEE) Xplore and Web of Science (WoS) were used as sources to gather the core set of articles. In both academic libraries following search query was used: *("University" OR "College") AND "AI" AND "Course*" AND "Programming"* and in WoS search was also limited to the topic (title, abstract and keywords). This resulted in 200 articles in IEEE and 27 articles in Web of Science. These 227 articles were filtered based on their title and abstract at first and later on by full text on how they answer at least one of the following questions: *how the use of GenAI affects students learning on courses*, *how GenAI can help in learning course topics* or *how well GenAI alone can pass programming courses*. First, a core set of 14 articles was identified. Forward snowballing added 15 articles, while backward snowballing contributed 11, culminating in a total of 40 articles. The literature review was conducted at the end of January 2024 and the beginning of February 2024. Due to time constraints, the process of gathering a core set, forward and backward snowballing was conducted only once. The literature was updated in June 2025, when the original databases were re-searched for new journal articles published in 2024 or 2025. 12 additional core articles were found.

The articles found can be divided into four distinct themes (Effects on learning, Use cases for GenAI, Performance of GenAI in courses, and Response to potential problems), with some articles fitting into multiple themes. These themes emerged by analysing the core topics of each article and grouping related insights. Three themes align with the questions used for article selection and the fourth theme was found during article analysis.

2.1 Effects of using Generative AI on student learning

Numerous studies have explored the effects GenAI (such as ChatGPT and GitHub CoPilot) has on students' learning in programming courses. Findings of Sheese et al. [42] suggests that students who used CodeHelp (a GenAI with guardrails preventing it from providing straight-up solution) for help in course topics more frequently also performed better in the course, although the study states that this does not suggest or support claims that higher usage results better performance. However, when Carreira et al. [6] used a chatbot of their creation as a teaching assistant on an introductory programming course and later conducted a survey to students, the results indicate that using a chatbot as a teaching assistant can facilitate students' learning. Another study examining students' experiences with a chatbot designed to teach Structured Query Language queries found responses to be positive about its usefulness, suggesting that the chatbot is a useful tool in learning [38]. Chatbots can be used to summarize code and explain different aspects of a given code snippet, such as time complexity [29], what mistakes and errors might happen when writing it and how to avoid them [29, 35], help to refactor code [17], acquire background knowledge [18] and provide simplified explanations of more complicated concepts [26]. In addition, chatbots can be used to give more personalised hints that are adjusted to the problem at hand [27] and as a brainstorming companion to support creativity [5, 23]. As chatbots give the answer almost instantly to the question, their speedy assistance can help students save time and effort [26]. In addition, using chatbots during courses exposes students to emerging technologies in a structured manner [20] and, as Ji et al. [23] and Wang et al. [49] found, chatbots can ease the students' learning process by alleviating their cognitive load.

Many studies also highlight negative impacts GenAI has on learning if students solely rely on it to solve assignments [32, 36, 5, 17, 49]. Personal development may suffer [5], critical thinking might reduce [17, 23, 49], plagiarism may go unnoticed, and knowledge on the topics of the course can be lacking [32]. In addition, students' self-efficacy has been reported to decrease if a chatbot is used as an advanced search engine offering ready-made solutions [8]. Despite these concerns, unless instructed otherwise, students are likely to choose to use ChatGPT to do homework on courses, including both asking for help and asking to solve the exercise for them (some not even consider the latter as cheating) [3].

The overall impact of GenAI on student learning largely depends on how it is utilized. As Banić, Konecki and Konecki found out in their research [2], after students performed certain algorithmic tasks on Python with ChatGPT they felt programming to be interesting and they learned faster with ChatGPT but they also agreed more with statement "I feel that I don't need to know how to program because I can create programs using ChatGPT" than before the tasks. The study by Suh et al. [45] found that beginner-level programming students did not significantly benefit from using the chatbot and even performed worse on exams but more intermediate-level students the chatbot seemed more helpful, suggesting that using chatbots is more effective when students have some prior knowledge on the topic. Students point out the possibility for misuse when using ChatGPT in studies but also acknowledge its benefits in assisting [30]. And while chatbots can ease students' cognitive load, it is important to keep an appropriate level of cognitive challenge to promote deep learning [49]. Other concerns arise from the fact that there are inaccuracies in GenAI's responses [8, 24] and it's not always clear how to apply solutions it provides to code queries [8]. In addition, students have reported the process of making an account and having to log in to use these GenAI services as a hindrance in the information-seeking process [26].

2.2 Use cases for Generative AI in education

One prominent use case for chatbots in university courses was to use them in a role similar to a teaching assistant (TA). Roldán-Álvarez and Mesa reported in their study [40] that the use of GenAI as a tutor has helped students to pass obstacles more quickly and learn from their mistakes more efficiently than traditional tutoring. This is because GenAI provides instant replies, coding examples and in-depth explanations to course topics. As TAs, chatbots would likely be most useful when explaining contextual applications for course's concepts and providing examples that are relevant to course topics [48, 19, 34, 35, 17, 23] but they could also provide guidance and help with syntax errors students are facing [47, 35], address queries about documentation and templates related to course assignments [33] or provide code reviews for student homework [9, 36]. In addition, chatbots are not limited to text-only answers but can also assist students in other forms, such as images and interactive simulations [21]. As such, using chatbots in university education can facilitate the learning process and enhance critical thinking if students do not rely solely on chatbot-provided answers [39, 17]. Another potential application is the use of chatbots as peers in a peer-instruction model in courses as results indicate it improved students' performance in skills and conceptual understanding when compared to traditional peer learning and students have reported higher learning, engagements and satisfaction through faster response and opportunity to learn from the GenAI [14].

To aid in learning, students could also use chatbots to create more exercises for self-practice when they wish to develop their skills further [10, 21]. It has been reported that GPT-3.5's predecessor GPT-3 is able to create well-crafted exercises for beginner to intermediate programming courses that focus on flow structure, use of Application Programming Interface (API) and inheritance, while error detection was particularly bad [44]. In addition, GPT-3 could create unit tests for programming exercises [9]. In an academic context, chatbots can also be used to provide suggestions to students on appropriate literature for self-study or literature review [10]. Students could also evaluate strengths and weaknesses of the chatbot by creating the same artefacts with a chatbot that students themselves created and compare their own manually created ones to those of the chatbot [20].

Another use case is to provide personalized learning with chatbots. For example, a system was proposed by Wu et al. [54], which can generate personalized tutorials based on students' learning preferences and adjust them to their knowledge levels. Personalized learning can also adapt to students' individual needs, like providing more resources for slower learners and more advanced guidance for faster ones [49]. For learning paths, a system called SKYRAG was proposed by Setyawan Soekamto et al. [41], which addresses weaknesses of the traditional retrieval augmented generation (RAG) by providing more accurate keyword retrieval for better personalization and creating more suitable learning paths.

However, in all of these cases, it must be kept in mind that chatbots are not 100% reliable and are susceptible to occasional errors and misleading information [35]. For example, ChatGPT's accuracy on topics may vary as it depends on the information it was trained with, phrasing of the prompts and what is the context of the questions [25]. When Jalil et al. (2023) [22] questioned ChatGPT on topics of software testing courses, the GenAI was able to give a correct answer 49.4% or a partially correct answer 6.2% of the time. ChatGPT is not efficient when used as a definitive guide [36, 25] or completely reliable in evaluating code quality or correctness but it can provide many suggestions on how to improve code modularity and structure [1].

2.3 Performance of Generative AI in courses

Chatbots have been tested on a variety of tasks including object-oriented programming (OOP) in Java [9, 34], Python programming [11, 47, 15, 16], data structures and algorithms exam including true/false (TF) and multiple choice (MC) questions. They have also been used to write pseudo-code and drawing diagrams [4], written exams and essays, and programming assignments in cyber security [31], embedded systems quizzes composing of TF questions, code analysis, code completion and writing code from scratch [43] and an exam on Swift programming language and SwiftUI [56]. In all of these cases, chatbots can achieve passing grades or higher.

GenAI struggle most with complicated concepts such as self-balancing binary search trees like AVL-trees [53] and interpreting non-textual information like API documentation and Unified Modelling Language (UML) diagrams. In a study by Zhang et al. [56], most of the time GPT-3.5 and GPT-4 models got lower scores as the sections of the exam got more challenging. The structure of task descriptions can have an effect on the chatbots' performance as GPT-3 was found to struggle with simple exercises embedded in lengthy, overly descriptive, or contextually unnecessary text [47]. Additionally, task-specific restrictions, such as prohibiting certain features or requiring specific formatting, affected performance.

2.4 Response to potential problems

Some introductory programming instructors wanted to ban the use of GenAIs in courses, while others wanted to implement them in some way to prepare students for future work life [28]. One potential response is to design programming exercises in a way that it is not possible to solve them simply by giving inputs to the chatbot [9, 34]. To do this, Chan et al. [7] created a method that uses ChatGPT to create exercises for programming courses by iteratively mutating the exercise until it cannot be solved simply using ChatGPT.

Courses could also teach students how to write correct prompts to a chatbot so that it can create accurate coding solutions [12], as a study found out many students may have problems with creating well-structured queries to the GenAI, resulting in poor responses [42]. Or teaching topics by fact-checking chatbot's solutions if it is known to perform poorly on the topic [55]. In addition, chatbots can be integrated with traditional teaching but must be done carefully, so that the chatbot remains as a supportive tool and does not become a replacement for human instructions [35, 17, 23].

Other responses include focusing on problem solving and mathematical reasoning over coding proficiency [13], develop students' critical thinking skills to analyse chatbots responses better [17, 23, 49, 21], distinguish accurate, partially correct and misleading answers and encourage students to verify information from reliable sources[18], adjusting grading criteria to accommodate the use of chatbots in exercises [24] or sifting the focus of the grading more on the whole process instead of the end product.

3 Research process

The process of evaluating GenAIs' performance followed the guidelines of an experiment. This approach involves establishing a predefined structure in which actors perform specific tasks while the researcher observes their performance [52]. Actors in this case are chosen GenAI tools and tasks course exercises. To help define the preset frame, the first course's exercises that were completed with GenAIs served as a "pilot course" to determine the appropriate steps for evaluating GenAIs' performance. Based on the literature review, a hypothesis was formed that "selected GenAIs should be able to complete exercises", meaning that they should obtain at least the minimum required score to pass the exercises.

The evaluation process also followed a simulation method, where a model is created which imitates behaviours of interest and its performance is observed [50]. The model here would be a student on these courses as the study tries to imitate how well a student could succeed by solely relying on the GenAI.

3.1 Selection of GenAI tools and models

Main criterion for choosing which GenAI tools was to select tools that students are potentially using. These tools were assumed to have a conversational interface, allowing users to input prompts and receive generated outputs. As most of the exercises were programming exercises, GenAIs also had to be able to write code. As ChatGPT provided both options, an initial search for potential GenAI tools was done by searching "AIs similar like ChatGPT" and "Programming AIs similar like ChatGPT". The results were used to get an understanding of what GenAIs were available at the time. To supplement this, we surveyed students (N=187) to see what GenAIs they use. 74% of the respondents reported using ChatGPT, 3% mentioned GitHub Copilot and 4% listed Gemini. Other GenAIs also appeared in student responses, like Claude, Microsoft Copilot and Bai Du, but they were mentioned less frequently. Some of the students reported that they did not use any GenAIs. In addition, our industry contact reported using Codeium, which was also identified during the initial search.

The identified GenAIs were evaluated based on their availability in Finland (i.e., no VPN needed), their ability to generate code and have a conversations, are they behind a paywall as free-to-use was preferred to simulate student use. The aim was to find five different GenAI tools and models to be used in the study for completing exercises. Based on this evaluation, the following tools were selected:

GPT-3.5 was developed by OpenAI and is the model that the free version of ChatGPT used until May 2024. GPT-3.5 is a language model trained to produce text and has been optimised with Reinforcement Learning with Human Feedback where human demonstration is used to guide the model to desired output [1].

GPT-4 was, before May 2024, the most advanced OpenAI's GPT model in consumer use. It is meant to be more reliable, creative and able to handle more complex tasks than GPT-3.5. It has been replaced by GPT-4o mini in free-to-use plan [2] and GPT-4o, o1 and o1-mini in paid subscriptions in November 2024.

Codeium is a GenAI programming tool designed for software developers. It features autocomplete, search and chat features, of which the last one was used in this study. Chat feature uses both Codeium's developers' own proprietary model and Open AI APIs in its operations [3]. The free version of Codeium was used in this study, which utilised GPT-3.5 model. As of December 2024, the plan with monthly cost offers GPT-4o support.

GitHub Copilot is an AI coding assistant developed by GitHub that has been trained on public natural language and code sources, including public GitHub repositories. GitHub Copilot uses Anthropic's Claude 3.5 Sonnet, Google's Gemini 1.5 Pro, and OpenAI's GPT-4o, o1-preview and o1-mini models [4], of which GPT-4o was used as it was only one with unlimited access in Finland at the time of November 2024. GitHub Copilot offers only paid subscriptions but students can use it for free through GitHub Pro.

Gemini is a GenAI model developed by Google [5]. For free, it offers Gemini 1.5 Flash model while more powerful Gemini 1.5 Pro model is included in paid Google One AI Premium subscription. 1.5 Flash model was used in this research.

ChatGPT, GitHub Copilot and Gemini were chosen as they were mentioned most in the student survey. We chose to include both at the time free (GPT-3.5) and premium (GPT-4) ChatGPT GenAIs to create a comparison between two

[1] https://help.openai.com/en/articles/6783457-what-is-chatgpt
[2] https://openai.com/index/gpt-4o-and-more-tools-to-chatgpt-free/
[3] https://codeium.com/faq
[4] https://github.com/newsroom/press-releases/github-universe-2024
[5] https://gemini.google.com/faq

different models from the same service provider, even though premium model requires paid membership to use. During the data collection process free and premium ChatGPT models were updated. As such, models we used in this research can be considered legacy models at this point. Finally, we included Codeium due to recommendation and its focus on programming.

3.2 Completing course exercises

Table 1. List of courses, their content and types of exercises.

ID	**Course**	**Year**	**Size**	**Content**	**Inspected course parts**
CS1	Introduction to Programming (in Finnish)	1st year	6 ECTS	Programming basics with Python language, history and current state of programming, good programming style	- Programming exercises - T/F quizzes
OOP	Object-Oriented Pro-gramming (in Finnish)	1st year	6 ECTS	Basics of Java language, classes and inheritance, graphical UI with Android Studio	- Programming exercises - T/F quizzes
C	Principles of C-Programming (in Finnish)	1st year	3 ECTS	C programming language, pointers and dynamic memory allocation, good programming style	- Programming exercises
Web1	Introduction to Web Programming (pilot course, in English)	3rd year	3 ECTS	Web standards like HTML, CSS and JavaScript, browser environment with Document Object Model, building websites with commonly used tools	- Programming exercises - T/F quizzes
Web2	Advanced Web Applications (in English)	3rd year	6 ECTS	Use of APIs, creating interactive server and client software (Node.js and React), managing web software with MongoDB	- Programming exercises - T/F quizzes

This study focused on the selection of bachelor-level programming courses, representing the stage in the degree program where students are acquiring foundational programming skills. The objective was to evaluate how effectively GenAI tools can perform at this level of university education. Exercises from five programming courses were included as part of this study. All courses featured programming assignments and most of them also included quizzes, as seen in Table 1. Programming exercises typically consisted of five tasks per week and were either their separate tasks or part of one assignment divided into five steps. All of the quizzes were in True/False format. These quizzes can be considered quite straightforward, thus their weight in the final course grade is relatively low, around 10%. The main purpose of the quizzes is to make sure students have reviewed all the material related to the topic before they start the programming assignments. The automatic evaluation of programming exercises was done with an automated assessment system used to evaluate coding assignments. Quizzes were created and graded by tools from the learning management system and Android Studio tasks in Object-Oriented Programming were manually graded.

In all courses, exercises and larger course projects are graded automatically or manually without meeting the student or the student group, and exams are done in a monitored exam hall with restricted-access computers (no use of GenAI is possible).

Most of the courses use a 100-point system or a similar system when grading students. 50 points are required to pass the course and receive grade 1 and every 10 points increase the grade by one, resulting in the highest grade of 5 with 90 points. All exercises contribute to these points but students cannot pass courses with exercises alone as there are other mandatory parts, such as course project or exam. Usually half of maximum points available from exercises are needed to pass the course (in addition to minimum 50 points total). Exceptions here are CS1 and C courses where exercise sections have their own grades and course grade is calculated from weighted average of grades from exercises, exam and course project. Still, to pass exercises with grade 1, half of them must be solved.

The process of completing exercises with GenAIs was as follows:

1. The assignment was copied from instructions for a prompt for GenAI.
   - If the prompt needed additional instructions, (e.g. definition of programming language), they were added and irrelevant information for completion of the task was removed.
   - If instructions referred to the previous material, the code was added to the prompt as text at the end.
   - For True/False quizzes, the claim was turned to a question.
   - If the quiz task was to fill sections with given options, both sections and options were added to the prompt with the task instructions.
2. The type of exercise determines what is done with the GenAI's output
   - For programming exercises the solution was copied to the code editor, the file was zipped and submitted to the automatic assessment system.
   - For quizzes the correct solution was chosen based on the GenAI's answer.
3. If the GenAI's output was insufficient, then the task was reiterated
   - The GenAI was given a description of what was missing (the output from assesment system) and what needed to be changed.
   - With T/F quizzes, reiteration was only done if the output did not clearly say the claim was either true or false.
4. After iterations and reiterations, the results were written down.

The process of iteration and reiteration was designed to simulate how a student with no prior knowledge of the subject could complete tasks using only the GenAI. As such, when reiterating, the prompt only included the response from the automatic assessment system, a description of what was missing based on exercise instructions or a request for a more precise answer. The process did not include any other prompt engineering besides what was described in this section.

4 Results

Five GenAIs, GPT-3.5, GPT-4, Gemini, GitHub Copilot and Codeium, were tested on weekly exercises in all courses. Web1 acted as a pilot course to determine how many iterations per exercise task we needed to see if a GenAI can solve the task. Apart from GPT-4 and GitHub Copilot, all were free to use. Students can obtain GitHub Copilot for free, and all tools featured a simple sign-up process, such as using Google or GitHub accounts.

Overall, the GenAIs performed well in the exercises. To pass, they needed to solve at least half of the available tasks in exercises and in most of the weekly exercise parts, they managed to pass that limit. Only Codeium failed programming exercises in one course, OOP. It can be concluded that students are very likely to pass exercises in these courses by using GenAI alone without trying to solve them by themselves and if a student decides to do that, it is not very difficult, as most of these GenAIs are free to use and easily accessible for students.

Score-wise wise GenAIs also performed well. In most cases, they passed at least half of the exercises. In quizzes, GenAIs performed very well (Figure 1), always getting at least 80% correct and most often getting 90% or more. Performance was weaker in CS1 and OOP courses, which were in Finnish, while in Web1 and Web2 courses, GenAIs performed better, which in turn were in English. However, while GenAIs performed well in quizzes, programming exercises are worth two or three times more in course grading. So, accomplishments in quizzes do not have the same impact as the same level of performance would have in programming exercises.

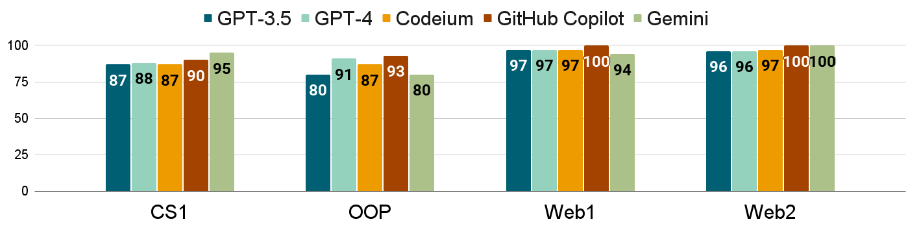

Fig. 1. Total percentage of quiz questions GenAIs managed to pass in all courses.

GenAIs managed to solve at least half of the available programming tasks and assignments in most courses, with only exception being Codeium in OOP (Figure 2). GitHub Copilot consistently delivered the best performance, while Codeium and GPT-3.5 often performed the worst. Between courses, Web1 and Web2 had a higher average pass percentage (73.2% and 72.2% respectively) than CS1, C and OOP (69.8%, 66% and 63.2% respectively). In this sense, GenAIs performed better on courses that are designed for third-year students (Web1 and Web2) than they did on first-year courses and Web2 itself is generally considered to be the most difficult programming course during bachelor's studies. Furthermore, CS1 is a prerequisite for all of these courses and while average performance in CS1 was better than it was on C and OOP it was also worse than in Web1 and Web2. Although averages between Web2 and CS1 are not very big and the Web2 average is affected by GitHub Copilot's and Gemini's scores, which for those GenAIs were their best scores across all five courses, only GPT-4 had its worst performance in Web2. Another aspect that can affect these results is that CS1, C and OOP have strict requirements for each task's output, as it has to match exactly what is the expected output, whereas in Web1 and Web2, automatic grading only expects certain elements or API calls. OOP had the lowest average percentage due to each AI's poor performance in Android Studio programming exercises.

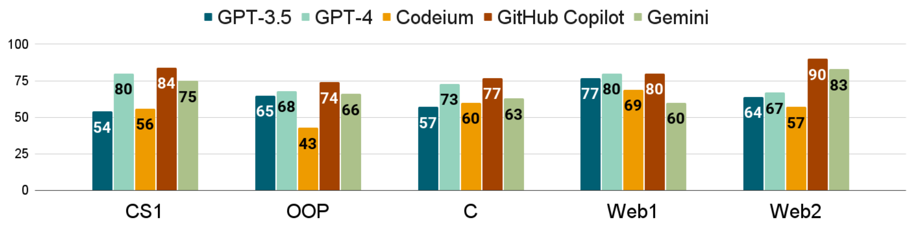

Fig. 2. Total percentage of programming exercises GenAIs managed to pass in all courses.

There were some common instances where GenAIs struggled. Reading files often caused troubles as the format of the text file was not always clear to GenAIs and it was not always easy to say if problems in the printed output originated from it. The same happened with APIs and how they were structured and how GenAI should read them. If the previous week's solution was used in the next programming exercise, it was more likely to fail, as the solution needed to be added as text to the prompt, making it more complex. Similarly, programming exercises where the entire exercise was one assignment and tasks were steps in it, failing one task could result in failing some or even all following tasks, depending on how much they rely on previous ones. Still, the most common cause for failing was the complexity of the programming task. A common trend was that exercises from the first weeks were generally completed successfully, while later weeks saw more failures. Typically, the simpler tasks at the beginning of each week were more likely to be passed.

5 Findings and discussion

In regard to the research on using GenAI in university programming courses (RQ1), multiple studies have explored this topic. Since GenAI became a hot topic after the launch of ChatGPT with model GPT-3.5, most of the research is very recent, with the majority of studies being from 2023 and onwards. From these studies, four major themes were identified, focusing on how the use of GenAI affects students' learning, how GenAI could be utilised in courses, how GenAIs have performed in courses and what sort of response there has been in using GenAI in courses.

Regarding which GenAI tools and models can be used to complete programming courses (RQ2) there are many options at the moment. ChatGPT still has a large monthly user base, many GenAI tools offer both free models and more powerful models for paid subscription, there are a lot of alternatives to ChatGPT and more specialised GenAI tools, such as Gemini and Codeium. Besides GenAIs used in this study, there are many more that could be used instead, like Claude3 and Microsoft Copilot, a combination of all Microsoft's previous Copilot GenAIs.

The main point of this study was to evaluate how selected GenAIs perform in university programming course exercises (RQ3). Results show that GenAIs can pass most, if not all, parts in selected courses and can often achieve a moderately high score in course grading. GitHub Copilot performed best out of all five, with GPT-4 usually coming as a second and Gemini tended to be a bit better than already outdated GPT-3.5 and Codeium in exercises. In quizzes, all GenAIs achieved very high scores and while results in programming exercises were not as good, GenAIs managed to pass at least half of them most of the time. Difficulty of the course (at least on the level of bachelor's studies) doesn't seem to affect GenAIs performance as they performed a little bit better on third-year courses (Web1 and Web2) than they did on first-year courses. The complexity of the exercise, iterating on previous tasks or week's solution, and relaying the format of the file to read or API to use caused trouble to GenAIs. These results indicate that a student with no prior knowledge on the course topics and with no personal input could pass exercises in these courses by solely relying on the GenAI. This aligns with previous research on how well GenAIs can pass courses. GenAIs' struggles identified in previous studies, such as complicated concepts [53] and API documentation [34] are also consistent with those found in this study.

It is possible that a student could pass exercises in courses discussed here using only a GenAI but by doing so might hinder their learning [36], critical thinking [17, 23] and give a lacking knowledge on course topics [32]. These could be prevented by giving focus in grading to the whole process of doing exercises instead of just the end product and having students give detailed statements on their AI usage [37] or emphasising more on problem solving and mathematical reasoning over coding proficiency [13]. Critical thinking skills can be improved if students are taught to analyse chatbots' responses better [23, 21] and distinguish accurate, partially correct and misleading answers and verify information from reliable sources [18]. In the end, teachers are now living in a world where students have the option to use GenAIs and it has and it will continue to form methods and ways of teaching. Our findings fortify existing research and underline that cheating is easier than ever.

There is also the option to use GenAI in a way that boosts learning. One way to do so is to use GenAI as a teaching assistant in courses [40] and GenAI can provide simplified and easy-to-understand explanations to course topics [26]. Personalised learning can also be provided more easily to students as GenAIs can quickly create tutorials and other learning resources based on individual students' preferences [54, 49]. As selected GenAIs in this study could pass exercises in the courses relatively well, they could potentially assist students struggling with topics. Of course, GenAIs are not 100% reliable and, as the study by Suh et al. [45] suggests, the usefulness of GenAI to students seems to depend on how much prior knowledge they have on the course topics, as more intermediate-level students benefited from GenAI more than beginner-level students, so this should be considered when deciding how to use GenAIs in courses.

However, the results here do not paint the whole picture of how GenAIs could pass these courses. Only exercises were considered in this study and their portion of the overall course grade is less than half in all cases. Courses have at least either a course project or an exam, usually both, besides exercises. While it is possible to use GenAI in those, except in supervised exams, this study did not include them and cannot tell how GenAIs would perform in those contexts. Also we do not have any long-term data yet. It would be interesting to see how the GenAI affects to students' skill retention or critical thinking. Also, it is noteworthy to emphasise how these results are valid in the programming education context and might not be the same outside this scope.

6 Conclusions

The results obtained here show that GenAIs can pass most of the different parts of weekly exercises and obtain decent scores, often reaching around 75% of available points. Five GenAIs were selected for this experiment, and every GenAI was used in every course: GPT-3.5, GPT-4, Codeium, GitHub Copilot and Gemini. GitHub Copilot performed better than others and GPT-3.5 and Codeium most often had the worst performance. GenAIs often excelled in quizzes, while performance in programming exercises was not as good but still decent. GenAIs struggled most with complex programming tasks and using external data sources like text files and APIs.

This indicates that a student could pass exercises in these courses by solely relying on GenAI. As previous research indicates, this would likely harm students' learning. This makes teaching challenging as university degree programs are trying to educate software professionals for the industry.

References

1. Anishka, Mehta, A., Gupta, N., Balachandran, A., Kumar, D., Jalote, P.: Can ChatGPT Play the Role of a Teaching Assistant in an Introductory Programming Course? (2023), `https://arxiv.org/abs/2312.07343`, version Number: 2
2. Banić, B., Konecki, M., Konecki, M.: Pair Programming Education Aided by ChatGPT. In: 2023 46th MIPRO ICT and Electronics Convention (MIPRO). pp. 911–915. IEEE, Opatija, Croatia (May 2023)

3. Bego, C.R.: Using ChatGPT for Homework: Does it Feel Like Cheating? (WIP). In: 2023 IEEE Frontiers in Education Conference (FIE). pp. 1–4. IEEE, College Station, TX, USA (Oct 2023)
4. Bordt, S., von Luxburg, U.: ChatGPT Participates in a Computer Science Exam (2023), `https://arxiv.org/abs/2303.09461`, version Number: 2
5. Budhiraja, R., Joshi, I., Challa, J.S., Akolekar, H.D., Kumar, D.: "It's not like Jarvis, but it's pretty close!" - Examining ChatGPT's Usage among Undergraduate Students in Computer Science. In: Proceedings of the 26th Australasian Computing Education Conference. pp. 124–133. ACM, Sydney NSW Australia (Jan 2024)
6. Carreira, G., Silva, L., Mendes, A.J., Oliveira, H.G.: Pyo, a Chatbot Assistant for Introductory Programming Students. In: 2022 International Symposium on Computers in Education (SIIE). pp. 1–6. IEEE, Coimbra, Portugal (Nov 2022)
7. Chan, W.K., Yu, Y.T., Keung, J.W., Lee, V.C.: Toward AI-assisted Exercise Creation for First Course in Programming through Adversarial Examples of AI Models. In: 2023 IEEE 35th International Conference on Software Engineering Education and Training (CSEE&T). pp. 132–136. IEEE, Tokyo, Japan (Aug 2023)
8. Choudhuri, R., Liu, D., Steinmacher, I., Gerosa, M., Sarma, A.: How Far Are We? The Triumphs and Trials of Generative AI in Learning Software Engineering (2023), `https://arxiv.org/abs/2312.11719`, version Number: 1
9. Cipriano, B.P., Alves, P.: GPT-3 vs Object Oriented Programming Assignments: An Experience Report. In: Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1. pp. 61–67. ACM, Turku Finland (Jun 2023)
10. Daun, M., Brings, J.: How ChatGPT Will Change Software Engineering Education. In: Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1. pp. 110–116. ACM, Turku Finland (Jun 2023)
11. Denny, P., Kumar, V., Giacaman, N.: Conversing with Copilot: Exploring Prompt Engineering for Solving CS1 Problems Using Natural Language (2022), `https://arxiv.org/abs/2210.15157`, version Number: 1
12. Denny, P., Leinonen, J., Prather, J., Luxton-Reilly, A., Amarouche, T., Becker, B.A., Reeves, B.N.: Promptly: Using Prompt Problems to Teach Learners How to Effectively Utilize AI Code Generators (2023), `https://arxiv.org/abs/2307.16364`, version Number: 1
13. Dickey, E., Bejarano, A., Garg, C.: Innovating Computer Programming Pedagogy: The AI-Lab Framework for Generative AI Adoption (2023), `https://arxiv.org/abs/2308.12258`, version Number: 1
14. Dos Santos, O.L., Cury, D.: Challenging the Confirmation Bias: Using ChatGPT as a Virtual Peer for Peer Instruction in Computer Programming Education. In: 2023 IEEE Frontiers in Education Conference (FIE). pp. 1–7. IEEE, College Station, TX, USA (Oct 2023)
15. Finnie-Ansley, J., Denny, P., Becker, B.A., Luxton-Reilly, A., Prather, J.: The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming. In: Proceedings of the 24th Australasian Computing Education Conference. pp. 10–19. ACM, Virtual Event Australia (Feb 2022)
16. Finnie-Ansley, J., Denny, P., Luxton-Reilly, A., Santos, E.A., Prather, J., Becker, B.A.: My AI Wants to Know if This Will Be on the Exam: Testing OpenAI's Codex on CS2 Programming Exercises. In: Proceedings of the 25th Australasian Computing Education Conference. pp. 97–104. ACM, Melbourne VIC Australia (Jan 2023)
17. Haindl, P., Weinberger, G.: Does ChatGPT Help Novice Programmers Write Better Code? Results From Static Code Analysis. IEEE Access **12**, 114146–114156 (2024)

18. Haindl, P., Weinberger, G.: Students' Experiences of Using ChatGPT in an Undergraduate Programming Course. IEEE Access **12**, 43519–43529 (2024)
19. Hajj, J.A., Sah, M.: Assessing the Impact of ChatGPT in a PHP Programming Course. In: 2023 7th International Symposium on Innovative Approaches in Smart Technologies (ISAS). pp. 1–10. IEEE, Istanbul, Turkiye (Nov 2023)
20. Haldar, S., Pierce, M., Fernando Capretz, L.: Exploring the Integration of Generative AI Tools in Software Testing Education: A Case Study on ChatGPT and Copilot for Preparatory Testing Artifacts in Postgraduate Learning. IEEE Access **13**, 46070–46090 (2025)
21. Hochmair, H.H.: Use and Effectiveness of Chatbots as Support Tools in GIS Programming Course Assignments. ISPRS International Journal of Geo-Information **14**(4), 156 (Apr 2025)
22. Jalil, S., Rafi, S., LaToza, T.D., Moran, K., Lam, W.: ChatGPT and Software Testing Education: Promises & Perils. In: 2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). pp. 4130–4137. IEEE, Dublin, Ireland (Apr 2023)
23. Ji, Y., Zhan, Z., Li, T., Zou, X., Lyu, S.: Human–Machine Cocreation: The Effects of ChatGPT on Students' Learning Performance, AI Awareness, Critical Thinking, and Cognitive Load in a STEM Course Toward Entrepreneurship. IEEE Transactions on Learning Technologies **18**, 402–415 (2025)
24. Johanyák, Z.C., Cserkó, J., Pásztor, A.: AI-assisted university programming education in practice. In: 2023 IEEE 35th International Conference on Software Engineering Education and Training (CSEE&T). pp. 185–186. IEEE, Tokyo, Japan (Aug 2023)
25. Joshi, I., Budhiraja, R., Dev, H., Kadia, J., Ataullah, M.O., Mitra, S., Akolekar, H.D., Kumar, D.: ChatGPT in the Classroom: An Analysis of Its Strengths and Weaknesses for Solving Undergraduate Computer Science Questions. In: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1. pp. 625–631. ACM, Portland OR USA (Mar 2024)
26. Joshi, I., Budhiraja, R., Tanna, P.D., Jain, L., Deshpande, M., Srivastava, A., Rallapalli, S., Akolekar, H.D., Challa, J.S., Kumar, D.: "With Great Power Comes Great Responsibility!": Student and Instructor Perspectives on the influence of LLMs on Undergraduate Engineering Education (2023), `https://arxiv.org/abs/2309.10694`, version Number: 2
27. Kochmar, E., Vu, D.D., Belfer, R., Gupta, V., Serban, I.V., Pineau, J.: Automated Personalized Feedback Improves Learning Gains in An Intelligent Tutoring System. In: Bittencourt, I.I., Cukurova, M., Muldner, K., Luckin, R., Millan, E. (eds.) Artificial Intelligence in Education, vol. 12164, pp. 140–146. Springer International Publishing, Cham (2020), series Title: Lecture Notes in Computer Science
28. Lau, S., Guo, P.: From "Ban It Till We Understand It" to "Resistance is Futile": How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot. In: Proceedings of the 2023 ACM Conference on International Computing Education Research V.1. pp. 106–121. ACM, Chicago IL USA (Aug 2023)
29. MacNeil, S., Tran, A., Mogil, D., Bernstein, S., Ross, E., Huang, Z.: Generating Diverse Code Explanations using the GPT-3 Large Language Model. In: Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 2. pp. 37–39. ACM, Lugano and Virtual Event Switzerland (Aug 2022)
30. Maher, M.L., Tadimalla, S.Y., Dhamani, D.: An Exploratory Study on the Impact of AI tools on the Student Experience in Programming Courses: an Intersectional

Analysis Approach. In: 2023 IEEE Frontiers in Education Conference (FIE). pp. 1–5. IEEE, College Station, TX, USA (Oct 2023)

31. Malinka, K., Peresíni, M., Firc, A., Hujnák, O., Janus, F.: On the Educational Impact of ChatGPT: Is Artificial Intelligence Ready to Obtain a University Degree? In: Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1. pp. 47–53. ACM, Turku Finland (Jun 2023)
32. Memarian, B., Doleck, T.: ChatGPT in education: Methods, potentials, and limitations. Computers in Human Behavior: Artificial Humans **1**(2), 100022 (Aug 2023)
33. Neyem, A., González, L.A., Mendoza, M., Alcocer, J.P.S., Centellas, L., Paredes, C.: Toward an AI Knowledge Assistant for Context-Aware Learning Experiences in Software Capstone Project Development. IEEE Transactions on Learning Technologies **17**, 1599–1614 (2024)
34. Ouh, E.L., Gan, B.K.S., Jin Shim, K., Wlodkowski, S.: ChatGPT, Can You Generate Solutions for my Coding Exercises? An Evaluation on its Effectiveness in an undergraduate Java Programming Course. In: Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1. pp. 54–60. ACM, Turku Finland (Jun 2023)
35. Ouyang, F., Guo, M., Zhang, N., Bai, X., Jiao, P.: Comparing the Effects of Instructor Manual Feedback and ChatGPT Intelligent Feedback on Collaborative Programming in China's Higher Education. IEEE Transactions on Learning Technologies **17**, 2173–2185 (2024)
36. Popovici, M.D.: ChatGPT in the Classroom. Exploring Its Potential and Limitations in a Functional Programming Course. International Journal of Human–Computer Interaction pp. 1–12 (Oct 2023)
37. Prather, J., Denny, P., Leinonen, J., Becker, B.A., Albluwi, I., Craig, M., Keuning, H., Kiesler, N., Kohn, T., Luxton-Reilly, A., MacNeil, S., Petersen, A., Pettit, R., Reeves, B.N., Savelka, J.: The Robots Are Here: Navigating the Generative AI Revolution in Computing Education. In: Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education. pp. 108–159. ACM, Turku Finland (Dec 2023)
38. Pérez-Mercado, R., Balderas, A., Muñoz, A., Cabrera, J.F., Palomo-Duarte, M., Dodero, J.M.: ChatbotSQL: Conversational agent to support relational database query language learning. SoftwareX **22**, 101346 (May 2023)
39. Rathore, A.S., Sharma, A., Massoudi, M.: Personalized Engineering Education Model Based on Artificial Intelligence for Learning Programming. In: 2021 6th International Conference on Computing, Communication and Security (ICCCS). pp. 1–10. IEEE, Las Vegas, NV, USA (Oct 2021)
40. Roldán-Álvarez, D., Mesa, F.J.: Intelligent Deep-Learning Tutoring System to Assist Instructors in Programming Courses. IEEE Transactions on Education **67**(1), 153–161 (Feb 2024)
41. Setyawan Soekamto, Y., Christopher Limanjaya, L., Kaleb Purwanto, Y., Kang, D.K.: From Queries to Courses: SKYRAG's Revolution in Learning Path Generation via Keyword-Based Document Retrieval. IEEE Access **13**, 21434–21455 (2025)
42. Sheese, B., Liffiton, M., Savelka, J., Denny, P.: Patterns of Student Help-Seeking When Using a Large Language Model-Powered Programming Assistant. In: Proceedings of the 26th Australasian Computing Education Conference. pp. 49–57. ACM, Sydney NSW Australia (Jan 2024)
43. Shoufan, A.: Can Students without Prior Knowledge Use ChatGPT to Answer Test Questions? An Empirical Study. ACM Transactions on Computing Education **23**(4), 1–29 (Dec 2023)

44. Speth, S., Meißner, N., Becker, S.: Investigating the Use of AI-Generated Exercises for Beginner and Intermediate Programming Courses: A ChatGPT Case Study. In: 2023 IEEE 35th International Conference on Software Engineering Education and Training (CSEE&T). pp. 142–146. IEEE, Tokyo, Japan (Aug 2023)
45. Suh, J., Lee, K., Lee, J.: Programming education with ChatGPT: outcomes for beginners and intermediate students. Education and Information Technologies (Apr 2025)
46. Suomalainen, E.: Education in transition: a study on artificial intelligences' performance in university software engineering courses. Master's thesis, LUT University (2024)
47. Torres, N.: Do Robots Dream of Passing a Programming Course? In: 2023 42nd IEEE International Conference of the Chilean Computer Science Society (SCCC). pp. 1–8. IEEE, Concepcion, Chile (Oct 2023)
48. Verleger, M., Pembridge, J.: A Pilot Study Integrating an AI-driven Chatbot in an Introductory Programming Course. In: 2018 IEEE Frontiers in Education Conference (FIE). pp. 1–4. IEEE, San Jose, CA, USA (Oct 2018)
49. Wang, H., Wang, C., Chen, Z., Liu, F., Bao, C., Xu, X.: Impact of AI-agent-supported collaborative learning on the learning outcomes of University programming courses. Education and Information Technologies (Mar 2025)
50. White, K.P., Ingalls, R.G.: Introduction to simulation. In: 2015 Winter Simulation Conference (WSC). pp. 1741–1755. IEEE, Huntington Beach, CA, USA (Dec 2015)
51. Wohlin, C.: Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. pp. 1–10. ACM, London England United Kingdom (May 2014)
52. Wohlin, C., Höst, M.: Special section: Controlled Experiments in Software Engineering. Information and Software Technology **43**(15), 921–924 (Dec 2001)
53. Wollowski, M.: Using ChatGPT to produce code for a typical college-level assignment. AI Magazine **44**(1), 129–130 (Mar 2023)
54. Wu, X., Wang, H., Zhang, Y., Zou, B., Hong, H.: A Tutorial-Generating Method for Autonomous Online Learning. IEEE Transactions on Learning Technologies **17**, 1532–1541 (2024)
55. Zastudil, C., Rogalska, M., Kapp, C., Vaughn, J., MacNeil, S.: Generative AI in Computing Education: Perspectives of Students and Instructors. In: 2023 IEEE Frontiers in Education Conference (FIE). pp. 1–9. IEEE, College Station, TX, USA (Oct 2023)
56. Zhang, Z., Wen, L., Jiang, Y., Liu, Y.: Evaluate Chat-GPT 's programming capability in Swift through real university exam questions. Software: Practice and Experience **54**(11), 2129–2143 (Nov 2024)