AI Technology: Threats and Opportunities for Assessment Integrity in Introductory Programming

Guttorm Sindre [0000-0001-5739-8265]

Institutt for datateknologi og informatikk, NTNU, Trondheim, Norge
guttorm.sindre@ntnu.no

Abstract. Recent AI tools like ChatGPT have prompted worries that assessment integrity in education will be increasingly threatened. From the perspective of introductory programming courses, this paper poses two research questions: 1) How well does ChatGPT perform on various assessment tasks typical of a CS1 course? 2) How does this technology change the threat profile for various types of assessments? Question 1 is analyzed by trying out ChatGPT on a range of typical assessment tasks, including code writing, code comprehension and explanation, error correction, and code completion (e.g., Parson's problems, fill-in tasks, inline choice). Question 2 is addressed through a threat analysis of various assessment types, considering what AI chatbots would be adding relative to pre-existing assessment threats. Findings indicate that for simple questions, answers tend to be perfect and ready-to-use, though might need some rephrasing work from the student if the task partly consists of images. For more difficult questions, solutions might not be perfect on the first try, but the student could be able to get a more precise answer via follow-up questions. The threat analysis indicates that chatbots might not introduce any entirely new threats, rather they aggravate existing threats. The paper concludes with some thoughts on the future of assessment, reflecting that practitioners will likely use bots in the workplace, meaning that students must also be prepared for this.

Keywords: AI, chatbots, assessment integrity, cheating, programming

1 Introduction

Academic integrity in the assessment and grading of students is key to the credibility of university degrees [1]. During the Covid-19 lockdowns, when many universities suddenly needed to switch from proctored on-campus exams to less supervised remote exams, there were many reports of increased exam cheating [2, 3]. There were concerns that students would be able to get undeservedly good grades by delivering work that was not really their own, achieved by collusion with peers or outsider assistance. However, cheating on assessments were a challenge long before Covid [4, 5], and will remain so long after.

Recently, advances in AI have been seen as a new factor that may aggravate cheating problems. Tools like ChatGPT have an impressive ability to write both natural language essays and program code from assignment texts, hence student access to

such tools during an assessment task may enable them to solve problems that would have been beyond them in an unaided situation [6]. For instance, a student who knows little or nothing about programming, might seemingly be able to perform quite well on assessment tasks in a programming course, achieving a passing grade or even a good grade where a fail would have been more correct based on the student's own competence [7].

This paper looks at the issue from the perspective of university-level introductory programming courses, posing the following two research questions: 1) How well does ChatGPT perform on various assessment tasks typical of a CS1 course? 2) How does this technology change the threat profile for various types of assessments?

The rest of this paper is structured as follows: Section 2 presents the research method. Then, section 3 reviews related work. Sections 4 and 5 present results for RQ1 and RQ2, respectively. Finally, section 5 provides the discussion and conclusions of the paper.

2 Research Method

RQ1 is analyzed by trying out ChatGPT on a range of assessment tasks that might be typical for CS1 courses. It was decided to focus on one single programming language for this study, namely Python, which is used in many CS1 courses – including our own. While a complete coverage of all kinds of problems would be beyond this paper, it was sought to achieve reasonable coverage in having tasks at various levels of difficulty, with student average scores typically ranging from 40% to 90%. The tasks targeted various programming concepts, such as loops, branching, different elementary and composite data types, functions and parameter passing, exception handling, input, print, and file read and write. GUIs or object-orientation was not included, as this does not feature in the CS1 course at the authors' department, rather being addressed in CS2.

We also sought to cover various question genres: code comprehension and explanation (e.g., what will be printed / returned? Or explain step by step), code writing, error identification and correction, and various code completion tasks (e.g., Parson's problems, fill-ins, inline choice). Specifically, as the first code writing task, we picked the Rainfall problem – as a well-known problem for which unaided student performance has been thoroughly studied. In addition, we looked at a selection of exam tasks from the authors' university, for which student performance without the help of ChatGPT was known from exam score averages.

RQ2 is addressed through a threat analysis [8] of various assessment types, considering what AI chatbots would be adding relative to pre-existing assessment threats. It must be acknowledged that this analysis is rather commonsensical rather than building on empirical evidence.

3 Related Work

Recently, several studies have investigated the efficiency of AI chat tools in solving programming tasks. Lo [9] in a broader review of literature found that ChatGPT had worked very well for solving assignments in economy, quite well in programming, and not so well in mathematics. However, for mathematics there are other tools already doing that very well (e.g., Wolfram Alpha [10]) so ChatGPT would anyway not be the students' first choice. Piccolo et al. [11] found that ChatGPT worked well on the programming assignments given in a bioinformatics course. Surameery et al. [12] investigated ChatGPT's efficiency in finding and solving programming bugs, concluding that it would be a helpful tool, though also having limitations, meaning that it could not replace the need for professional debugging tools. Kashefi and Mukerji [13] looked at ChatGPT's capabilities in solving numerical programming problems, finding that good answers were provided on many problems, though there were also some limitations and failures on more advanced problems. Savelka et al. performed a study where they looked at ChatGPT's ability to solve several different types of multiple choice questions about code – some formulated only in natural language, others also containing snippets of code [14]. They found that problems described by text only tended to be solved well, while those including code snippets sometimes were wrongly answered by ChatGPT. In another study by Savelka et al. [15] they investigated how well ChatGPT did on all the assignments in an introductory (CS1) and intermediary (CS2) Python course at their university. The assignments range from very simple ones (Python functions / code snippets of a few lines) to larger ones (projects with code bases spread across multiple files). They found that ChatGPT scored far from perfectly – around 70% on the CS1, 55% on the CS2 – but this would still be of substantial help to a low competence cheater who might not have achieved such a score otherwise.

Several researchers have looked into the cheating threat related to ChatGPT. Rahman [16] discussed the usage of ChatGPT for learning and teaching programming, as well as looking briefly at cheating risks. Oravec [17] discussed student cheating by means of ChatGPT as a form of misattributed co-authorship, and suggests that one remedy would be to educate students better on how to document their sources and collaborations (both human-machine and human-human). Ouh et al. [18] found that ChatGPT did well on Java problems described textually – both easy ones and medium complexity problems – but did more poorly if part of the problem was explained by figures / diagrams. They suggested this could be utilized to make assessment tasks that would be harder to cheat on. Manoharan et al. [19] performed a study comparing ChatGPT with Chegg (a question-answer service depending on human experts in low-cost countries) concerning the quality of answers to a range of CS1 and CS2 questions. They found that the answers by ChatGPT were generally as good or better than the answers from Chegg, although the latter cost money and took 30 minutes, while ChatGPT was almost instant and free.

Other authors, while mentioning cheating risks, have focused more on the need for renewal in how to teach programming, and urge to avoid panic on how to avoid cheating. Maher et al. [20] discuss how AI could help the students learn how to program.

Similarly, Kendon et al. [21] argue that it will be futile to make strictly controlled assessments that prevent the usage of AI yet remain valid – instead the technological development should be taken as a wake-up call to rethink how we teach and assess programming competencies. Similar views are stated by Bull and Kharrufa [22], arguing that industry is showing clear intent to increase their usage of AI in the production of code, so this is a competence students also need to master. Denny et al. [23] specifically discuss that an essential future competence for our students is to be able to write good prompts to the AI (so that the AI in turn comes up with code that solves the problem), rather than drilling the students on how to write basic code snippets from scratch, and they also suggest a new type of assessment task – the prompt problem – to target such competence.

4 Results for RQ1: How well does ChatGPT perform?

4.1 The Rainfall Problem

As our first entry in the chat, we used the same phrasing of the Rainfall Problem as provided as "traditional rainfall phrasing" in Lappalainen et al. (2017, p.3), except adding the phrase "in Python" (their task was implicitly in Java) and replacing "array" with "list". We followed up with various additional requests to further explore the capabilities of ChatGPT. Results are summarized in Table 1, where 100% means a perfectly correct response, i.e., in case of code a correctly running solution, and less than 100% means a sub-perfect solution that would likely have caused some point deduction in exam grading. For the follow-up requests in the same dialogue, it was unnecessary to repeat the initial assignment text (then part of established context), one just needed to add specific requests, such as please make another implementation using recursion rather than a loop.

As can be seen from the table, there was only one mistake, and this was a minor one. Asked to provide test cases for the first function, it came up with 5 (normal case, lower limit case, sentinel case, empty list, and all values lower than lower limit). In test case #3, the sentinel value had been changed from 999 to 5, but the expected value had not been changed accordingly. The error was thus just in one code line of 36 total code lines for the test cases. It could be noted though that for the recursive variant of the Rainfall Problem, ChatGPT first proposed a solution that would be sub-optimal – namely using slicing to provide the reduced list in the recursive call – which wastes time since slicing implies copying of the sub-list. This might not have been penalized in a CS1 course as optimal efficiency is seldom stressed (and if it were, the assignment text would not demand a recursive solution for this task). With a follow-up question to avoid slicing, ChatGPT adapted the solution to using indices instead.

All in all, if students had access to ChatGPT during an exam, and some variation of the Rainfall Problem was among the tasks, even the weakest students would likely be able to deliver good solutions for that problem.

Table 1. ChatGPT's results for the Rainfall Problem

Request	Result	Comment
Traditional phrasing (??)	100%	
… and test cases for function	90%	Tiny error in case 4 (of 5)
New solution (-- plagiarism)	*risky*	*Only changed var. names*
…other structure	100%	Using list(filter(lambda…))
…use list comprehension	100%	
…use while-loop	100%	
Rainfall, recursion	100%	Slightly sub-optimal (Slicing)
… without slicing	100%	
Rainfall as script w/ input()	100%	while True:, 'q' to quit
… only sentinel, not q	100%	while True:
… use sentinel while-test	100%	
…add exception handling	100%	
Rainfall, some other lang.	100%	JavaScript, C#, C++
Rainfall, Java	100%	
… in requested context	100%	Cf. Lappalainen

4.2 Other short coding tasks

ChatGPT was tried on various tasks from CS1 exams given in the authors' own university. These exams were given in English and native language versions. We used the English versions as input to ChatGPT, and results are summarized in Table 2.As our first entry in the chat we used the English version of the exam question as given, though notably only the text that could be directly pasted into the chat question field. Hence, illustrations or graphic parts of the Inspera UI (such as drag and drop objects, alternatives for inline choice gaps, etc.) would not be included, and similarly, the question would not include pre-existing code shown in the form of an image (which is sometimes done because the display of code from the standard text editor of Inspera is rather lackluster).

As Table 2 indicates, ChatGPT wrote perfect code for all the six code writing tasks when given a complete task description. The only case that it did not solve perfectly was with an incomplete task description of counting local minima in a 2D numpy matrix. This task was partly explained by an example image. Since the image could not be pasted into the chat field, we tried first to paste just the text (as one would imagine a student faced with the task might also have done). Based on the text alone, ChatGPT assumed that edge elements were not eligible candidates for local minima – while the image clearly indicated that they were. The code written by ChatGPT did present a perfect solution given its assumptions, but disregarding edge elements eliminates a lot of complications from the task (e.g., index errors), so that the resulting code would hardly score anything above 50% on the task. However, thereafter provid-

ing ChatGPT with extra information – which only needed to be a brief follow-up request to also consider edge elements as potential local minima – it delivered a perfect solution to the exam task.

Table 2. Results for other short coding tasks

Code writing	**Stud**	**Chat**	**Comment**
Conditional sum	78%	100%	sum() and list comprehension
Soccer field ok size?	78%	100%	
Exponential string	61%	100%	Added text (code screenshot)
String of factors	65%	100%	Added text (code screenshot)
File to list of lists	53%	100%	Including exception handling
Local minima (text)	--	50%	partly explained by image
…add explanation	36%	100%	
Comprehension	**Stud**	**Chat**	**Comment**
Digital root	74%	100%	9 LOC, recursive function
Symbol count func	70%	100%	15 LOC, loop, if-elif-else
Completion	**Stud**	**Chat**	**Comment**
List -> dict (fill-in)	58%	100%	Paste + insert ? for gaps
Str -> dict (2D Parson)	78%	100%	2 extra Qs to get same lines
Fee brackets (inline choice)	51%	100%*	Must list options to get exactly same
Sequence in matrix (choice)	37%	100%	Options not needed (paste w gaps)

For code comprehension tasks, the typical format in the CS1 course in the authors' university is to present the students with some code with short non-informative variable names, and asking what will be printed or what will be returned from a function given certain inputs. ChatGPT answered these questions perfectly. For such questions, the students would not need an advanced AI tool to cheat – any simple code editor with possibility to run the code would suffice. What ChatGPT offers in addition, however, is a nice natural language explanation of what the code does. Indeed, one does not even need to ask any explicit question – simply paste the code into the chat field, and it will immediately be explained. Subsequently, if you want to know what is printed or returned for specific inputs, just follow up with a short extra question about that, and it will provide the answer, accompanied with a step by step explanation of the code for that specific input.

Some program completion tasks were also tried. These were somewhat more cumbersome – not because ChatGPT had any problems with the tasks as such, but because they could not so easily be pasted directly into the chat field. When copy-pasting the fill-in task, the fill-in fields did not show in the pasted text, so the user would have to explicitly add indication where something was supposed to be filled in. However, this was not very difficult – it would suffice for instance to insert a question mark in the code in each position where the task had a gap.

Similarly, for the 2D Parsons task, implemented as a drag-and-drop question in the e-exam system, it was impossible to copy-paste the drag objects, which were images, not text. However, copy-pasting only the text explaining requirements for the function caused ChatGPT to respond with a function which was essentially the same as the solution, though some minor differences in approach caused some code lines to be slightly different from the options available to the student in the task. In particular, ChatGPT used enumerate in its for-loop while the student only had for i in range… available, and this difference caused two later code lines to be slightly different, too. However, only two extra questions were needed to steer ChatGPT towards the exact same solution as the exam task targeted – namely to use range instead of enumerate, and to name variables in a certain way. A similar challenge emerged with the inline choice question about fee brackets, where the code had 8 gaps, each with 3 alternative options that could be selected. The inline choice fields as such were not possible to paste, as students would need to click on each one to see the three options. However, pasting the code and simply getting "Select alternative" in each gap, was sufficient to have ChatGPT come up with essentially the same solution. For 6 of 8 fields ChatGPT had the same as the solution, while 2 of 8 contained something that did not correspond to any of the three options. The code proposed by ChatGPT was also a correct solution to the task (disregarding options available), but it had chosen a slightly different approach from the teacher's model solution. Again, a couple of extra questions would suffice to get an exactly same solution, for instance asking ChatGPT if it could replace the fragment that was different from any options with one of the three provided options and still get a working solution. For the last completion task we had tried – which had been very challenging for the students with only 37% score in the exam (albeit with penalties for choosing wrong options), ChatGPT did not even need options, it ended up providing the exact solution when pasting the task description and code with the text "Select alternative" in each gap and ask it to replace all occurrences of "Select alternative" to make a working solution.

4.3 Larger Projects

The CS1 course that the authors have been involved in does not have any larger project work as part of the graded assessment. However, as this might be the case in other CS1 courses, it was also explored whether ChatGPT might be of help to students in such a context. If asked to write a Python project of considerable size (for a novice) – say 500 code lines – ChatGPT will respond that it is a chat program and cannot write such long programs. However, you can ask it for an outline. We requested an outline for a Python project to help a household reduce food waste, by somehow warning the user when food items were soon to expire, and then also suggesting recipes that included these items. ChatGPT aptly suggested a structure with a main function and 8 other functions, where food items and recipes would be stored in files. Two follow-up questions yielded proposed structures for the files (CSV for the food items file, JSON for recipes), and code for reading and writing the CSV file, and it would be straightforward to keep on asking for suggestions for code for the other proposed functions, too.

5 Results for RQ2: How does AI change the threats towards assessments?

Exactly how exams and other graded assessments are conducted will vary a lot between and even within learning institutions, and this will impact the risk of various cheating threats. A detailed threat analysis for one specific exam setup was not conducted here, rather we took a more general view, considering some broad types of assessment, with the purpose of identifying how ChatGPT and similar AI tools might change the overall risk profile of assessments. The types of assessment considered were thus four types of written exams: on-site proctored exams using PCs with lockdown, on-site proctored with open PCs, remote proctored exams, and remote unproctored exams; two types of oral exams: on site and remote by video conference, and two types of longer-duration coursework: with and without checkpoints during the semester. Across these various assessment types, there are some broad categories of cheating threats that can be considered: (a) *Impersonation* – another person does the assessment work altogether. For an on-site oral or written examination, this might entail somebody else showing up with a fake ID (or with sufficient resemblance to the candidate that the read ID could be used) (b) *Illegal assistance* – the right person is in the exam room, technically answering the questions, but is doing so by means of hints or answers somehow received from other persons, who could be other exam-takers, corrupt employees, or outside helpers. (c) *Usage of forbidden tools or materials* – like using a calculator in a math test supposed to be done without one, googling for answers, or using cheat notes or a book in a test not allowing such sources. (d) *Plagiarism* – you were allowed to use sources, if explicitly cited, but instead deceptively gave the impression that source content was written by yourself.

Technically, AI is in the category (c) of forbidden tools, and potentially (d) plagiarism if the candidate deceptively presents AI text as self-authored. Traditionally, categories c and d have been less potent for grade gain than (a) and (b). If an impersonator or helper is much more competent than the candidate, this can easily give an F to A grade lift due to cheating. Googling for answers, plagiarizing etc., would seldom lead to A answers unless the candidate is lucky to find a direct hit which perfectly matches the task at hand. Rather, weak candidates cheating in this way would often end up with answers where different parts of code or text do not quite fit together. The thing about AI, however, is that while it is technically in the category c/d (forbidden tool, plagiarism), the quality it delivers – at least for many types of questions – is more on par with what a cheater could achieve by help of a competent impersonator or human assistant during the exam.

There could be several ways of cheating by means of AI, depending on the rules concerning AI usage in each course. We will consider two main options in that respect: (i) *Forbidden usage*: The candidate uses AI while answering an assessment task where AI usage was not allowed. (ii) *Undeclared usage:* The usage of AI was allowed on the condition that the candidate is explicit about it, e.g., precisely reporting what code was generated by AI, and what was written by oneself, or attaching screenshots of all communication with the AI tool. However, the candidate's reporting on this could be dishonest, making excessive claims of self-authoring where code

was really AI generated, and omitting the required attachment of conversation screenshots.

Table 3 presents a summary of various cheating threats that are aggravated by AI. Altogether, there are many more cheating threats related to exams and other assessments, but we deliberately skip discussions of threats for which AI chatbots do not change the risk assessment. For instance, old fashioned tricks like cheat notes, whispering, looking at the answers of neighbors (school exams), plagiarism by copy-paste (home-exams, term papers), and free-riding (group projects; posing to have contributed a fair share while really doing nothing) will still be available to cheaters regardless of the development in AI. The table instead concentrates on ways of cheating which are made easier (less effort needed to cheat), more effective (bigger potential gain in performance by cheating), or less risky (reduced chance of getting caught) by the advent of AI chatbots.

Table 3. Cheating threats most importantly aggravated by AI

Type of exam	Way of cheating (Risk)	Role of AI
Supervised on-site exam with lockdown	Breaking lockdown (L) **Using cellphone (H)** Help via earpiece (?)	Answer search more effective Answer search more effective Outside helper more effective w AI
Supervised on-site exam without lockdown (assuming tool usage allowed)	Plagiarism (M) Prompt sharing (H)	Answers of better quality Prompts are shorter and easier to share than the resulting code
Remotely proctored home-exam with lockdown	Use extra device (H) Use third p. helper (H)	Answer search more effective, quicker Third person helper more effective w AI
Un-proctored home-exam (assuming tool usage allowed)	Plagiarism (H)	Can deliver unique answer which is not caught by plagiarism check
Long-term project / coursework (assuming tool usage allowed)	Plagiarism (H)	AI can deliver unique answer not caught by plagiarism check
On-site oral exam	Hidden ear-piece (L)	AI makes remote helper more effective
Remote oral exam (video conf.)	Use extra device (H)	AI makes search quicker

The contents of Table 3 could be explained as follows:

- *Supervised on-site exam with lockdown:* Since lockdown is typically used to prevent students from accessing the internet and files on their own PC, most such exams will likely have the rule that usage of AI is forbidden. The risk that students circumvent this lockdown is fairly low, since few would have the necessary technical competence to do it, and even then facing some risk of discovery if there is also on-site supervision. The much bigger risk for supervised school exams is the usage of mobile phones, especially during toilet visits [24]. Exam regulations may state that invigilators should be able to see candidates, meaning that the door to the toilet booth should be left open. However, it is extremely rare that invigilators enforce this, as it feels embarrassing for both parties. Googling for answers to a programming problem during a toilet visit might not be fully effective. Unless it is a

very standard and straightforward problem the would-be-cheater would only find partial solutions, which would then have to be adapted to fit the task, meaning that weak students would still have weak answers (though might have achieved the crucial lift from F to E). Chatbots like ChatGPT can do a lot better, giving perfectly fitting answers to typical CS1 tasks, as seen in section 4.

- *Supervised on-site exams without lockdown:* The motivation for using this form of assessment would typically be to enable students to use professional programming environments – and this type of exam might even allow the usage of AI tool support. If AI is allowed, one might think there is no particular cheating threat related to it – and this might be true. Instead, the challenge becomes to develop exam tasks that measure relevant student competence in spite of their using AI. For instance, this could be perceived as moving the task of the student from "writing good code" to "writing good docstrings" (which in turn causes GitHub Copilot to write good code) or "writing clear natural language requests" (which in turn causes ChatGPT to write good code). This does still cause some change to the cheating risk: The docstring or natural language request is typically a lot shorter than the resulting code. Hence, while it would be prohibitively risky and time-consuming for a student to try to help a peer by sending one's entire code to the other candidate, the docstrings or natural language requests can much more easily be shared by sms or similar – typed and read during toilet breaks. Moreover, if the AI generates unique code even with similar prompts, this kind of cheating has much less risk of getting caught for plagiarism than what is the case if sharing the answer code directly.
- *Remotely proctored home exams:* These are rare in Norway, so we will not dwell a lot on them. It may be impossible or risky for the student to use forbidden AI tools on the exam PC, due to lockdown or screen capture by the remote proctoring system. Using a second device instead will be more feasible, if able to do so without looking suspicious on the video. If it is hard to accomplish alone, you could also have a third person helper using AI on your behalf (e.g., using a cable splitter to send the monitor picture also to the next room, outside video surveillance, where the helper can work out answers by means of AI and then just needs to somehow communicate them back, e.g., by wireless communication to the candidate's hidden ear-piece.
- *Un-proctored home-exams and coursewor*k: Here it is typically allowed to use tools and sources, but you may have to declare what you have used, and be explicit about what code you wrote yourself, and what was written by AI. It is however hard to check whether candidates are truthful about this. Similar to on-site exams where tool usage is allowed, the teacher would typically try to change the assessment task, not having something where the assignment text can simply be fed into the AI tool. Instead, one might opt for a task where substantial work is needed by the student to understand and structure the problem, before the student can even begin to pose requests to the AI chatbot. Again, then, the cheating threat would move from the sharing of code to the sharing of natural language requests likely to result in good code. A mitigation against this would be to give each student a different programming assignment for the home-exam, or a different case for the project work, removing the potential for sharing of the natural language requests.

However, this might increase the burden of assessing and grading the work. Moreover, the old threat of getting help from somebody more competent will always loom large over unsupervised home-exams and coursework – and if this helper only needs to write some quick natural language requests rather than hundreds of code lines, more candidates might be able to secure such help.

- *On-site oral exams*: These are considered to have low cheating risk [25], and AI does not change this – at least not currently. In a face-to-face situation with two examiners, it will be very difficult for a student to type on a device to search for answer hints along the way. A device able to interpret voice might be helpful, for instance transcribing the examiner's question, feeding it automatically to ChatGPT, and then delivering the answer back to the student. However, receiving the answer is also hard, as the candidate cannot be seen looking at a device – it would have to be a synthetically spoken answer delivered to the candidate through a hidden earpiece. Unless answers are meant to be very short and factual (which would be poor usage of oral examination relative to its potential for assessing a wider range of learning outcomes), this might not give much gain in performance. In an oral examination having the form of an instant dialogue, the candidate would struggle to respond fluently while at the same time listening to prompts via a concealed earpiece.
- *Remote oral exams via video conference:* These have somewhat higher cheating risk than face-to-face orals, since the candidate has more options in receiving hints from AI, e.g., having a device strategically placed outside the webcam angle, yet in a position which makes it seem the candidate is just looking in the general direction of the screen. Still, cheating is much more difficult in the oral examination than in the written one, especially if it is conducted as a dialogue with frequent twists and follow-up questions, rather than just having the candidate answering one long question [26].

6 Discussion and conclusion

AI chatbots can do very well on short, concise programming tasks that are typical of many CS1 courses – as found in our study, and also in those mentioned in related work. They can also do well on longer programming tasks, though this will require a longer dialogue, where the candidate may first need to ask what functions or classes the program should consist of, then ask for code for each of these. As we have indicated in section 5, even supervised on-campus written exams carry a considerable cheating risk – a likely approach of a would-be cheater is to use a mobile phone with an AI chatbot to seek answers to exam questions. Usage of mobile phones is not a new cheating threat, but ChatGPT or similar will deliver answers of much higher quality than what the candidate would likely achieve by ordinary web search. For home exams, another issue is that while copy-paste from web sources may get you caught for plagiarism, the usage of AI chatbots might not, since the answer could be different every time even if several students ask the same question. AI might not (yet)

beat the quality you could achieve by having a highly competent friend – or hired ghost writer – do the home-exam or project for you. However, not all have access to such a friend, nor the money to hire a ghost writer or use a pay-per-answer web-service such as Chegg – yet everybody can be able to paste the question text into ChatGPT or GitHub Copilot. Hence, AI chatbots have not changed the worst-case cheating threat – which is still that an F level candidate gets his exam done by an A level ghostwriter – but it has made somewhat similar assistance available to everybody, thus "democratizing" a cheating threat that was previously exclusive to the lucky few.

Based on our investigations, as well as advice found in some of the related work, some potential advice are as follows:

- For exams, whether to be conducted at home or supervised on campus, have a mixture of different question types. In particular, avoid having only questions with short textual descriptions requesting short code written from scratch, as this is the task type most aptly solved by ChatGPT.
- Images as a vital part of the task description will make it much more difficult to get a quick solution from current mainstream AI-tools – but take care that visually impaired candidates are not disadvantaged. One use of images might be to convey input-output relations, e.g., with some 2d numpy array as input, this other array should be the resulting output; or maybe the code is going to plot some graphic output. Another usage of images could be to show partly complete code where the students are going to fill in what is missing. With some digital exam systems, there is good reason to use images for this anyway, as writing the code directly as text in the system (e.g., Inspera) gives a poor display of the code.
- Fill-in tasks can be more cumbersome to solve with AI than tasks where the code is to be written from scratch. The fill in fields, especially inline choice, do not paste well from digital exam systems like Inspera and into a text editor or input field of an AI chatbot. Although the AI may come up with an ok generic solution for the task, this solution might not fit the one used by the question author, in which case it will not be obvious to the student what to put in the various fill-in gaps in the code.
- Especially for home-exams, remember that AI chatbots is not the only cheating option, and maybe not even the most potent one. Students who have access to a much more competent friend who can do the exam for them, or funds to hire such help online, would be able to come up with good solutions even for tasks where AI chatbots currently fall short.
- If the learning outcomes of your course makes it viable to allow the usage of AI chatbots, this could be a good idea, since allowing it sort of makes that type of cheating disappear. Still, remember that this will not make all cheating disappear. As the challenge for the students moves from code writing to prompt writing, cheating is likely to move along, for instance to sharing of prompts instead of sharing of code or having a more competent person write the prompts for you.

Hence, countermeasures against AI chatbots should not be your only concern when it comes to cheating – there should be a wide range of countermeasures, not only focusing on control and punishment, but also addressing the factors that make stu-

dents want to cheat in the first place, such as test anxiety, grade pressure, etc. If cheating can be reduced by giving students an improved feeling of mastery during the semester, thus reducing grade anxiety, this will be better than controlling and prosecuting cheating, which causes a lot of labor both for teachers and the university administration, as well as cost to the society in terms of students being excluded from their studies for 6 or 12 months due to cheating.

References

1. Sheard, J., Butler, M., Falkner, K., Morgan, M., Weerasinghe, A.: Strategies for maintaining academic integrity in first-year computing courses. In: Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education, pp. 244-249. (2017)
2. Bilen, E., Matros, A.: Online cheating amid COVID-19. Journal of Economic Behavior & Organization 182, 196-211 (2021)
3. Nguyen, J.G., Keuseman, K.J., Humston, J.J.: Minimize online cheating for online assessments during COVID-19 pandemic. Journal of Chemical Education 97, 3429-3435 (2020)
4. Cizek, G.J.: Cheating on tests: How to do it, detect it, and prevent it. Routledge (1999)
5. McCabe, D.L., Butterfield, K.D., Trevino, L.K.: Cheating in college: Why students do it and what educators can do about it. JHU Press (2012)
6. Cotton, D.R., Cotton, P.A., Shipway, J.R.: Chatting and cheating: Ensuring academic integrity in the era of ChatGPT. Innovations in Education and Teaching International 1-12 (2023)
7. 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. (2023)
8. Buyens, K., Win, B.D., Joosen, W.: Empirical and statistical analysis of risk analysis-driven techniques for threat management. In: The Second International Conference on Availability, Reliability and Security (ARES'07), pp. 1034-1041. IEEE, (2007)
9. Lo, C.K.: What is the impact of ChatGPT on education? A rapid review of the literature. Education Sciences 13, 410 (2023)
10. Richardson, S.: Mathematics assessment integrity during lockdown: experiences in running online un-invigilated exams. International Journal of Mathematical Education in Science and Technology 53, 662-672 (2022)
11. Piccolo, S.R., Denny, P., Luxton-Reilly, A., Payne, S., Ridge, P.G.: Many bioinformatics programming tasks can be automated with ChatGPT. arXiv preprint arXiv:2303.13528 (2023)
12. Surameery, N.M.S., Shakor, M.Y.: Use chat gpt to solve programming bugs. International Journal of Information Technology & Computer Engineering (IJITC) ISSN: 2455-5290 3, 17-22 (2023)

13. Kashefi, A., Mukerji, T.: ChatGPT for programming numerical methods. Journal of Machine Learning for Modeling and Computing 4, (2023)
14. Savelka, J., Agarwal, A., Bogart, C., Sakr, M.: Large language models (GPT) struggle to answer multiple-choice questions about code. arXiv preprint arXiv:2303.08033 (2023)
15. Savelka, J., Agarwal, A., Bogart, C., Song, Y., Sakr, M.: Can Generative Pre-trained Transformers (GPT) Pass Assessments in Higher Education Programming Courses? arXiv preprint arXiv:2303.09325 (2023)
16. Rahman, M.M., Watanobe, Y.: ChatGPT for education and research: Opportunities, threats, and strategies. Applied Sciences 13, 5783 (2023)
17. Oravec, J.A.: Artificial Intelligence Implications for Academic Cheating: Expanding the Dimensions of Responsible Human-AI Collaboration with ChatGPT. Journal of Interactive Learning Research 34, 213-237 (2023)
18. Ouh, E.L., Gan, B.K.S., Shim, K.J., Wlodkowski, S.: ChatGPT, Can You Generate Solutions for my Coding Exercises? An Evaluation on its Effectiveness in an undergraduate Java Programming Course. arXiv preprint arXiv:2305.13680 (2023)
19. Manoharan, S., Speidel, U., Ward, A.E., Ye, X.: Contract Cheating–Dead or Reborn? In: 2023 32nd Annual Conference of the European Association for Education in Electrical and Information Engineering (EAEEIE), pp. 1-5. IEEE, (2023)
20. Maher, M., Tadimalla, Y., Dhamani, D.: Is ChatGPT good for your students? A study design of the impact of AI tools on the student experince in learning Java. In: EDULEARN23 Proceedings, pp. 5702-5709. IATED, (2023)
21. Kendon, T., Wu, L., Aycock, J.: AI-Generated Code Not Considered Harmful. In: Proceedings of the 25th Western Canadian Conference on Computing Education, pp. 1-7. (2023)
22. Bull, C., Kharrufa, A.: Generative AI Assistants in Software Development Education: A vision for integrating Generative AI into educational practice, not instinctively defending against it. IEEE Software (2023)
23. 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. arXiv preprint arXiv:2307.16364 (2023)
24. Braaten, M., Ali, H.M.: Gutter jukser klart mest. Universitas, (2015)
25. Ohmann, P.: An Assessment of Oral Exams in Introductory CS. In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, pp. 613-619. (2019)
26. Akimov, A., Malin, M.: When old becomes new: a case study of oral examination as an online assessment tool. Assessment & Evaluation in Higher Education 1-17 (2020)