# Cheating with AI: Are Large Language Models Undermining Academic Integrity?

Juuso Rytilahti[1], Panu Puhtila[1] and Erkki Kaila[1]

[1]University of Turku, Turku, Finland

**Abstract**

The performance of Large Language Models (LLMs) has evolved significantly in a short time. In relation to this, the pedagogical landscape has also changed, as more and more students adopt these technologies. In this article, we outline how the transforming landscape has affected academic cheating in higher education through the context of programming education. For this, we reflect on 3 different, easily available cheating methods utilizing large language models: generating answers to programming exercises, generating essay answers, and modifying existing resources, such as blog texts or Wikipedia articles, into new answers. For each method, we show simple examples and analyze the quality of the artifacts produced by the models and their usability as real assignment solutions. Moreover, we discuss the difficulties in detecting or preventing the illicit use of AI tools and the potential problems caused by such efforts. Finally, we try to predict how AI will shape the future of education.

**Keywords**

artificial intelligence, academic integrity, cheating, computing education

## 1. Introduction

Large language models (LLMs) have seen rapid performance improvements in the past few years. Since the introduction of ChatGPT-3.5 in late 2022 [1], the phase of change has been breathtaking. The performance of the LLMs has significantly increased, along with a significant increase in the context length (from thousands to 10 million tokens [2]), support for multimodal input (e.g., text, video, sound), the introduction of autonomous agents, reasoning models, and most recently, deep research tools that can synthesize and seek information autonomously. The change in the performance of the models has not happened in a vacuum. The students from lower to higher education have learned to utilize these tools as aids in teaching, but sometimes also with the intent of academic dishonesty.

The new, high-performance tools make academic cheating (in this context, presenting automatically generated content as one's own) easier than ever before. Previously, students needed to ask someone else to complete their assignments or laboriously copy and paste, then transform submissions made by others, which substantially increased the risk of being caught. Current tools eliminate these requirements; a few prompts to an openly available AI model may suffice. This makes academic cheating more accessible and lowers the barrier to entry for academic dishonesty more than ever before. In this paper, we examine different easily accessible cheating methods and analyze the quality of the outcomes they produce. The methods we discuss are as follows:

- Utilizing large language models to automatically solve exercises.
- Shortcut to producing essays: malicious use of synthesized knowledge-gathering tools.
- Cheating through transformation: transforming existing content into new content, which is then presented as one's own.

The methods were chosen for two reasons: first, the produced outcomes represent typical artifacts produced in computer science education, which is the authors' own area of expertise. Second, the cheating methods selected represent typical methods utilized by students (although by no means cover

*8th Conference on Technology Ethics (TETHICS2025), November 11-12, 2025, Vaasa, Finland*

jubery@utu.fi (J. Rytilahti); papuht@utu.fi (P. Puhtila); ertaka@utu.fi (E. Kaila)

https://rytilahti-juuso.github.io/meet-the-researcher/ (J. Rytilahti)

0009-0002-8269-5645 (J. Rytilahti); 0009-0004-6418-1063 (P. Puhtila); 0000-0002-2407-9492 (E. Kaila)

all of them), and are easily available and possible to utilize for all students. In addition to discussing the methods, we also discuss the methods for revealing academic misconduct and the upsides and downsides associated with different methods. While there have been numerous studies about using LLMs in cheating (see Section 2), the ongoing development of LLMs and new models presented means that new studies are constantly needed to reflect the rapidly transforming landscape.

Our paper is structured as follows: In Section 2, we provide an overview of the current state-of-the-art research about our subject. In Section 3, we will present three different scenarios that demonstrate how the LLMs can be used to jeopardize academic integrity, and the results of analyzing these scenarios. In Section 4, we present the mitigation strategies for dealing with academic misconduct. The scenarios, results, and strategies are discussed in Section 5, and finally, the paper is concluded in Section 6.

## 2. Related Work

### 2.1. Performance of LLMs

Large language models have multiple caveats. The most notable are the models' tendency to hallucinate and reliance on shortcuts. Hallucinations are a phenomenon where a model's output is fluent and semantically correct, but the message itself is incorrect [3]. Shortcut refers to a larger phenomenon, where the output relies on spurious correlations, for example, the used vocabulary in the answers [4, 5]. It encompasses a larger subset of definitions using spurious correlations; for example, position bias [6] can be perceived as a subset of the shortcut phenomena.

Many studies have tried to find out the reasoning capabilities of LLMs. A study conducted by Jiang et al. [7] tested LLMs' ability to reason in multiple-choice questions and found that models still can't formally reason. Mirzadeh et al. [8] tested LLMs ability in mathematical reasoning, and found that the performance of the models decreases when numbers in multiple-choice math-related questions are altered, or if there have been added additional clauses are added that seem relevant to the question.

The models are already quite capable of producing answers to short-form essays on general topics, see for example Yeadon et al. [9], where answers created with davinci-003 and ChatGPT-3.5 for short-form (300-word) essays in a university physics module were reviewed with quite high scores of around 70% of the max score. Additionally, their performance has, at least for now, been on the rise. According to Yamada et al. [10], a paper written fully by AI passed peer review in the ICLR 2025 Workshop titled "I Can't Believe It's Not Better: Challenges in Applied Deep Learning"[1].

Susnjak et al. [11] studied the use of ChatGPT in cheating in online exams of higher education institutions. In the experiments they conducted with GPT-4 they concluded that it can be used to cheat in university exams very well, although not perfectly. Savelka et al. [12] noted in their study that GPT-4 was able to pass three Python-related courses, covering from essentials to object-oriented programming to a more practical course.

### 2.2. Techonological Advancements

The phase in which new models are introduced has been rapid. Models are published by different entities, e.g. OpenAI, Meta, Alibaba, DeepSeek, and Anthropic. Moreover, often an entity does not publish only one model, but multiple, different models. And of course, there are multiple fine-tuned variants of the open-sourced models available. To highlight the fast phase in which the new models are introduced, one could mention a study conducted by Zhang et al. [13], where they created a taxonomy for multi-modal LLMs that encompassed 126 different LLMS published between 2022-2024.

Furthermore, Wang et al. [14] presented 20 different autonomous agents from papers that were published from January 2021 to August 2023. These included general agents, such as Auto-GPT[2] and autonomous agents built for more specific purposes, for example, ChatDev by Qian et al. [15], where LLMs build software independently. Autonomous agents are also being integrated into existing tooling,

[1] https://sites.google.com/view/icbinb-2025/home
[2] https://github.com/Significant-Gravitas/AutoGPT

for example, Visual Studio Code recently made autonomous agent integration available for all users [16].

Additionally, there is also a surge of different generative AI leveraging tools. One example is NotebookLM[3], which is an interactive environment by Alphabet. It allows interaction from multiple sources of input. It can also convert the used sources into an AI-generated discussion or a podcast. Other, rather relevant example could be Elicit [17], tool that can, according to their website, "*analyze research papers at superhuman speed*" [18].

## 2.3. Automated Detection of AI-generated Content

There are multiple ways to try to detect AI-generated text. One of the most recent works by Reinhart et al. [19] attempted to recognize text generated by multiple different variants of a set of lexical, Llama 3, and GPT-4o grammatical, and rhetorical features. However, their method also identified 9.8% of human texts as AI-generated. Their corpus consisted of six different categories of text. Then there are also retrieval-based detectors [20] and watermark-based detectors. According to Kirchenbauer et al., "*A watermark is a hidden pattern in text that is imperceptible to humans, while making the text algorithmically identifiable as synthetic.*" [21]. There are multiple works that utilize watermarks with differing techniques, and they have achieved quite good performance [22, 23, 24]. A more concrete example could be Kirchenbauer et al. [21], who created a method for creating "soft watermarks", a method that makes it easier to create and utilize watermarks via patterns.

Salim et al. [25] investigated techniques to deter the use of LLMs in cheating, and came to the conclusion that adversarial perturbations, when correctly applied, could reduce the correctness of LLM-generated materials by 77% at best. According to a study made by Sadasivan et al. [26], recursive paraphrasing reduces the watermarking detection of LLMs quite significantly. They also note in their study that a teacher could utilize a paraphrasing attack on a student's work to get it falsely marked as AI-generated content on Retrieval-based detectors. Hoq et al. [27] achieved high accuracy in detecting AI-generated submissions from human-made submissions on CS1-level simple programming exercises that were made with Java, achieving over 90% accuracy.

Wang and Li [28] developed a system to detect LLM-generated content, which was based on the idea that such a system must be capable of detecting also LLMs that have yet to emerge. Their experimental tests concluded that the system reached 97.04% LLM-detection accuracy when examining English language materials, and 91.23% when examining Chinese materials. Niu et al. [29] inspected the problem of how to detect manually copy-typed LLM-generated answers in proficiency tests, and developed a system that achieved 100% accuracy on most categories, and nearly perfect accuracy on all, beating the previous similar systems by a wide margin.

## 2.4. Impact on Learning From Students' Perspective

Yilmaz et al. [30] studied how students perceived ChatGPT use on an object-oriented programming course (N=41), and noted that students perceived advantages of ChatGPT providing correct answers, debugging help, and increased self-confidence, and the downsides being facilitating laziness and possible incorrect answers provided by the LLM. Abbas et al. [31] found that the use of ChatGPT can reduce the academic performance of a student. Lehmann et al. [32] found that how students use LLMs depends on how it affects their learning. They note that using AI to solve exercises can increase the amount of learned topics, but reduce the depth of understanding. If AI is a complementary tool, it can aid in learning in-depth knowledge by providing additional explanations. They note that LLMs seem to support students with good prior knowledge, but hamper the learning of students with less prior knowledge. Bastani et al. [33] studied the use of GPT-4 in the context of math classes at a high school and noted that generative AI can harm learning, although their study setting has received critique [34]. Jošt et al. [35] studied the effect of LLM usage in software development and found a significant negative connection with reliance on LLMs on critical thinking tasks and lower final grades. They

[3]https://notebooklm.google/

**Table 1**
Points collected by solving the programming exercises in two modes. The total maximum points available from programming exercises was 1470.

| Mode | Points | % |
|---|---|---|
| Beginner | 1072 | 73.0% |
| Programming Student | 1114 | 75.6% |

also note that if LLM is used only as a supplementary learning tool, for example, to give additional explanations, the negative correlation with lower grades was not as strong. Zirar et al. [36] note in their study, inspecting 25 different papers, that reliance on LLMs without critical thinking negatively affects student performance. They also note that LLM use should be restricted to a specific, defined role.

LLMs can also provide a positive impact on students' learning. With careful consideration, they can be used as 24/7 accessible tutors [37, 38], or as enablers for more adaptive learning approaches, for example, to create new programming exercises or to modify existing exercises [39, 40].

## 3. Practical Examples

In this section, we will discuss three distinct cheating methods that utilize freely available AI tools. These examples demonstrate the ease with which these technologies can be used for malicious purposes. The first method involves automatically generating exercise answers. The second method involves synthesizing answers. The third method involves appropriating and transforming existing materials or answers into new ones for more complex submissions.

### 3.1. LLM as an Automatic Solver of Exercises

First, we will discuss how to use a large language model or other AI tool to solve an exercise. This involves providing the tool with the exercise description or task and then copying and pasting the answer as one's own. To illustrate this process, we will observe the solution to a task from an introductory programming course. This course is a typical CS1 course [41], taught using Python as the programming language. This eight-week course requires no prior programming knowledge and is mandatory for all computing and computer science majors, as well as many other students across different departments. Around 500 to 600 students typically take the course annually.

Since the course is quite large, the weekly exercises are automatically assessed. The course consists of seven weekly modules, each containing a combination of learning material and exercises. Each module contains 20 to 40 individual exercises. The exercises work in a standard web browser, can be compiled and executed with a single click, and provide immediate feedback upon submission. Students can try each exercise as many times as needed within the time frame (each module is typically open for nine days).

An example of an exercise is displayed in Figure 1.

To test the LLM's capabilities in solving the exercises, we first tried to solve all exercises in the course in two modes: in the *beginner* mode, we provided the task to ChatGPT and copied and pasted the solution back as-is. In the *programming student* mode, we again provided the task to ChatGPT, but before pasting the solution back to the assessment tool, obvious errors (such as duplicate variable names) in the solution were fixed. Hence, the latter mode simulated a programming student with some experience with programming, while the first one simulated someone who had practically no programming experience. The results are displayed in Table 1.

As we can see, the points are a little higher when simple errors are accounted for (the programming student mode), but the difference is quite small. To further see how well AI can potentially solve the exercises, we tried to complete them all in an iterative mode. In this mode, each exercise was similarly given to the AI model to be solved. However, if the solution was not accepted by the assessment tool,

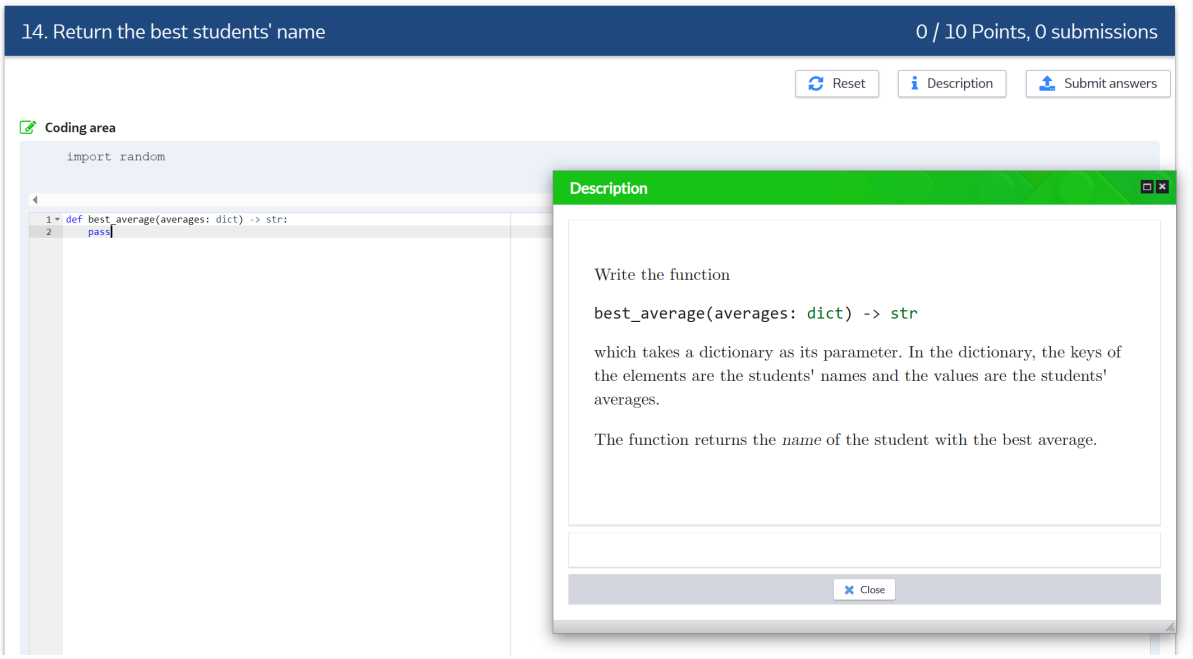



**Figure 1:** Example of an exercise in the course. In the exercise, the student is supposed to write the required program code to complete the given task. The code can be tested and modified as many times as needed.

**Table 2**
Points collected by solving the programming exercises in iterative mode with two models of ChatGPT. The total maximum points available from programming exercises was 1470.

| Model | Points | % |
|---|---|---|
| GPT-3.5 | 1303 | 88.6% |
| GPT-4 | 1457 | 99.1% |

the error message or other feedback was given back to the AI model to improve the solution. We tested this approach with two different models of ChatGPT: 3.5 and 4. With the earlier model, the maximum number of iterations was 3. With the latter, we kept on trying as long as their answer kept improving. The results for the iterative mode are displayed in Table 2.

As shown in the table, version 4 of the model nearly achieved a perfect score. The errors made by the AI models were quite trivial; most of the time, missing points were due to incorrectly formatted output. In some cases, the exercises restricted the use of external modules without explicitly stating so in the description. This resulted in code that produced correct output but was not accepted by the assessment tool. Complete results and a full description of the process are available on Zenodo [42].

## 3.2. Shortcut to Producing Essays

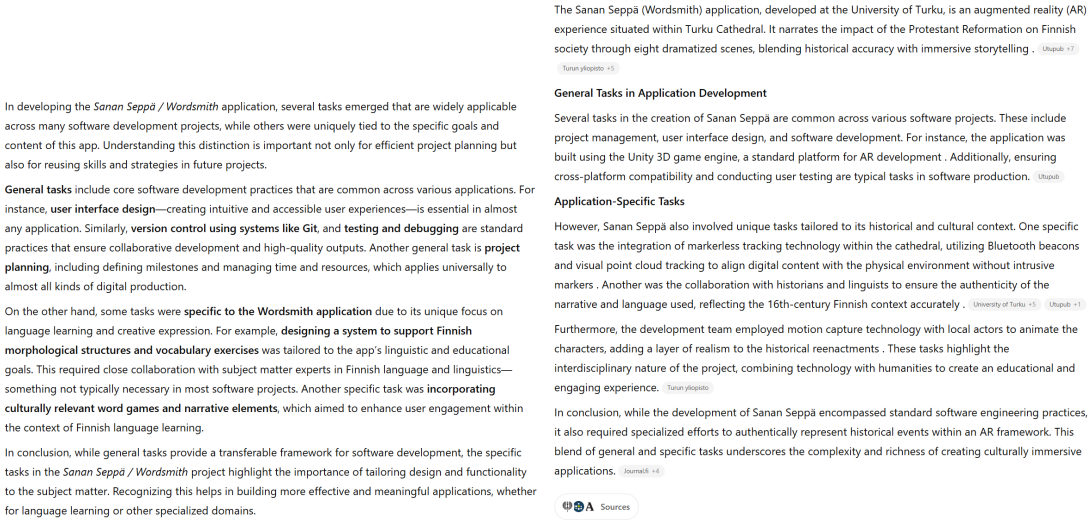
As mentioned earlier, LLMs can generate text responses that vary in difficulty and length. However, autonomous agents can synthesize knowledge from various sources. Since there are now multimodal models, their prompts can include lecture videos, academic papers, and other study materials that might not be present in even the state-of-the-art models' training data. This originally renders LLMs unable to answer the question. Furthermore, we note that these tools can blur the line between academic dishonesty and honest intentions.

The example is an essay prompt from an introductory course on AR and VR development. The target audience for the course is master's level students. The prompt contained a role, a length limit, and the essay question. To make the model search the internet as well, the instructions included an additional clause to search the web. The prompt used was:

"You are a university master's student studying computer science. Create a 300-word essay about the topic below. Search the web to find the [Wordsmith] application-related works that were created at the University of Turku.

Considering all the tasks in making the Wordsmith application, which tasks do you think are general (also needed in many other productions) and which are specific to this application and subject? Pick a few examples from each group."

Two versions of the answer were produced, with and without additional prompting. The results can be seen in the Figure 2. Notably, the LLMs lacked knowledge of the AR application "Wordsmith" produced at the University of Turku as part of a project, resulting in a low-quality hallucinated answer. However, due to rather recent advancements in the functioning of GPT-4o, the model was able to synthesize an answer from multiple different online sources. The answer is not perfect, but its quality improved considerably when compared against the original answer produced by the LLM.



In developing the *Sanan Seppä / Wordsmith* application, several tasks emerged that are widely applicable across many software development projects, while others were uniquely tied to the specific goals and content of this app. Understanding this distinction is important not only for efficient project planning but also for reusing skills and strategies in future projects.

**General tasks** include core software development practices that are common across various applications. For instance, **user interface design**—creating intuitive and accessible user experiences—is essential in almost any application. Similarly, **version control using systems like Git**, and **testing and debugging** are standard practices that ensure collaborative development and high-quality outputs. Another general task is **project planning**, including defining milestones and managing time and resources, which applies universally to almost all kinds of digital production.

On the other hand, some tasks were **specific to the Wordsmith application** due to its unique focus on language learning and creative expression. For example, **designing a system to support Finnish morphological structures and vocabulary exercises** was tailored to the app's linguistic and educational goals. This required close collaboration with subject matter experts in Finnish language and linguistics—something not typically necessary in most software projects. Another specific task was **incorporating culturally relevant word games and narrative elements**, which aimed to enhance user engagement within the context of Finnish language learning.

In conclusion, while general tasks provide a transferable framework for software development, the specific tasks in the *Sanan Seppä / Wordsmith* project highlight the importance of tailoring design and functionality to the subject matter. Recognizing this helps in building more effective and meaningful applications, whether for language learning or other specialized domains.

The Sanan Seppä (Wordsmith) application, developed at the University of Turku, is an augmented reality (AR) experience situated within Turku Cathedral. It narrates the impact of the Protestant Reformation on Finnish society through eight dramatized scenes, blending historical accuracy with immersive storytelling . Utupub +7 Turun yliopisto +5

**General Tasks in Application Development**

Several tasks in the creation of Sanan Seppä are common across various software projects. These include project management, user interface design, and software development. For instance, the application was built using the Unity 3D game engine, a standard platform for AR development . Additionally, ensuring cross-platform compatibility and conducting user testing are typical tasks in software production. Utupub

**Application-Specific Tasks**

However, Sanan Seppä also involved unique tasks tailored to its historical and cultural context. One specific task was the integration of markerless tracking technology within the cathedral, utilizing Bluetooth beacons and visual point cloud tracking to align digital content with the physical environment without intrusive markers . Another was the collaboration with historians and linguists to ensure the authenticity of the narrative and language used, reflecting the 16th-century Finnish context accurately . University of Turku +5 Utupub +1

Furthermore, the development team employed motion capture technology with local actors to animate the characters, adding a layer of realism to the historical reenactments . These tasks highlight the interdisciplinary nature of the project, combining technology with humanities to create an educational and engaging experience. Turun yliopisto

In conclusion, while the development of Sanan Seppä encompassed standard software engineering practices, it also required specialized efforts to authentically represent historical events within an AR framework. This blend of general and specific tasks underscores the complexity and richness of creating culturally immersive applications. JournalS +4

Sources

**Figure 2:** Generated essay answers with OpenAI's GPT-4o. On the left the the answer produced by the model without internet browsing instruction, and on the right with internet browsing. The sources for each paragraph are clearly marked by the tool.

## 3.3. Cheating by Transformation

Cheating by transformation involves using an LLM to create a new version of existing content. Examples of this include borrowing an answer from a friend or copying content from sources such as GitHub or Wikipedia, modifying it, and presenting it as one's own. Previously, creating a new version of a work required extra effort. Now, students can use AI tools to transform content into another form and pass it off as their own.

To illustrate this, we selected three original sources to be transformed.

- Program code, written in Python, for a simple library database application. Similar exercises can usually be found at the end of introductory courses or in CS2 courses.
- A blog text about screen time. As the blog is "scientific", including some references to articles, it can be considered quite similar content-wise to essays produced by students.
- An excerpt of a Wikipedia article about plagiarism.

All the original sources were transformed by using two freely available AI models, GPT-4o[4] and Mistral[5], an open-source AI tool. To check the similarity to the original source, we used two openly available web tools:

[4]https://openai.com/index/hello-gpt-4o/
[5]https://mistral.ai/

**Table 3**
The similarity of transformed program code to original code.

| AI Model | Similarity (TCT) | Similarity (TSC) |
|---|---|---|
| GPT-4o | 68% | 51% |
| Mistral | 93% | 87% |

**Table 4**
The similarity of transformed blog text to original text.

| AI Model | Similarity (TCT) | Similarity (TSC) |
|---|---|---|
| GPT-4o | 81% | 49% |
| Mistral | 79% | 62% |

**Table 5**
The similarity of the transformed article to the original text from the Wikipedia article.

| AI Model | Similarity (TCT) | Similarity (TSC) |
|---|---|---|
| GPT-4o | 77% | 51% |
| Mistral | 78% | 64% |

- Text Compare Tool[6] (TCT) by GoTranscript.
- Text Similarity Checker[7] (TSC) by Search/natural.

For the program code, the tools were given the original code accompanied by a prompt: "Change as much as possible of the code below while keeping the function names, output, and functionality intact". The functions and the output needed to be preserved for the assessment tool to accept the solution. The results of the comparison are displayed in Table 3.

Both models preserved the necessary parts quite well, but it seems that GPT-4o provided more changes in the code. When taking a closer look, a lot of differences by GPT-4o were done by changing variable names and formatting where possible, while the changes made by Mistral were more about using different programming statements and constructs.

For the blog text, we used the following prompt: "Rewrite the text of this essay while preserving the idea and keeping the links intact." The results are displayed in Table 4.

Again, both models preserved the original idea very well. The changes were made to the wording of sentences, such as "Recently, Finland introduced new legislation allowing schools to ban mobile phones during the school day" becoming "Finland has recently enacted a law permitting schools to restrict mobile phone use during school hours." There does not seem to be a huge performance difference between the models, although the distance calculated to the original by TSC seems to be greater for GPT-4o-produced text.

Finally, for the Wikipedia article excerpt, we used a prompt similar to blog text earlier: "Rewrite the text of this article while preserving the idea and keeping the links intact." The results are displayed in Table 5.

Again, the transformed article seems to preserve the idea of the original article quite well, this time mainly substituting individual terms with their synonyms. For example, the original text " As such, a person or entity that is determined to have committed plagiarism is often subject to various punishments or sanctions..." becomes "As a result, individuals or entities found guilty of plagiarism often face various punishments or sanctions..."

[6]https://gotranscript.com/text-compare
[7]https://searchnatural.co.uk/text-similarity-checker

# 4. Mitigation Strategies for Academic Dishonesty

In this section, we discuss the possible, commonly suggested strategies for identifying and preventing academic misconduct.

## 4.1. Automatic Detection of AI-Generated Text

In some instances, tools that automatically detect LLM-produced text might be useful. However, problems arise from false positives, which could threaten students' rights. Proving that something was not done by AI is at least as difficult as proving that it was. This issue has even made headlines before. In Finland, for example, teachers used an AI detector tool that allegedly produced a false positive [43].

As mentioned in the news article, the edit history of the document could be used to determine if an answer is AI-generated. While looking at the edit history is certainly useful and can help to detect clear misconduct, it is not a foolproof method, be it for code or a Word file. For a while, there have been programs that aim to simulate typing, even simulating typing mistakes, including, for example, Duey.ai's[8] Chrome extension for writing Google Docs. It is likely that such tools will get better over time.

Additionally, many AI detectors seem to assume that students use only one model. What's to stop more capable students from writing a script that generates each answer sentence from a randomly selected AI model? This is especially possible now that various models are available for anyone to use. Furthermore, one could argue that the vast number of models poses a challenge in itself because creating an automatic detector of AI-generated text that encompasses most models is perceived as a challenging, resource-intensive task. However, as noted in Section 2.3, there are efforts within the academic community to achieve this exact goal.

## 4.2. Oral Verification

Asking students to explain their code, answer, and the steps they took to prepare it can be an effective approach. However, there are still potential problems, especially in remote courses. These range from apparent issues, such as verifying that the student enrolled in the course is indeed the one taking the exam, to more imaginary issues. Some available tools can evade detection in screen-sharing applications, such as InterviewCoder [9]. Although the tool's main purpose is to help students cheat on job interviews for positions requiring programming skills, it can easily be repurposed to help students pass oral course verifications. Oral interviews can also be problematic for honest students who fear social situations, and they can produce false positives in such cases.

## 4.3. Making the Exercise Harder to Solve for AI

There are numerous methods to try to make an exercise harder for AI to solve. One quite popular method is to alter the question setting, making exercises, such as essays, more complex. Another possible solution would be to use only recently released research papers as an additional context, so the answer is not present in the model's training data. However, these can be perceived as not feasible approaches, for two reasons: The state-of-the-art models are quite performant, and can even write complete papers, as mentioned in 2.1. Additionally, the multimodal models can accept input in various formats, making the strategy of scattering the information needed to complete the exercise into multiple sources insufficient. Lastly, new models (and the different autonomous agents utilizing them) can browse the internet independently, finding the most recent information on any topic.

Another method could be to try to require the answers to the exercises to be created so that either exercise descriptions or the correct answers have to contain visual outputs, such as graphs or tables. However, even an old GPT-4-based model could answer questions requiring a visual output answer, as

[8] https://www.duey.ai/duey-ai-auto-typer-for-google-docs
[9] https://www.interviewcoder.co/

can be seen in the Appendix B. Furthermore, OpenAI's newest models, o3 and o4-mini, seem to perform even better with prompts containing elements that require visual interpretation [44]. Additionally, the LLM integration in different tools has become more commonplace, further disqualifying this method as a valid approach.

A more fruitful approach could be to try to add additional clauses to the exercise descriptions, leading LLMs' reasoning astray. This effect can be seen in MCQ benchmarks as degrading the performance of the state-of-the-art LLMs significantly [8]. However, this could also make the exercise harder for students to understand. Furthermore, this could emphasize the challenges faced with false positives, as humans could be led astray from the original intent exercise as well due to such additional clauses.

## 5. Discussion

When it comes to coding, AI models seem quite capable of solving individual exercises at the moment. However, to achieve the best results, those using the models should have at least a little understanding of basic programming concepts and terminology. Copying and pasting the exercise and producing programming code did not result in maximum performance, but regardless awarded 73% of available points, which would easily exceed the maximum requirements in the course. With a little programming knowledge and a little determination, it is possible to achieve almost the maximum score.

The transformation of program code, blog text, or a Wikipedia article seems to preserve the original meaning or functionality quite well. The new versions may appear different to automated similarity checkers, but for human observer, recognizing them as copies is not too difficult (at least for an essay or an article): the structure is very similar to the original, and the alteration is done one sentence at a time by replacing words or short subsentences with synonyms, which makes recognition quite easy for human. For the program code, it may not be as easy as programs of the size observed are fundamentally already quite similar, and the changes made in variables and in used programming techniques may fool even more experienced teachers, too.

Cheating is easier to detect because cheating students often seem lazy, leading them to use rudimentary, easily detectable methods. Examples of such behavior include directly copying and pasting from a popular model's output or copying someone else's submission. One could argue that these are the most common methods of cheating. However, as students become more accustomed to AI tools and as tools that enable academic dishonesty become more common and mainstream, this may change.

Savelka et al. [12] noted in their study that future work should focus on developing innovative assessments resilient to automatically generated solutions, e.g., by requiring human interaction. Adding such resilience is particularly challenging in MOOC courses and might hamper the biggest strengths of such courses, flexibility and scalability, and can be seen as a difficult task. To get around this, one could argue that a more adaptive approach could be, for example, flipping the exercise so that instead of producing only an answer, students would create similar exercise variants (description and answer) using a different context. However, a student can produce a mostly working variant quite easily with modern AI tools: Rytilahti et al. [45] noted that AI is quite capable of producing these kinds of exercise variants.

Of course, the advancements made in the field of LLMs and other forms of generative AI are not all bad. Although recent advancements have been made in the field of generative AI, and especially the ever-more common integration of generative AI into different workflows, Model Context Protocol (MCP) [46] as a recent example, can both threaten and be useful for the learning. MCP creates opportunities for integrating adaptive learning approaches into teaching. Allowing a model in Visual Studio Code to provide additional context, such as study materials of a particular course, directly in the code editor could be beneficial.

There are several ethical considerations related to the use of AI. First, if a teacher rewrites course tasks so that they are difficult to solve by LLMs, they are likely also difficult for humans to solve, which is unfair to students who are not cheating. Second, the distinction between AI- and human-produced material should be clearly defined, though this can be difficult, even when the permitted use of AI is

clearly announced within the course context. Having different rules for AI usage in different courses (or even in different assignments within the same course) can lead to some students not using AI, even when it is permitted, which puts them at a disadvantage or may increase their workload beyond the intended level. In literature, fairness seems to be the most often noted ethical problem as well [47].

Students' AI-related skills vary greatly [48]. This means that students accustomed to using LLMs may have an unfair advantage when it comes to completing the degree. For this reason, teachers should teach students to productively use such tools in the course context; however, not all teachers can do this themselves. In the worst-case scenario, the inability to detect or prevent AI usage may eventually lead to entire degrees losing their value. The use of AI tools can also increase the number of students suffering from impostor syndrome [49], as students may feel unable to solve problems without AI tools.

## 5.1. Impacts to Learning

Needless to say, using LLMs to cheat in academic studies negatively impacts what students learn and how prepared they are for careers in technology after graduation. In the worst-case scenario, we may see a generation of computer science majors who have never written a line of code, an essay, or a diagram by the time they graduate. These students would lack the skills related to these activities, as well as two crucial abilities necessary for working in the tech industry. The first is the ability to conceptualize things and perceive problems on an abstract level. The second is the ability to search for relevant information, process it, and synthesize it into new insights.

These are skills that cannot be learned without practice. Overreliance on LLMs avoids that practice altogether. If you think that's bad, consider that it's just the earthquake that precedes the real tsunami coming our way. After all, the current generation of college students grew up in an era before LLMs were widely available. However, future generations will consist of people who have grown accustomed to using LLMs from the moment they learned to write — or possibly even before, as voice-directed AI assistants may become more common.

This is not to say that LLMs would not have their place in computing education or software development. In the hands of someone who has acquired the fundamental skills needed to work as a developer, LLMs are powerful instruments that can increase productivity significantly. The problem we are discussing here is highly situational. For someone with no development experience, LLMs offer an easy way out, eliminating the need to learn or adapt.

The best possible solution to the problem is probably to motivate students to understand why they need to learn to do things on their own instead of using AI. This logic is similar to learning basic math before using a calculator, but it may not be as easy in reality. The problem is that it's difficult to predict how work will change in the future because of AI tools. Therefore, motivating students can be challenging because we are not sure what kind of future we are preparing them for.

## 5.2. Limitations and Ethical Considerations

This study has some limitations. The methods for misconduct are provided as examples, and as such, no systematic analysis of different tools, methods, models, or prompt engineering is conducted. However, we note that academia has inspected the performance of the LLMs in various studies in different use cases and finds them to be quite performant. Thus, there is no need to validate the methods presented with a large data set, but only outline them as examples to tie the discussion into a relevant context. Still, the current study is based on a limited number of models and case studies to demonstrate the possibilities, and does not provide a holistic view of AI-enabled cheating.

One could raise a point about the ethical dilemma of introducing tools and methods for academics to cheat. However, these tools are readily available, and the methods presented are rather straightforward. This means that the students are already likely utilizing similar methods, meaning the academic discussion also needs more context about the effect these tools bring to the landscape of education. Additionally, the advancements in the field of generative AI have been rapid, meaning that informing teachers on the possibilities the openly available models provide is highly important.

## 6. Conclusion and Future Work

In this paper, we discussed how LLMs can be used to cheat on academic assignments and how this jeopardizes the learning potential of future graduates on a larger scale. While the quality of content produced by AI models varies in different contexts, it is generally good enough to pass assignments or entire courses. Furthermore, as the models improve, so does the quality of the content they produce. Since current methods for detecting academic misconduct are insufficient, motivating students to learn may be the only solution.

Future work should examine how different AI detectors handle frequent changes to the models used to produce answers. Additionally, more research should focus not just on the raw performance of the models or hypothetical attack vectors, but also on simulating cheaters. This would provide a better understanding of the ways in which the current generation of students can engage in academic dishonesty.

## Acknowledgments

This work has been supported by FAST, the Finnish Software Engineering Doctoral Research Network, funded by the Ministry of Education and Culture, Finland.

## Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT, Grammarly, and DeepL in order to: Citation management, Content enhancement, Grammar and spelling check, Improve writing style, Paraphrase and reword, and Text translation. After using these tools/services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

## References

[1] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, 2020. arXiv:2005.14165.

[2] M. AI, The llama 4 herd: The beginning of a new era of natively multimodal ai innovation, https://ai.meta.com/blog/llama-4-multimodal-intelligence/, 2025. Accessed: 2025-04-09.

[3] Z. Xu, S. Jain, M. Kankanhalli, Hallucination is inevitable: An innate limitation of large language models, arXiv preprint arXiv:2401.11817 (2024).

[4] Y. Zhou, R. Tang, Z. Yao, Z. Zhu, Navigating the shortcut maze: A comprehensive analysis of shortcut learning in text classification by language models, arXiv preprint arXiv:2409.17455 (2024).

[5] R. Song, Y. Li, L. Shi, F. Giunchiglia, H. Xu, Shortcut learning in in-context learning: A survey, arXiv preprint arXiv:2411.02018 (2024).

[6] P. Pezeshkpour, E. Hruschka, Large language models sensitivity to the order of options in multiple-choice questions, arXiv preprint arXiv:2308.11483 (2023).

[7] B. Jiang, Y. Xie, Z. Hao, X. Wang, T. Mallick, W. J. Su, C. J. Taylor, D. Roth, A peek into token bias: Large language models are not yet genuine reasoners, arXiv preprint arXiv:2406.11050 (2024).

[8] I. Mirzadeh, K. Alizadeh, H. Shahrokhi, O. Tuzel, S. Bengio, M. Farajtabar, Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models, arXiv preprint arXiv:2410.05229 (2024).

[9] W. Yeadon, O.-O. Inyang, A. Mizouri, A. Peach, C. P. Testrow, The death of the short-form physics essay in the coming ai revolution, Physics Education 58 (2023) 035027.

[10] Y. Yamada, R. T. Lange, C. Lu, S. Hu, J. Foerster, J. Clune, D. Ha, The AI scientist-v2: Workshop-level automated scientific discovery via agentic tree search, Sakana AI Blog Post. Available online: https://pub.sakana.ai/ai-scientist-v2/paper, 2025.
[11] T. Susnjak, T. R. McIntosh, Chatgpt: The end of online exam integrity?, Education Sciences 14 (2024) 656.
[12] J. Savelka, A. Agarwal, M. An, C. Bogart, M. Sakr, Thrilled by your progress! large language models (gpt-4) no longer struggle to pass assessments in higher education programming courses, in: Proceedings of the 2023 ACM Conference on International Computing Education Research-Volume 1, 2023, pp. 78–92.
[13] D. Zhang, Y. Yu, J. Dong, C. Li, D. Su, C. Chu, D. Yu, Mm-llms: Recent advances in multimodal large language models, arXiv preprint arXiv:2401.13601 (2024).
[14] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, et al., A survey on large language model based autonomous agents, Frontiers of Computer Science 18 (2024) 186345.
[15] C. Qian, W. Liu, H. Liu, N. Chen, Y. Dang, J. Li, C. Yang, W. Chen, Y. Su, X. Cong, et al., Chatdev: Communicative agents for software development, arXiv preprint arXiv:2307.07924 (2023).
[16] I. Nikolic, Agent mode: available to all users and supports mcp, 2025. URL: https://code.visualstudio.com/blogs/2025/04/07/agentMode, accessed: 2025-04-16.
[17] Elicit, Elicit: The ai research assistant, 2023. URL: https://elicit.com.
[18] Elicit, Analyze research papers at superhuman speed, 2025. URL: https://elicit.com, accessed: 2025-04-23.
[19] A. Reinhart, B. Markey, M. Laudenbach, K. Pantusen, R. Yurko, G. Weinberg, D. W. Brown, Do llms write like humans? variation in grammatical and rhetorical styles, Proceedings of the National Academy of Sciences 122 (2025) e2422455122.
[20] K. Krishna, Y. Song, M. Karpinska, J. Wieting, M. Iyyer, Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense, Advances in Neural Information Processing Systems 36 (2023) 27469–27500.
[21] J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, T. Goldstein, A watermark for large language models, in: A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, J. Scarlett (Eds.), Proceedings of the 40th International Conference on Machine Learning, volume 202 of *Proceedings of Machine Learning Research*, PMLR, 2023, pp. 17061–17084. URL: https://proceedings.mlr.press/v202/kirchenbauer23a.html.
[22] E. Giboulot, T. Furon, Watermax: breaking the llm watermark detectability-robustness-quality trade-off, arXiv preprint arXiv:2403.04808 (2024).
[23] Z. Xu, K. Zhang, V. S. Sheng, Freqmark: Frequency-based watermark for sentence-level detection of llm-generated text, arXiv preprint arXiv:2410.10876 (2024).
[24] X. Xu, Y. Yao, Y. Liu, Learning to watermark llm-generated text via reinforcement learning, arXiv preprint arXiv:2403.10553 (2024).
[25] S. I. Salim, R. Y. Yang, A. Cooper, S. Ray, S. Debray, S. Rahaman, Impeding llm-assisted cheating in introductory programming assignments via adversarial perturbation, arXiv preprint arXiv:2410.09318 (2024).
[26] V. S. Sadasivan, A. Kumar, S. Balasubramanian, W. Wang, S. Feizi, Can ai-generated text be reliably detected?, arXiv preprint arXiv:2303.11156 (2023).
[27] M. Hoq, Y. Shi, J. Leinonen, D. Babalola, C. Lynch, T. Price, B. Akram, Detecting chatgpt-generated code submissions in a cs1 course using machine learning models, in: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1, 2024, pp. 526–532.
[28] Q. Wang, H. Li, On continually tracing origins of llm-generated text and its application in detecting cheating in student coursework, Big Data and Cognitive Computing 9 (2025) 50.
[29] C. Niu, K. Yancey, R. Liu, M. Baig, A. Horie, J. Sharpnack, Detecting llm-assisted cheating on open-ended writing tasks on language proficiency tests, in: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track, 2024, pp. 940–953.
[30] R. Yilmaz, F. G. K. Yilmaz, Augmented intelligence in programming learning: Examining student

views on the use of chatgpt for programming learning, Computers in Human Behavior: Artificial Humans 1 (2023) 100005.
[31] M. Abbas, F. A. Jam, T. I. Khan, Is it harmful or helpful? examining the causes and consequences of generative ai usage among university students, International Journal of Educational Technology in Higher Education 21 (2024) 10.
[32] M. Lehmann, P. B. Cornelius, F. J. Sting, Ai meets the classroom: When does chatgpt harm learning?, Available at SSRN 4941259 (2024).
[33] H. Bastani, O. Bastani, A. Sungu, H. Ge, O. Kabakcı, R. Mariman, Generative ai can harm learning, Available at SSRN 4895486 (2024).
[34] S. Tan, V. Rajaratnam, Critique of generative ai can harm learning study design, Available at SSRN 4898213 (2024).
[35] G. Jošt, V. Taneski, S. Karakatič, The Impact of Large Language Models on Programming Education and Student Learning Outcomes, Applied Sciences 14 (2024) 4115–4115. doi:10.3390/app14104115.
[36] A. Zirar, Exploring the impact of language models, such as ChatGPT, on student learning and assessment, Review of Education 11 (2023). doi:10.1002/rev3.3433.
[37] H. Huo, X. Ding, Z. Guo, S. Shen, D. Ye, O. Pham, D. Milne, L. Mathieson, A. Gardner, Accelerating learning with ai: Improving students' capability to receive and build automated feedback for programming courses, in: 2024 World Engineering Education Forum-Global Engineering Deans Council (WEEF-GEDC), IEEE, 2024, pp. 1–9.
[38] A. Al-Abri, Exploring chatgpt as a virtual tutor: A multi-dimensional analysis of large language models in academic support, Education and Information Technologies (2025) 1–36.
[39] N. B. D. Ta, H. G. P. Nguyen, S. Gottipati, Exgen: Ready-to-use exercise generation in introductory programming courses, in: International Conference on Computers in Education, 2023.
[40] E. Logacheva, A. Hellas, J. Prather, S. Sarsa, J. Leinonen, Evaluating contextually personalized programming exercises created with generative ai, in: Proceedings of the 2024 ACM Conference on International Computing Education Research-Volume 1, 2024, pp. 95–113.
[41] M. Hertz, What do" cs1" and" cs2" mean? investigating differences in the early courses, in: Proceedings of the 41st ACM technical symposium on Computer science education, 2010, pp. 199–203.
[42] J. Rytilahti, E. Kaila, How easy is it to cheat? solving programming exercises automatically with ai, 2024. URL: https://doi.org/10.5281/zenodo.17177696. doi:10.5281/zenodo.17177696, presentation in the "Projects in Progress" track, 20th International CDIO Conference, Ecole Supérieure Privée d'Ingénierie et de Technologies (ESPRIT), Tunis, Tunisia, June 10–13, 2024.
[43] H. Sanomat, Opiskelijan kurssityö tuomittiin vilpiksi – adriann nyman latasi pöytään todisteet, 2025. URL: https://www.hs.fi/suomi/art-2000011119871.html, accessed: 2025-04-15.
[44] OpenAI, Thinking with images, https://openai.com/index/thinking-with-images/, 2025. Accessed: 2025-04-24.
[45] J. Rytilahti, O. Weerakoon, E. Kaila, Exploring ai-driven programming exercise generation, in: Proceedings of the 20th International CDIO Conference, hosted by Ecole Supérieure Privée d'Ingénierie et de Technologies (ESPRIT) Tunis, Tunisia, June 10 – June 13, 2024, 2024.
[46] Anthropic, Introducing the model context protocol, 2024. URL: https://www.anthropic.com/news/model-context-protocol, accessed: 2025-04-15.
[47] B. Memarian, T. Doleck, Fairness, accountability, transparency, and ethics (fate) in artificial intelligence (ai) and higher education: A systematic review, Computers and Education: Artificial Intelligence 5 (2023) 100152.
[48] A. Grájeda, J. Burgos, P. Córdova, A. Sanjinés, Assessing student-perceived impact of using artificial intelligence tools: Construction of a synthetic index of application in higher education, Cogent Education 11 (2024) 2287917.
[49] I. A. Almohammadi, E. O. Hamad, B. A. Alqarni, M. A. Alzahrani, J. A. Aljohani, Graduate students' use of artificial intelligence applications in scientific research: Prevalence and association with perceived impostor syndrome, International Journal for Research in Education 49 (2025) 123–188.

# A. Ensuring Academic Integrity: Oral Verifications of Your Work

Due to the recent rise of AI capabilities, and that the course is held fully remotely, we have decided to use semi-randomized oral spot checks to ensure academic integrity and a genuine understanding of the material. You might be asked to participate after submitting any assignment or exam on the course. The selection on which students are required to participate is made by teaching personnel or selected by random.

Additionally, these checks aim to discourage cheating, would it be through the use of generative AI tools or plagiarism. In the oral verification, you are required to explain the given submission/answer or discuss concepts used in those exercises in your own words. The schedule for the Zoom meeting is quite flexible, and the time will be decided in mutual agreement.

Feel free to ask for clarification if you're unsure about anything. We're all here to learn together remotely, and your honesty and effort are greatly appreciated

# B. Visual output

A simple exercise that's answer requires a flowchart to solve. "Present it as an ASCII tree." was only part of the input given to the model, and was not part of the original exercise description.

## B.1. Exercise Description

```
Assume the following context-free grammar:

<Program> -> <Statement List>
<Statement List> -> <Statement> <Statement List> | <empty>
<Statement> -> <Assignment Statement>
<Statement> -> <While Statement>
<Statement> -> <For Statement>
<Statement> -> <Method Call>
<Assignment Statement> -> <Identifier> <Math Operator> <Identifier>
<While Statement> -> WHILE <Condition>: <Statement List> ENDWHILE
<For Statement> -> FOR (<Assignment Statement>; <Condition>; <
   Assignment Statement>) <Statement List> ENDFOR
<Method Call> -> <Identifier> ( <Identifier> )
<Condition> -> <Identifier> <Logical Operator> <Identifier>
<Logical Operator> -> < | > | <= | >= | == | !=
<Math Operator> -> += | -= | *= | /= | =
<Identifier> -> [a-zA-Z]+ | [0-9]+

--------------------

What kind of parse tree results from the program below? Present it
   as an ASCII tree.

sum = 0
FOR (i=0;i<5;i++)
    sum += i
    print(sum)
ENDFOR
```

### B.2. Answer produced by GPT-4

```
<Program>
|
+- <Statement List>
   |
   +- <Statement>
   |   |
   |   +- <Assignment Statement>
   |       |
   |       +- <Identifier> "sum"
   |       +- <Math Operator> "="
   |       +- <Identifier> "0"
   |
   +- <Statement List>
       |
       +- <Statement>
       |   |
       |   +- <For Statement>
       |       |
       |       +- <Assignment Statement> "i = 0"
       |       +- <Condition> "i < 5"
       |       +- <Assignment Statement> "i++"
       |       +- <Statement List>
       |           |
       |           +- <Statement>
       |           |   |
       |           |   +- <Assignment Statement>
       |           |       |
       |           |       +- <Identifier> "sum"
       |           |       +- <Math Operator> "+="
       |           |       +- <Identifier> "i"
       |           |
       |           +- <Statement List>
       |               |
       |               +- <Statement>
       |                   |
       |                   +- <Method Call>
       |                       |
       |                       +- <Identifier> "print"
       |                       +- <Identifier> "sum"
       |
       +- <Statement List> <empty>
```