Investigating the Role of ChatGPT in Supporting Text-Based Programming Education for Students and Teachers

Markus Wieser[1(✉)], Klaus Schöffmann[2], Daniela Stefanics[2], Andreas Bollin[1], and Stefan Pasterk[1]

[1] Institute of Informatics Didactics, University of Klagenfurt, Klagenfurt am Wörthersee, Austria
{markus.wieser,andreas.bollin,stefan.pasterk}@aau.at
[2] Institute of Information Technology, University of Klagenfurt, Klagenfurt am Wörthersee, Austria
{klaus.schoffmann,daniela.stefanics}@aau.at

Abstract. Teaching text-based programming poses significant challenges in both school and university contexts. This study explores the potential of ChatGPT as a sustainable didactic tool to support students, freshmen, and teachers. By focusing on a beginner's course with examples also relevant to vocational schools, we investigated three research questions. First, the extent to which ChatGPT assists students in solving and understanding initial examples; secondly, the feasibility of teachers utilizing the chatbot for grading student solutions; and finally, the additional support ChatGPT provides in terms of teaching. Our findings demonstrate that ChatGPT offers valuable guidance for teachers in terms of assessment and grading and aids students in understanding and optimizing their solutions.

Keywords: AI and Machine Learning · Pedagogy/Teaching Approach · Programming Education · Higher-Secondary and Vocational Schools

1 Introduction

Teaching text-based programming is a challenge that requires a lot of practice, reflection, and thus time. It involves technical knowledge and understanding of the fundamental concepts needed to solve programming problems. In this context, there is currently hype around ChatGPT (and similar tools based on large-language models) that teachers, students, and learners use to write texts, formulate papers, and solve programming tasks [7].

ChatGPT was initially developed as a text generation tool and is known for its ability to produce human-like text. This development has led to its use for

This work has been partially funded by the Austrian Federal Ministry of Education, Science and Research (BMBWF), via the CodeAbility project.

J.-P. Pellet and G. Parriaux (Eds.): ISSEP 2023, LNCS 14296, pp. 40–53, 2023.
https://doi.org/10.1007/978-3-031-44900-0_4

solving programming tasks, as well. It may be seen as a tool for cheating, but with its text and program generation skills, it provides many teaching opportunities [1]. Of course, there are reasonable doubts about the quality of the results provided by ChatGPT, and some experts refer to such tools as stochastic parrots [2], as they merely assemble prefabricated text modules and do not perform actual problem-solving. This issue may be a problem in the classroom. When using ChatGPT as a tool during a lesson, students need help recognizing incorrectly generated answers or solutions. Lack of knowledge can lead to the internalization of wrong concepts, preventing the development of new competencies [9].

With this in mind, this paper is dedicated to the quality of support ChatGPT offers students and teachers when using it. It is about recognizing the limitations of this tool and understanding how it can best be used to support learning and understanding of programming. In particular, to better understand these issues, we selected examples and solutions from a beginners programming course at University of Klagenfurt that is also part of the computer science curriculum at vocational schools in Austria, and we investigate the following research questions:

- RQ-1: How much help does ChatGPT provide pupils to solve practical exercises?
- RQ-2: Can teachers use ChatGPT for grading solutions of the pupils?
- RQ-3: What other programming teaching assistance does ChatGPT offer?

The paper is structured as follows. Section 2 looks closely at publications on teaching programming and using ChatGPT in the school and university context. Section 3 provides an overview of the different possibilities that ChatGPT can be used in the classroom. Section 4 presents our study to evaluate the quality of the tool's responses. Section 5 presents our didactic recommendations, and Sect. 6 concludes the contribution with a summary and a short outlook.

2 Related Work

If we want to test the usefulness of a new tool in a school (or university) context, we must consider how the learning process takes place. An abundance of resources and activities can be found to master text-based programming.

Scientific work on teaching programming focuses on understanding and supporting the cognitive development of novice programmers. Research by Lister et al. explores the application of Bloom's and SOLO taxonomy to differentiate programming tasks and assess programming skill levels [13,14]. Investigations of the hierarchical development of programming skills, including reading, tracing, and writing program code, provide valuable educational insights [12]. Neo-Piagetian developmental stages offer a framework for understanding novice programmers' cognitive abilities and transitions, emphasizing the importance of aligning programming tasks with students' cognitive development [20].

In the realm of programming education, a lot of strategies have been explored. These include explicit teaching of problem-solving strategies [16], understanding novice programmers' mental models [3,15], evaluation of problem-solving

techniques in Java [6], and challenges faced by low-skilled students in code comprehension [19]. Studies also highlight the significance of detecting programming skills early [10], the analysis of sketch types in code reading tasks [4], learning trajectories for programming concepts [17], and the utilization of Parson's Problems as a skill acquisition tool [5]. Together, these diverse studies offer a comprehensive view of effective strategies and techniques for teaching programming.

Large language models, like GPT-3.5 or Open-AI Codex, are increasingly being utilized in programming education, and they are fascinating and terrifying simultaneously [8]. They offer an interactive learning environment and can assist in various tasks. They are used to improve error messages [11] and they are used to create programming exercises as well as explanations [18]. However, at the time of writing this paper, there are no articles looking closer at the quality of the results and their influence on teaching programming. As a contribution to existing knowledge, our work, therefore, takes a closer look at typical application areas and also illuminates them from a didactic perspective.

3 Quality Considerations

3.1 Setting and Methodology

We selected the *Introduction to Java Programming* lab course at University of Klagenfurt (winter term 2023), which is part of the *Informatics* curriculum (besides a few others) and had about 120 students enrolled. The lab course required students to solve 103 Java exercises (distributed over 10 assignments) that had to be submitted via the CodeRunner plugin of Moodle, which checks every exercise with a few test cases (and often some pre-defined code, where the student submission is embedded). The lab exam is also conducted via Moodle/CodeRunner, and consists of programming exercises only. As such, the exercises and tasks are comparable to tasks in the last two classes of vocational schools in Austria, and the results should also be applicable in the school context.

Our experiment, which is performed with ChatGPT 3.5, has two parts, aside from just letting ChatGPT solve the programming tasks. In the first part, we want to check whether ChatGPT could be used as a personal tutor; in the second part, we check its grading and assessment capabilities.

Solving Programming Tasks: For the evaluation of RQ-1, we copy the description of every exercise from Moodle to the ChatGPT prompt, check the response, and copy it back to the CodeRunner input window, where it is submitted and automatically checked with our test cases. In case the first provided solution of ChatGPT does not meet the requirements, we ask to regenerate the solution and provide more necessary details (or small change requests).

Grading and Assessment: For the evaluation of RQ-2, we ask ChatGPT first to assess students' solutions to eight defined tasks. We use the solutions from an exam at the end of the course, in which a total of 84 students participated (for eight different tasks to solve). From all of the solutions, we end up with 672 code snippets to analyze. We give ChatGPT the respective task definition and solutions and ask it to score each solution up to a given maximum of points. We then analyze the results and compare them to the points the teachers have given, using the Spearman Correlation. We use this method because the data is not normally distributed. Furthermore, we select a random group for each of the eight tasks, for which we ask ChatGPT to assess the tasks two more times in different chats, and we finally compare the consistency of the assessments.

We also take a sample of five students' solutions to the four tasks with the highest possible points. We select those with the highest difference between teacher and ChatGPT. When there are more than five, we randomly select some. We then give the teachers three statements to be evaluated on a five-point Likert scale. In addition, there is the possibility to submit explanatory notes to each question.

Another aspect we are interested in is the code understanding of ChatGPT compared to a teacher. Therefore, we define four types of error that are typical for beginners of the Java programming language: (1) syntax errors, (2) runtime errors, (3) logical errors, and (4) semantic errors. We then compare the error analysis of ChatGPT with the teacher's analysis.

Individualized Instruction: To get tasks adapted to the learning needs of students or whole groups, we present ChatGPT with the student's submissions and ask it, based on the mistakes, to create tasks that could be used by the students to specifically practice these topics. We use two runs: in the first run we just present ChatGPT the task and the student's solution and ask how a possible task could look like, so that the students could improve their skills. In the second run, we also give suggestions, such as: "It seems that the student has problems with indexing." We give five exemplary tasks to ChatGPT. Each task is submitted once as a single submission and another time as a group submission.

4 Findings and Recommendations

In this section, we will present and discuss the results for the areas *Solving Ability*, *Grading and Assessment* and *Individualised Instruction*. We briefly introduce each topic and close the respective section with didactical recommendations.

Some papers are currently discussing the possibilities of ChatGPT in education. As Zhai [21] points out in the paper focusing on academic writing, AI is certainly capable of supporting students and teachers in a wide range of tasks. Some of these areas are also relevant to the teaching and learning of programming. In our experiment, we found that ChatGPT, aside from being a simple solving tool, can be an asset to programming education in other areas.

4.1 Solving Ability

Results: From a total of 103 Java programming exercises, 100 (97.09%) could be correctly solved by ChatGPT. For the vast majority of exercises (59.22%), already the first provided Java code could be directly submitted via CodeRunner and all test cases succeeded. Another 28.16% of exercises could be quickly solved by a simple *Regenerate response* request, so that the second proposed solution was correct. For another 9.71% of exercises, some further minor changes were required, which were successfully performed by ChatGPT after we provided more details. Examples of such changes are extreme situations that are checked by CodeRunner test cases (empty arrays), unnecessary Getter and Setter methods (that needed to be removed), wrong types of loops (e.g., for-each loops instead of normal for loops), and wrong naming of variables or methods. Figure 5 (see Appendix) shows such a code example where a CodeRunner test case failed for the parameters (null, 0). A request was then made to ChatGPT that the code should work for these parameters as well and it adapted its solution to what is shown in Fig. 6 (see Appendix). With this solution all tests finally succeeded. Only 2.91% of all exercises could not be solved by ChatGPT.

Discussion: It should come as no surprise that ChatGPT is able to solve almost all the tasks of a beginner's course in programming. If the solution does not meet the requirements, adding some minor changes will solve the problem. Even at this stage, ChatGPT could be considered as "Solution for all problems", at least as far as the beginners' courses in programming are concerned. So, from now on it is no longer possible to measure the students' learning progress on the basis of their submissions to assignments (or to determine what their abilities and weaknesses are), since the solutions could have been generated by generative AI tools. But do harder tasks solve the problem? In this case, the tasks would have to be so complex that Chat GPT cannot properly solve them. However, such tasks are definitely not suitable for a beginner's course in programming.

Didactical Recommendations: The very existence of ChatGPT now creates a major problem for all teachers in programming: How can I correctly assess and evaluate the skills and abilities of my students? What part of their tasks is made by themselves and where did they use tools like ChatGPT? Unfortunately, we have to assume that ChatGPT will solve all tasks of this kind in the future, and if we continue to set tasks as we are now, there will be no learning progress for students. So there have to be other methods of measuring the students' abilities in programming. First, there is the possibility of writing tests and exams. In such a situation, it is much harder to use unauthorized tools and so the skills and weaknesses are shown. On the other hand, there are students with exam anxiety. For those such a mode would be horrifying. Another possibility is to test the students' abilities in class. In our University the students get weekly task sheets and they then hand in the solved tasks week by week. In classes, students are randomly selected to present an example they have solved. It would not be a big deal to let them solve very similar tasks instead of the task already

solved. There are several advantages. First, because they do not know what task is given to them in the exercise, the students are not able to just present a ChatGPT solution and therefore have to understand the concepts and learn programming by themselves. For students who are always learning along, these changes do not pose much of a challenge, as long as the new task actually differs only slightly from the original task.

4.2 Grading and Assessment

The quality of a program depends on different criteria, such as functionality, efficiency or correctness. Assessing a program according to these criteria and grading the student accordingly can be very time-consuming and complex, depending on the number of students and the size of the program. Here, ChatGPT could be used to automatically grade assignments based on criteria defined by the teachers.

Results: When looking at the results of this analysis, we were pretty much surprised. In general, the grading of ChatGPT was pretty useful. The awarding of points was coherent and comprehensible at first sight. It also explained the awarding scheme (see Fig. 7 in Appendix).

We computed the Spearman correlation for the teacher's points in comparison to the points awarded by ChatGPT, separate for all tasks. We used Spearman because the data is not normally distributed. The results received are shown in Fig. 1. As visible in the figure, there is a maximum correlation of 0.951, which is pretty strong, and a minimum correlation of 0.678, which also indicates a medium to strong positive relationship between the ranks.

When we then compared the assessments of ChatCPT to each other, we found that ChatGPT's assessments are not constant and accordingly not always the same. In Fig. 2 the Spearman correlation between the three different attempts (A1,A2,A3) and the respective teacher (T) is shown.

Task	Ex 1	Ex 2	Ex 3	Ex 4	Ex 5	Ex 6	Ex 7	Ex 8
r	0.844	0.727	0.840	0.678	0.839	0.934	0.951	0.905

Fig. 1. Spearman Correlation of the different tasks

1 - D	A1	A2	A3
T	0.822	0.771	0.833
A1		0.704	0.704
A2			0.989

2 - C	A1	A2	A3
T	0.861	0.861	0.881
A1		1.000	0.978
A2			0.978

3 - A	A1	A2	A3
T	0.936	0.843	0.991
A1		0.911	0.927
A2			0.843

4 - E	A1	A2	A3
T	0.529	0.684	0.526
A1		0.688	0.773
A2			0.930

5 - D	A1	A2	A3
T	0.932	0.937	0.758
A1		0.868	0.687
A2			0.899

6 - B	A1	A2	A3
T	1.000	0.974	0.974
A1		0.974	0.974
A2			1.000

7 - C	A1	A2	A3
T	0.861	0.978	1.000
A1		0.947	0.861
A2			0.978

8 - F	A1	A2	A3
T	0.778	0.710	0.869
A1		0.767	0.743
A2			0.579

Fig. 2. Spearman Correlation of different tasks including ChatGPT attempts

The correlation between T and A1 differs from the correlation in Fig. 1, because here we only used one exam group instead of the entire population. As you can see, different grading attempts return different results, although the input was the same. When going into detail, we found that ChatGPT gave the maximum amount of points and the minimum amount of points to the same submission in different grading rounds. So on one attempt, it gave 10 Points to the student, and on another attempt, it gave 0 Points.

When evaluating the selected student's solutions and ChatGPT's explanations, we found some pretty interesting things. We took the five solutions of each task with the highest difference between ChatGPT and the teacher's assessment and then asked the teachers to evaluate three statements by five-point Likert scale, where 1 corresponds to *does not apply* and 5 corresponds to *applies* for each solution.

1. ChatGPT's awarding of points is coherent and comprehensible
2. Based on ChatGPT's scoring, I would change my own assessment
3. The assessment only by ChatGPT would have led to a reasonable result for this solution

The mean of the answers can be found in Fig. 3. The number of the item corresponds to the number of the question.

Mean	Item 1	Item 2	Item 3
Ex 2	2.20	1.40	2.20
Ex 3	1.00	1.00	1.00
Ex 5	1.40	1.40	1.60
Ex 8	2.25	2.00	3.00
Total	**1.68**	**1.42**	**1.89**

Fig. 3. Evaluation of the Likert scale of all three items

As we can see, the teachers share the opinion that ChatGPT's awarding of points is neither coherent and comprehensible (1.68/5), nor do they want to change their own assessments (1.42/5), nor do they think the assessment only by chatGPT would have led to a reasonable result for this solution (1.89/5). When we take a closer look at the answers of the teachers, then we find, that ChatGPT ignored some mistakes. So the explanation of one of the teachers was: "ChatGPT overlooks some errors here: (1) the current element is never compared with the minimum, but with the neighbor, (2) array index and value are confused (in the variable min the index is stored instead of the value), (3) the initialization of min is syntactically wrong. The reason for the deduction of only 1 point (loop condition) is not comprehensible."

Regarding the error analysis capabilities of ChatGPT we computed statistics on the five error types for five programming exercises of a programming exam where 84 students participated. It is important to note that the teacher checked for syntax and runtime errors first and did not look further if one of these errors

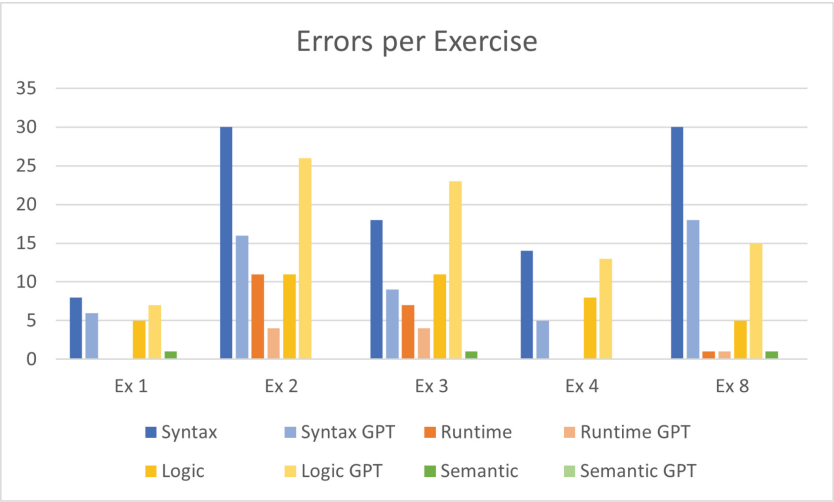

Fig. 4. Comparison of errors per exercise

was detected. ChatGPT on the other hand always checked for several types of errors and mentioned all of them in a list. However, for statistics, we only counted the first error type mentioned for the teacher and ChatGPT. The overall sum of errors is similar: 162 errors were detected by the teacher, while 147 were detected by ChatGPT. The total number of errors for each of the five exercises is also very similar (Teacher vs. ChatGPT): Ex 1: 14 vs. 13, Ex 2: 52 vs. 46, Ex 3: 37 vs. 36, Ex 4: 22 vs. 18, and Ex 8: 37 vs. 34. When taking a closer look at the different types, the statistics reveal that the teacher detected 100 syntax errors, 19 runtime errors, 40 logical errors, and 3 semantic errors, while ChatGPT found 54 syntax errors, 9 runtime errors, 84 logical errors, and 0 semantic errors (see Fig. 4). The difference in syntax and logical errors can be explained by the different prioritisation as described above: the teacher did not further assess the solution if it did not compile (e.g., wrong syntax, incomplete code, etc.) or crash (e.g., IndexOutOfBoundsException), while ChatGPT inspected the code in detail. As the sum of both error types is very close (140 vs. 138) and the teacher agreed after another check of several solutions that many solutions with syntax errors also include logical errors, we can conclude that the assessment of ChatGPT is very accurate in our qualitative evaluation. The same is also true for runtime errors. For example, in one exercise, the student used the following code for an inner for-loop: `for (int j=0;i<arr2d[i].length; j++)`. This led to a run-time error, and this is what the teacher counted. ChatGPT can also detect the run-time error but first mentions the logical error that variable i is used instead of variable j for the check, which is the type of error we counted for the statistics. Semantic errors were classified by the teacher when incorrect operators were used (e.g., the XOR operator instead of Math.pow()), or when students used any fixed hard-coded values instead of variables. These errors

were also found by ChatGPT but classified as logical errors. Overall we can summarize that the error analysis by ChatGPT is very accurate and complete, no single error could be identified that was not revealed by the chatbot as well.

Discussion: When taking a first look at the results of the first assessment attempt we found, that the grading of ChatGPT was strongly correlated to the assessment of the teachers. We had a maximum Spearman correlation of 0.951 and a minimum correlation of 0.678. Both are strong correlations, the higher one even *very strong* (see Fig. 2). So from a statistical point of view everything was fine. The problem is, grading and assessment is not statistical, it is individual. The teacher's job is to determine if a student has achieved the learning objectives or competencies and where the weaknesses and strengths are. And such a task can not be performed by a statistical model that assigns zero points to a solution in one prompt, and full points on the same submission in another prompt. This inconsistency in grading disqualifies ChatGPT as a standalone grading and assessment tool. Another problem here is, that ChatGPT overlooked some mistakes and, according to the teachers, gave non comprehensible assessments, on the other hand it was pretty accurate in finding errors and error types.

Didactical Recommendations: In summary it could be said, that ChatGPT should not be used as an unsupervised grading tool. In our opinion, the negative aspects, such as incorrect assessment or inconsistent scoring, outweigh the statistical correlation. But it could be used as an *alternative opinion.* If you have a submission additionally checked by ChatGPT, you will get a further error analysis and a score hint if needed. This could help to find errors that were overlooked by oneself. However, it should be noted that such a procedure could require additional time resources. The assessment and grading itself should definitely continue to be done by the teacher.

4.3 Individualized Instruction

Individualized Instruction offers several benefits to schools and learners alike. It enables students who may be above or below average in their learning abilities to proceed at their own pace, ensuring optimal learning outcomes. This personalized approach allows students to avoid repeating portions of a course that they have already mastered, leading to a more efficient use of time and resources. In this area, ChatGPT could take on the function of a personal tutor who, based on the available data, provides individual explanations and error analyses.

Results: When testing the tutoring abilities of ChatGPT, we found, that its general recommendations without a defined learning goal were pretty generic. In these cases, the suggestion of ChatGPT is mostly to solve the same example again, only it adds "Additional Guidelines". For example, if the student has simply specified the length in the form of hardcoding instead of array. length, the additional guidelines say: "avoid hardcoding". However, there were also suggestions that were not suitable despite the learning objective being pointed out.

For example, one suggestion from ChatGPT was that the student should just fix their mistakes. In other suggestions it wasn't really clear what the student was supposed to do. However, the majority of ChatGPT's suggestions after defining the learning objective was quite good. Most of the time ChatGPT let the student solve the same or similar example again, but gives detailed solution paths. We consider these examples suitable, because the student can work through the example again step-by-step and thus get an idea how one could approach such tasks. In the group comparisons, there is sometimes even a separate task for each student to practice, which addresses the individual errors of each individual and then another common task.

Discussion: As already pointed out in the paragraph above, the general recommendations of ChatGPT are generic. If students want to practice and therefore know their exercise potential, recommendations like doing the same task again, just fix the mistakes or avoid hardcoding are not useful and in fact demotivating. If a learning goal is submitted, the recommendations are more useful and therefore could be used in individual practicing, combined with step-by-step solutions and detailed solution paths. Here, ChatGPT could be used as an individual tutor, giving the students the opportunity of working through a different example or getting explanations on their own pace.

Didactical Recommendations: When trying to use ChatGPT as personal tutor there are some limitations. In order to get useful results, the request should be defined by the teacher. So the teacher should tell each student, which subject areas need further practice and how the request should be made. The students then should be able to practice their learning needs with the aid of ChatGPT.

5 Conclusion

Our research questions can now be answered as follows. Concerning question (RQ-1) "How much help does ChatGPT provide pupils to solve practical exercises?" we found that unsurprisingly ChatGPT is in fact capable of solving all the tasks given in a programming beginner's course. This problem could be solved by increasing the number of tests and exams or challenging the students in class using slightly modified tasks. Concerning question (RQ-2) "Can teachers use ChatGPT for grading solutions of the pupils?" we found that ChatGPT could be used as a grading and assessment tool, but in our opinion, the negative aspects, such as incorrect assessment or inconsistent scoring, outweigh the strong statistical correlation. If you want to get a "second opinion" on your student's submission, ChatGPT probably will do a good job. Concerning question (RQ-3) "What other programming teaching assistance does ChatGPT offer?" we found out that ChatGP could be used as a personal tutor. When given useful requests ChatGPT is able to modify given examples in order to practice a defined subject and also provides step-by-step solutions, which allow students to review and learn at their own pace. In future work, we plan to investigate whether the

results presented in this paper are also valid for ChatGPT4 and later models and to use the findings of this paper to further investigate the use of AI systems in didactics of programming.

A Codesnippets and Grading Suggestion

```
public static boolean search(int[][] arr, int num) {
    for (int i = 0; i < arr.length; i++) {
        for (int j = 0; j < arr[i].length; j++) {
            if (arr[i][j] == num) {
                return true; // if num is found in the array, return true
            }
        }
    }
    return false; // num is not found in the array, return false
}
```

Fig. 5. Solution by ChatGPT which failed one of the CodeRunner test cases.

```
public static boolean search(int[][] arr, int num) {
    if (arr == null) {
        return false;
    }
    for (int i = 0; i < arr.length; i++) {
        if (arr[i] == null) {
            continue;
        }
        for (int j = 0; j < arr[i].length; j++) {
            if (arr[i][j] == num) {
                return true; // if num is found in the array, return true
            }
        }
    }
    return false; // num is not found in the array, return false
}
```

Fig. 6. Working solution by ChatGPT after a small change request.

Based on the issues identified in the student's solution, I would deduct points as follows:

1. Variable type: Deduct 1 point for using the wrong variable type for `pi`.
2. Division by integer: Deduct 2 points for not using floating-point division in the expression for the Leibniz series.
3. Exponentiation function: Deduct 1 point for not importing the `Math.pow` function.
4. Loop range: Deduct 1 point for using the wrong loop range.
5. Calculation of pi: Deduct 2 points for not dividing the result by `n` before multiplying it by 4.

This results in a total deduction of 7 points. Therefore, the student should receive 3 points out of 10 for this exercise.

Fig. 7. Grading suggestion of ChatGPT

References

1. Abramson, A.: How to use ChatGPT as a learning tool. Monitor Psychol. **54**(3), 36–44 (2023). https://www.apa.org/monitor/2023/06/chatgpt-learning-tool
2. Bender, E.M., Gebru, T., McMillan-Major, A., Shmitchell, S.: On the dangers of stochastic parrots: Can language models be too big? In: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and TransparencyMarch (FAccT 2021), pp. 610–623. Association for Computing Machinery, New York (2021). https://doi.org/10.1145/3442188.3445922
3. Bornat, R., Dehnadi, S., Simon: mental models, consistency and programming aptitude. In: Proceedings of the Tenth Conference on Australasian Computing Education (ACE 2008), vol. 78, pp. 53–61. Australian Computer Society Inc., AUS (2008)
4. Cunningham, K., Blanchard, S., Ericson, B., Guzdial, M.: Using tracing and sketching to solve programming problems: replicating and extending an analysis of what students draw. In: Proceedings of the 2017 ACM Conference on International Computing Education Research (ICER 2017), pp. 164–172. Association for Computing Machinery, New York (2017). https://doi.org/10.1145/3105726.3106190
5. Ericson, B.J., Foley, J.D., Rick, J.: Evaluating the efficiency and effectiveness of adaptive parsons problems. In: Proceedings of the 2018 ACM Conference on International Computing Education Research (ICER 2018), pp. 60–68. Association for Computing Machinery, New York (2018). https://doi.org/10.1145/3230977.3231000
6. Hanks, B., Brandt, M.: Successful and unsuccessful problem solving approaches of novice programmers. In: Proceedings of the 40th ACM Technical Symposium on Computer Science Education (SIGCSE 2009), pp. 24–28. Association for Computing Machinery, New York (2009). https://doi.org/10.1145/1508865.1508876
7. Honegger, B.D.: ChatGPT & School - Assessments by the Chair of "Digitalisierung und Bildung" at the University of Teacher Education Schwyz. pädagogische hochschule schwyz (2023). https://mia.phsz.ch/pub/LLM/WebHome/2023-chatgpt-und-schule-v128-en.pdf. Accessed 27 Apr 2023

8. Jacques, L.: Teaching cs-101 at the dawn of chatgpt. ACM Inroads **14**(2), 40–46 (2023). https://doi.org/10.1145/3595634
9. Joyner, D.A.: ChatGPT in education: partner or pariah? XRDS **29**(3), 48–51 (2023). https://doi.org/10.1145/3589651
10. Kesselbacher, M., Bollin, A.: Towards the Use of Slice-based cohesion metrics with learning analytics to assess programming skills. In: Third International Workshop on Software Engineering Education for the Next Generation (SEENG). arXiv preprint arXiv:2105.04974 (2021)
11. Leinonen, J., et al.: Using large language models to enhance programming error messages. In: Proceedings of the 54th ACM Technical Symposium on Computer Science Education (SIGCSE 2023), vol. 1, pp. 563–569. Association for Computing Machinery, New York (2023). https://doi.org/10.1145/3545945.3569770
12. Lister, R., Fidge, C., Teague, D.: Further evidence of a relationship between explaining, tracing and writing skills in introductory programming. SIGCSE Bull. **41**(3), 161–165 (2009). https://doi.org/10.1145/1595496.1562930
13. Lister, R., Leaney, J.: Introductory programming, criterion-referencing, and bloom. ACM SIGCSE Bull. **35**(1), 143–147 (2003). https://doi.org/10.1145/792548.611954
14. Lister, R., Simon, B., Thompson, E., Whalley, J.L., Prasad, C.: Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. ACM SIGCSE Bull. **38**(3), 118–122 (2006). https://doi.org/10.1145/1140123.1140157
15. Ma, L., Ferguson, J., Roper, M., Wood, M.: Investigating and improving the models of programming concepts held by novice programmers. Comput. Sci. Educ. **21**(1), 57–80 (2011)
16. de Raadt, M., Watson, R., Toleman, M.: Teaching and assessing programming strategies explicitly. In: Proceedings of the Eleventh Australasian Conference on Computing Education (ACE 2009) vol. 95, pp. 45–54. Australian Computer Society Inc, AUS (2009)
17. Rich, K.M., Strickland, C., Binkowski, T.A., Moran, C., Franklin, D.: K-8 learning trajectories derived from research literature: sequence, repetition, conditionals. ACM Inroads **9**(1), 46–55 (2018). https://doi.org/10.1145/3183508
18. Sarsa, S., Denny, P., Hellas, A., Leinonen, J.: Automatic generation of programming exercises and code explanations using large language models. In: Proceedings of the 2022 ACM Conference on International Computing Education Research (ICER 2022), vol. 1, pp. 27–43. Association for Computing Machinery, New York (2022). https://doi.org/10.1145/3501385.3543957
19. Snowdon, S., Snowdon, S.: Explaining program code: giving students the answer helps - but only just. In: Proceedings of the Seventh International Workshop on Computing Education Research (ICER 2011), pp. 93–100. Association for Computing Machinery, New York (2011). https://doi.org/10.1145/2016911.2016931
20. Teague, D., Lister, R.: Programming: reading, writing and reversing. In: Proceedings of the 2014 Conference on Innovation and Technology in Computer Science Education (ITiCSE 2014), pp. 285–290. Association for Computing Machinery, New York (2014). https://doi.org/10.1145/2591708.2591712
21. Zhai, X.: ChatGPT user experience: implications for education (2022)