



Plan More, Debug Less: Applying Metacognitive Theory to AI-Assisted Programming Education

Tung Phung¹(✉) , Heeryung Choi² , Mengyan Wu³ , Adish Singla¹ ,
and Christopher Brooks³

¹ MPI-SWS, Saarbrücken, Germany
mphung@mpi-sws.org

² University of Minnesota, Twin Cities, USA

³ University of Michigan, Ann Arbor, USA

Abstract. The growing adoption of generative AI in education highlights the need to integrate established pedagogical principles into AI-assisted learning environments. This study investigates the potential of metacognitive theory to inform AI-assisted programming education through a hint system designed around the metacognitive phases of planning, monitoring, and evaluation. Upon request, the system can provide three types of AI-generated hints—planning, debugging, and optimization—to guide students at different stages of problem-solving. Through a study with 102 students in an introductory data science programming course, we find that students perceive and engage with planning hints most highly, whereas optimization hints are rarely requested. We observe a consistent association between requesting planning hints and achieving higher grades across question difficulty and student competency. However, when facing harder tasks, students seek additional debugging but not more planning support. These insights contribute to the growing field of AI-assisted programming education by providing empirical evidence on the importance of pedagogical principles in AI-assisted learning.

Keywords: Programming Education · Feedback Generation · Metacognitive Theory · Generative AI

1 Introduction

Recent advancements in generative AI have sparked significant interest in the field of programming education [10, 25, 34, 45], especially in the generation of personalized feedback [23, 35, 45, 48]. However, existing studies often focus on technical correctness [15] or student preference [25, 30] and overlook the importance of grounding AI-generated feedback in well-established pedagogical theories, potentially limiting the effectiveness of such feedback in student learning.

To address this gap, we propose enhancing AI-generated hints through the use of metacognitive scaffolds [24, 40, 44]. Metacognitive scaffolds are instructional support mechanisms that help students plan, monitor, and evaluate their

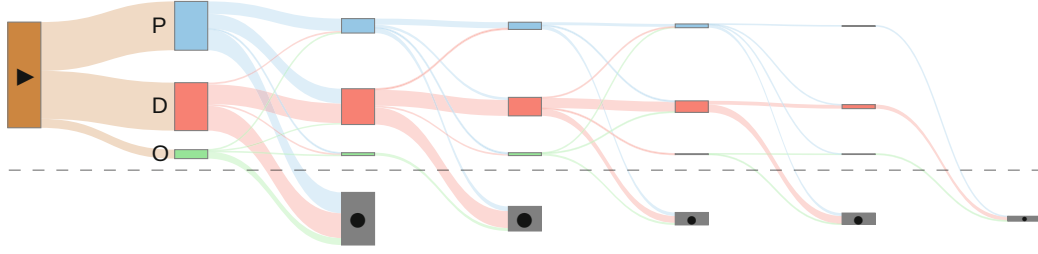


Fig. 1. Overview of requested hints (725 hints in 366 student-question pairs). Throughout the paper, P, D, and O denote a planning, debugging, and optimization hint, respectively. ► marks the beginning and ● marks the end of problem-solving. The graph displays sequences of hints, link sizes depicting counts.

learning processes while fostering self-regulation and strengthening adaptive problem-solving skills [14, 18, 39]. These scaffolds are crucial in programming education, where students often struggle with structuring approaches to solving (planning) [11, 32], identifying and fixing errors (monitoring) [31], and optimizing solutions (evaluation) [38]. By grounding AI-generated hints in these metacognitive phases, we aim to provide not only technical assistance but also structured yet flexible support to promote students’ metacognitive development. Specifically, we design three corresponding types of hint: *planning*, *debugging*, and *optimization*, which we collectively refer to as **AI-generated hints based on Metacognitive Scaffolds (AIMS hints)**. To foster students’ metacognitive awareness of their problem-solving stages, we adopt a learning-assisted approach in which we set a quota for total hints per question and let the students decide for themselves which hint type to request based on their needs. Figure 1 provides an overview of how students utilized these hints in our deployment of this system.

Our study is centered on the following research questions:

RQ1: How do AIMS hints impact students’ help-seeking behaviors?

RQ2: How do those behaviors relate to students’ problem-solving performance?

We examine these questions across all students and subsets based on question difficulty and student competency levels. Our contributions are as follows:

- **Hint and system design.** Using metacognitive theory, we design a hint system that offers three types of AIMS hints: planning, debugging, and optimization, and evaluate these hints through a classroom field study.
- **Student behavior analysis.** We analyze student help-seeking behaviors, revealing trends such as students value planning hints highly but often under-utilize them in favor of debugging hints, especially when facing harder tasks.
- **Student performance analysis.** We find that planning hints are linked to better performance, notably for higher-competency students.
- **Code release.** To enhance reproducibility and aid future research, we publicly release the implementation of our AIMS hint generation techniques.

By integrating metacognitive scaffolds into AI-generated hints, this work not only contributes to a deeper understanding of how personalized AI support can enhance programming education but also provides practical insights for designing pedagogically-grounded educational AI systems.

2 Related Work

Metacognitive Scaffolds and Self-regulated Learning. Many efforts have been made to incorporate metacognitive scaffolds in programming education. The benefits of metacognition on learning behaviors and performance have been consistently shown through research [8, 37, 40, 43, 44, 49]. For example, Vieira et al. [43] found that novice computer science students wrote longer self-explanation in-code comments compared to experienced peers because they saw self-explaining as a learning opportunity. Choi et al. [8] showed that prompting reflection after programming tasks was correlated with better learning perception and performance on both immediate and delayed post-tests. Yilmaz and Yilmaz [49] showed that students who received personalized metacognitive feedback weekly engaged significantly more in a Computer I course. Inspired by these insights, we ground in metacognitive theory to design AI-generated scaffolding hints.

AI-Generated Feedback in Programming Education. Recent research has explored AI-generated feedback in programming education [45, 50]. Phung et al. [35] found that providing symbolic information such as the buggy output and fixed programs in prompts can improve the quality of AI-generated debugging hints. Lohr et al. [23] showed that AI can be directed to provide feedback that focuses on specific aspects, such as knowledge about task constraints or performance. Xiao et al. [48] explored students’ perceptions of hints with varying detail levels, revealing that their effectiveness depends on context and that high-quality next-step and debugging hints do not always facilitate student progress. Building on this line of work, our study employs AI techniques that utilize symbolic information to generate different types of hints tailored to students’ needs. Our hint system utilizes a button-based interface, as opposed to a chat-based one [25, 42], since the latter’s pedagogical effects are still unclear.

Students’ Help-Seeking Behaviors with Automated Hints. Several studies have examined how students interact with automated hints [21, 47]. Marwan et al. [28] found that data-driven next-step hints improved immediate performance, and when paired with self-explanation prompts, led to learning gains. Expanding on this, we investigate the association between AI-generated hints and student performance. Price et al. [36] found that the quality of initial hints positively correlates with help abuse. To address this, we set a fixed quota for hint use to prevent over-reliance on help. Wiggins et al. [47] characterized student’s hint-seeking behavior along two axes of elapsed time and code completeness, while Bui et al. [7] explored different hint formats, including text and skeleton code. Our study extends this research by analyzing additional aspects, including engagement, perception, and hint request sequence in the context of AIMS hints.

AI Support and Metacognition. Recently, concerns have arisen that AI technologies might reduce students’ engagement in metacognitive practices, a concept referred to as *Metacognitive laziness* [13]. Our work explicitly addresses this by integrating metacognitive theory into AI-generated hints, aiming to support self-regulated learning (SRL) skills development rather than replace it.

Hint type	Description
Planning	A hint aimed at helping you to identify the steps needed to solve the question.
Debugging	A hint aimed at helping you to identify and fix a bug in your current program.
Optimization	A hint aimed at helping you to optimize your current program for better performance and readability.

Fig. 2. AIMS hint types with the descriptions provided to students.

3 Study Setup

This section outlines the study context, our proposed AIMS hints, the deployment, and methods to estimate question difficulty and student competency.

3.1 Course and Students

Course Overview. This study was conducted in a Python-based introductory data science course as part of an online Master’s program at the University of Michigan. The four-week course featured weekly assignments covering key topics such as regular expressions, *pandas* data frame manipulation, Excel processing, and CSV file handling. Each assignment, delivered as a Jupyter notebook, consisted of three to four programming questions (14 in total), requiring students to complete Python functions for specific tasks. Assignments were due weekly, and students could submit multiple times before the deadline, with their highest score counting toward the final grade.

Student Overview. Overall, 102 students enrolled in the course: 71 males, 27 females, and four unspecified. Their ages range from 18 to 58 (mean = 32, stddev = 8.5). Requesting hints was voluntary, with no additional incentives or penalties, and instructors were unaware of whether or how often students requested hints. Students were informed about the research aspect of the initiative, including anonymous data recording and the AI-generated nature of hints which might not always be correct. This study was deemed exempt from oversight by the Institutional Review Board under application number HUM00251143.

3.2 Hint Types and AI-Generation Techniques

AIMS Hints. We mapped the three metacognitive phases of planning, monitoring, and evaluation onto disciplinary terms of planning, debugging, and optimization. Each hint type was designed with a specific goal: planning hints assist in initial strategy formulation, debugging hints aid in issue identification and resolution, and optimization hints foster code quality reflection and improvement for students aiming to exceed assignment expectations (see Fig. 2).

Techniques for Generating AIMS Hints. Each hint type is generated by a technique, all with careful consideration of incorporating “guard-rails” instruc-

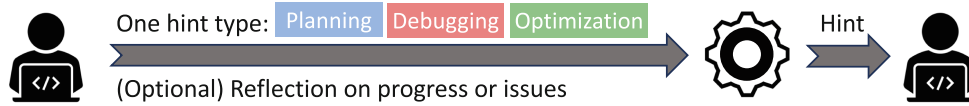


Fig. 3. Interaction between a student and our hint system for requesting a hint.

tions [22] to prevent the AI model (GPT-4o [19]) from revealing the solutions.¹ Our technique for debugging hints is adapted from previous studies that showed good performance in data science programming education [35, 50]. It follows a two-phase process: First, it extracts symbolic information including (1) the buggy output (obtained from running the student’s program) and (2) a repaired program (obtained from requesting the AI model). Second, it uses this information along with the student’s code and any reflection (detailed in Sect. 3.3) to prompt the AI to generate an explanation (leveraging Chain-of-Thought [46]) and a Socratic-style hint for a single bug (to be provided to the student). Since there were no existing techniques for generating high-quality planning and optimization hints, we adjust this technique for the other two hint types. For planning hints, we modify the prompt’s language to focus on problem-solving steps rather than debugging and remove the repaired program to shift emphasis away from errors. For optimization hints, we follow a similar two-phase process but replace the repaired program with an AI-generated optimized program—focusing on short running time while still requiring correctness.

3.3 Hint System and Student Interaction

AIMS Hint System. We develop a hint system consisting of two main components: (1) a backend server for generating hints using the techniques introduced in Sect. 3.2 and (2) a JupyterLab extension as the interface for interacting with students. To prevent over-reliance on AI and foster students’ metacognitive awareness, we set a quota limit of five hints per question and allow students to choose the type of hint to request. The extension displays three buttons below each assignment question, enabling students to request planning, debugging, or optimization hints. Before the course, the instructor introduced the hint system and demonstrated how to use it to the class. Students could always click a “?” button located next to the hint buttons to view descriptions of the hint types (as in Fig. 2). The first time a student requests a hint, a “Consent” pop-up informs them about the research aspect of the system (see Sect. 3.1). Students can only proceed to request hints after agreeing to this notice. To ensure easy access to previous hints, each hint is stored in a collapsible widget below the corresponding question’s hint buttons. When a student reopens a notebook, these widgets are collapsed by default and can be expanded with a click to revisit previous hints.

Student Interaction. Figure 3 demonstrates the interaction between students and our system for requesting hints. When requesting hints, students are encouraged to reflect on their progress or issues. These reflections serve dual purposes: promoting students’ engagement by prompting them to articulate their

¹ <https://github.com/machine-teaching-group/aied2025-plan-more-debug-less>.

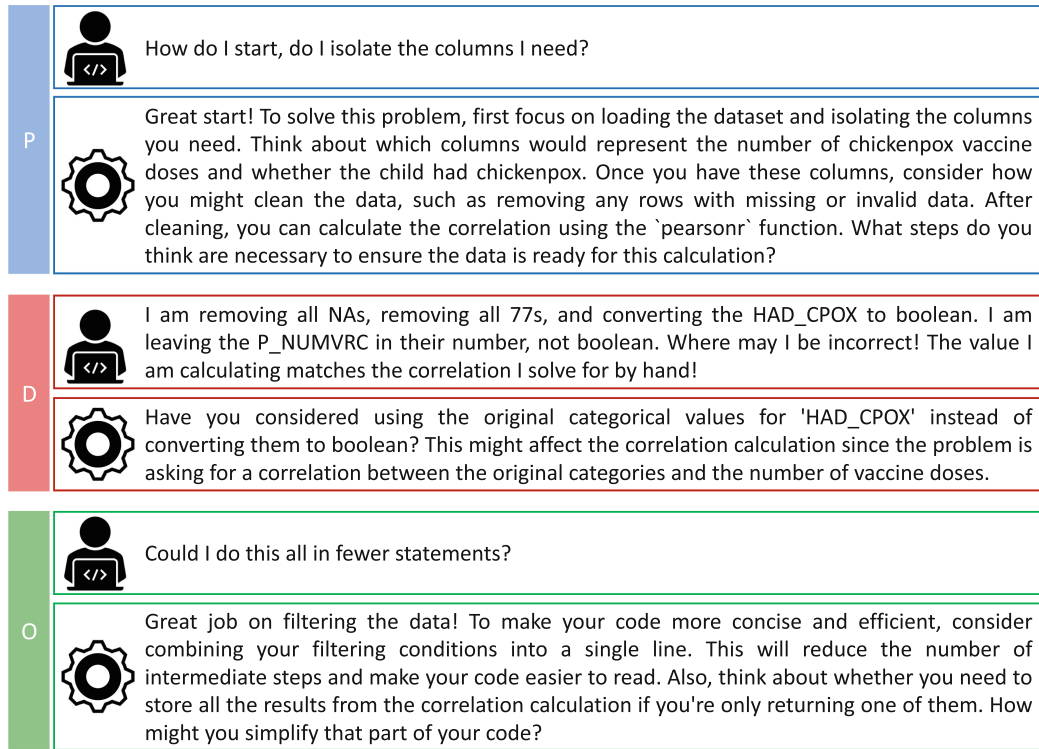


Fig. 4. Examples of hint types: student reflections and received hints in the study.

thoughts and providing the AI model with context to generate a relevant hint. Once the backend generates a hint, it is sent back to the interface and displayed to the student, accompanied by two feedback buttons: “thumb up” and “thumb down”, allowing the student to rate the hint as helpful or unhelpful, respectively.

3.4 Data Collection

We collected comprehensive data on student behavior and performance, including hint requests and revisits, assignment submissions, and final solving status. Of 102 students, 101 (99%) activated the JupyterLab extension, and 76 (75%) requested at least one hint. Figure 4 shows examples of provided hints. Figure 5 provides a breakdown of students and hints across 14 assignment questions.

3.5 Question Difficulty and Student Competency

To analyze the impacts of AIMS hints across varying conditions, we categorize questions by difficulty and students by competency. Question difficulty is estimated using past student performance in two prior iterations of the same course: the easier (harder) group consists of four highest- (lowest-) scored questions, one per assignment. Student competency is approximated based on the number of attempts until solving all questions in Assignment 1: fewer attempts indicate higher competency. We designate the top third (34 students) as the higher-competency group and the bottom third (34 students) as the lower-competency group. Since Assignment 1 served as a proxy, it is excluded from competency-based analyses.

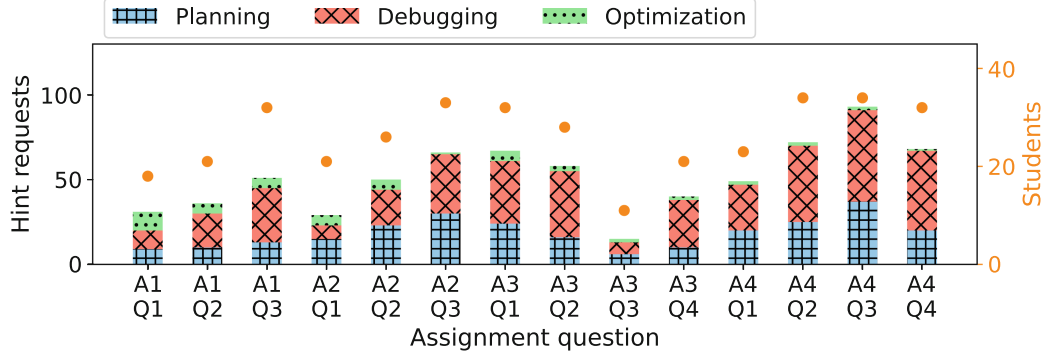


Fig. 5. Overview of students and hint requests. The orange dots and the right y-axis indicate the number of students who requested at least one hint (of any type) for each question. The stacked bars and the left y-axis represent the total number of hint requests per question, categorized by AIMS hint types. In total, students requested 258 planning, 411 debugging, and 56 optimization hints. (Color figure online)

4 RQ1: Impacts of AIMS Hints on Help-Seeking Behavior

This section addresses RQ1 by outlining our analysis setup (Sect. 4.1), presenting results (Sect. 4.2), and discussing key findings (Sect. 4.3).

4.1 Analysis Setup

To evaluate the impact of AIMS hints on student help-seeking behavior, we decompose RQ1 into three sub-questions: [RQ1a]: How do students engage with and perceive AIMS hints?, [RQ1b]: What behavioral patterns emerge in students' interactions with AIMS hints?, and [RQ1c]: How do these patterns vary based on question difficulty and student competency?

For RQ1a, engagement is measured by contemplation time and hint revisits, while perception is assessed through students' hint ratings. The contemplation time is defined as the time between receiving a hint and performing the next major action (i.e., requesting another hint or submitting a solution). To exclude irrelevant delays such as students taking a break, only durations up to $t = 1$ hour are considered for analysis.² Hint revisits are quantified by counting the number of times a student expands hint widgets to view previous hints.

For RQ1b, we analyze the sequence of requested hint types, the frequency of a type being present in the sequence, the first-requested type, and the most-requested type. Each of these is counted based on all student-question pairs.

For RQ1c, we examine how hint sequences and the presence of hint types vary across question difficulty and student competency (as defined in Sect. 3.5).

4.2 Results

Engagement and Perception of AIMS Hints. Figure 6 summarizes engagement and perception results. A Mood's median test [29] reveals a significant

² We note that other choices, such as $t = 0.5$ or $t = 2$ hours also yield similar results.

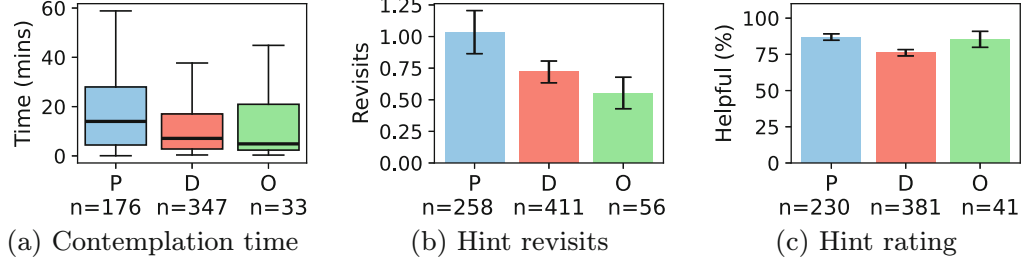


Fig. 6. Results for RQ1a: Student engagement and perception of hints. (a) demonstrates the amount of time students contemplated after receiving a hint. (b) shows the number of revisits per hint. (c) presents the average rating of hints.

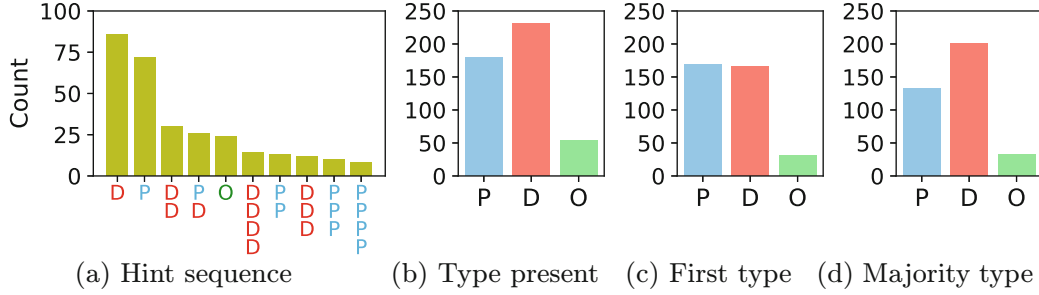


Fig. 7. Results for RQ1b: Patterns of student hint usage. In all plots, y-axis represents the counts of student-question pairs. (a) displays the most common hint sequences; (b), (c), and (d) show the number of times a hint type is present in a sequence, is the first in a sequence, and is the majority in a sequence, respectively.

difference in contemplation time across hint types ($p = 0.002$). Post hoc pairwise comparisons with Bonferroni correction [6] confirm that the contemplation time on planning hints (median = 14.0 minutes) is longer than debugging hints (median = 7.1 minutes, $p = 0.006$). For hint revisits, a Kruskal-Wallis H test [20] indicates a significant difference ($p = 0.009$), with Dunn’s post hoc tests using Bonferroni correction confirming more revisits for planning than for debugging hints ($p = 0.015$). Similarly, a χ^2 test [33] detects a significant difference in hint rating ($p = 0.003$), with pair-wise comparison using a Bonferroni correction confirming higher ratings for planning than for debugging hints ($p = 0.005$). Optimization hints, with smaller sample sizes (33–56), do not show any significant differences.

Help-seeking Patterns. Fig. 7 shows students’ help-seeking patterns. Nine out of ten most frequent hint sequences consist of a single hint type, with debugging hints being requested the most, followed by planning hints, while optimization hints were rarely used. When both planning and debugging hints were sought, planning hints were more likely to be requested first, aligning with metacognitive phases [39]. Notably, 43% of optimization hints (24 out of 56) were requested in isolation. Upon investigation of students’ code and reflections, these cases often belonged to high-performing students who solved without hints and then sought further improvements. However, some other students seemed to

mistakenly request these optimization hints when they needed debugging support.

Behavioral Patterns by Difficulty and Competency. Figure 8 compares help-seeking behaviors based on question difficulty (Figs. 8a and 8b) and student competency (Figs. 8c and 8d). As question difficulty increases, students request more debugging hints, while planning hint usage remains constant. In contrast, optimization hints were requested more often for easier questions (in 20 student-question pairs) than harder ones (10 pairs). Regarding student competency, higher-competency students request more hints overall, particularly planning hints, compared to their lower-competency peers.

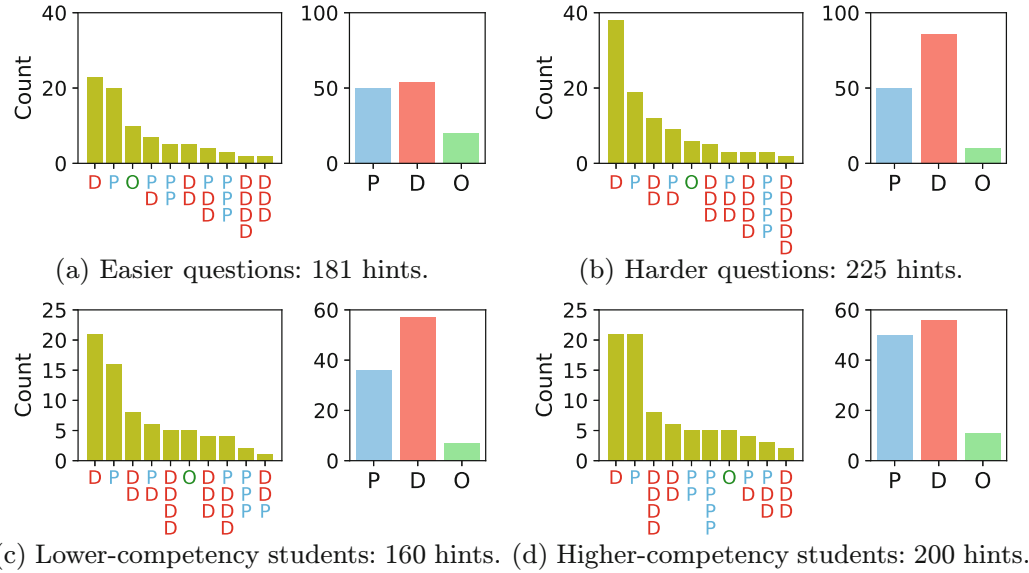


Fig. 8. Results for RQ1c: Behaviors by difficulty and competency. In each subfigure, the left shows Hint sequence and the right shows Type present. (a) and (b) present results by difficulty while (c) and (d) present results by competency.

4.3 Discussions

Our findings reveal key patterns in students' help-seeking behavior and hint usage. The high engagement and positive perception of planning hints suggest that structured guidance at the planning stage can be highly beneficial. However, it remains overlooked in many existing feedback systems, which primarily focus on debugging support [30, 35]. By emphasizing planning, educators and AI systems could better scaffold students' problem-solving processes and reduce inefficient trial-and-error cycles. Despite the high perceived value of planning hints, students requested debugging hints more often, especially for harder questions. This suggests reactive rather than proactive strategies [1, 17, 26], where students rely on troubleshooting rather than strategic planning. While debugging is an

essential skill, over-reliance on it may hinder deeper conceptual understanding. Future AI-assisted learning environments should promote proactive planning, encouraging students to articulate their problem-solving strategies before coding. Optimization hints, which were aimed at mastering skills rather than improving grades, were underutilized (8% of total requests). Further research is needed to make them more attractive and effective, fostering student mastery learning [5].

Students predominantly requested a single hint type per question, indicating a potential lack of metacognitive awareness—they may not always recognize when they need planning, monitoring, or evaluation support [24,41]. The variation in hint-seeking behavior by question difficulty further reinforces this issue. While debugging hints were requested more for harder questions, planning hint usage remained unchanged, even though structured planning is particularly useful for complex tasks [12]. Intelligent tutoring systems could use adaptive prompts or reflective exercises to help students better assess their difficulties [2,8]. Additionally, designing interventions that make planning more explicit—such as requiring students to draft pseudocode before coding—could help bridge the gap.

The difference in hint usage between higher- and lower-competency students may provide further insights into how metacognition contributes to learning. While some studies reported that weaker students require more help [3,4], our results show that higher-competency students requested more hints overall, especially planning hints. This may be because they are more persistent in problem-solving and thus, are more willing to engage with available support. In contrast, lower-competency peers may be more prone to give up earlier. This difference likely contributes to performance disparities, as discussed next.

5 RQ2: AIMS Hints and Performance

Building on the behavioral patterns identified in RQ1, this section addresses RQ2: how those patterns relate to student problem-solving performance.

5.1 Analysis Setup

We break RQ2 into two sub-questions: [RQ2a]: How do students’ interaction patterns with AIMS hints relate to their problem-solving performance? and [RQ2b]: How do these effects vary by question difficulty and student competency? To answer these, we focus on students’ final solving rates in relation to hint usage.

5.2 Results

Hint Usage and Overall Performance. As shown in Fig. 9a, requesting planning hints is associated with significantly higher performance than no hints ($p = 0.013$). Other types (debugging, optimization) show no significant effects.

Performance Variation by Difficulty and Competency. Across all difficulty and competency conditions, planning hints are consistently (even though not always significantly) associated with higher performance than no hints (see

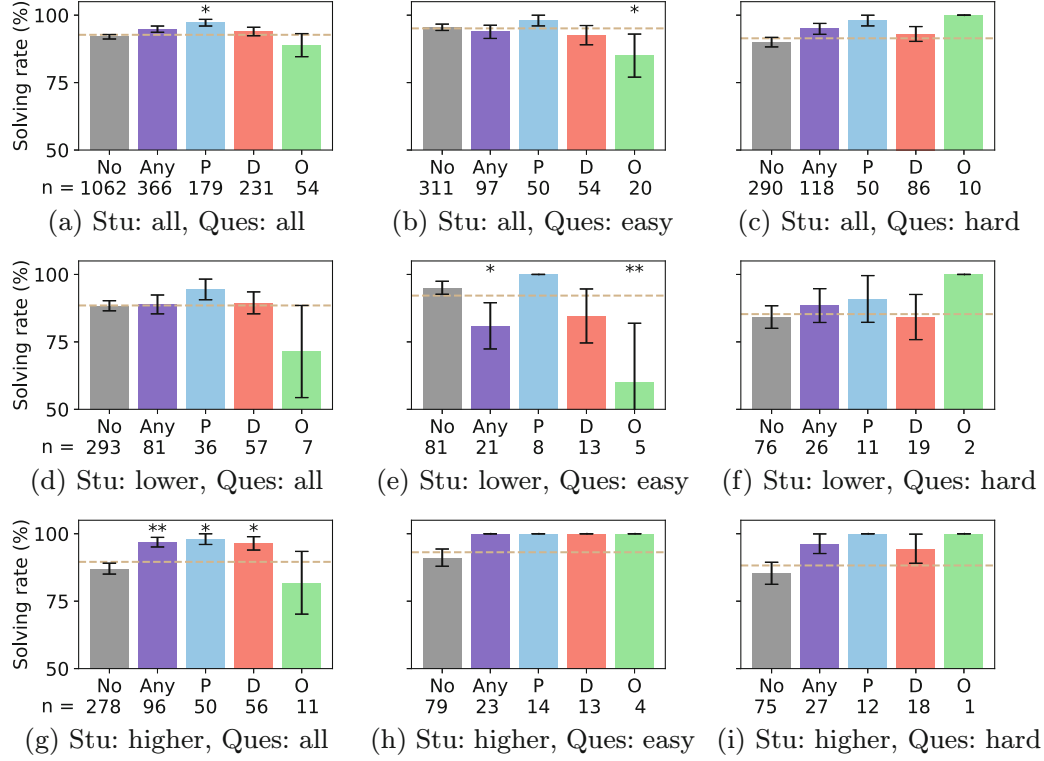


Fig. 9. Results for RQ2: Performance by help-seeking behavior. The dashed line depicts the average overall performance of the condition; the five bars represent no hint requested, any type requested, and each type present in the sequence of requested hints (i.e., Type present). The vertical lines indicate standard errors; * indicates a significant difference in performance to the ‘No’ group w.r.t a χ^2 test with $p < 0.05$, while ** indicates $p < 0.01$. Y-axes are plotted from 50%.

Fig. 9). In contrast, optimization hints are linked to lower performance on easier questions ($p = 0.039$), particularly by lower-competency students ($p = 0.003$). A closer examination of code and reflections reveals that in most of these cases, students misused optimization hints for debugging aid. Only higher-competency students, but not lower-competency ones, exhibit significantly better performance associated with hint use. Among higher-competency students, requesting any hints is linked to significantly higher performance ($p = 0.007$), with independent positive effects for planning ($p = 0.024$) and debugging hints ($p = 0.044$).

5.3 Discussions

Our findings underscore a consistent association between planning hints and higher performance. This aligns with metacognitive theory, which emphasizes planning as a critical step in problem-solving [9, 12, 16]. While the study did not establish causality (students’ high intrinsic SRL skills could be a confounding factor that caused both requesting of planning hints and higher final performance), these results suggest potential instructional value from planning hints.

In contrast, optimization hints are sometimes linked to lower performance, likely due to students' misuse, despite clear descriptions and availability of reference (see Sect. 3.3). This highlights the need for AI-assisted learning systems to ensure awareness of the available support and its alignment with their issues.

Our results differ from some studies that found hints more beneficial for lower-competency students [3, 4]. This may be because lower-competency students may have weaker SRL skills, making it harder to leverage hints effectively. Additionally, since our hints are purposely Socratic and non-direct, lower-competency students may learn less from them than higher-competency ones. Future research should explore varying hint detail levels to fit different student groups [27, 48].

6 Limitations

This work has several limitations. First, it was conducted in a single programming course, where students could make multiple submissions, resulting in high overall grades ($>92\%$). This makes it challenging to isolate the impacts of AIMS hints on student performance. Future work should examine AIMS hints in diverse courses with varying grading schemes. Second, students' under-utilization of optimization hints limited our ability to assess their impact on students. However, this highlights an opportunity for future work on strategies to encourage students to pursue mastery learning beyond correct solutions. Third, we did not measure long-term learning gains. Future research should evaluate the long-term effects using methods such as retention tests, delayed post-tests, or longitudinal tracking. Fourth, we focused solely on button-based hints. Future work could explore alternative interfaces, such as chatbots or voice assistants. Finally, we investigated AIMS hints in isolation from other forms of support. Future studies should investigate their integration with complementary support, such as instructor-led office hours, to create a more comprehensive learning environment.

7 Conclusions

This paper investigates the integration of metacognitive theory in AI-assisted programming education through a hint system aligned with planning, monitoring, and evaluation phases. By designing three corresponding hint types—planning, debugging, and optimization—and allowing students to select hints within a quota, our approach not only tailors the support to students' problem-solving stage but also fosters students' metacognitive awareness. A field study reveals that students engage most with planning hints, which are consistently linked to higher performance. However, students often request only one hint type per question and, when facing harder tasks, request more debugging but not more planning hints. This insight warrants future work on better metacognitive guidance in student awareness. Our findings provide empirical evidence of the synergy between AI and pedagogical theory in programming education, opening avenues for future research on pedagogically informed AI-tutoring systems.

Acknowledgments. Funded/Co-funded by the European Union (ERC, TOPS, 101039090). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

1. Alasmari, N.J., Althaqafi, A.S.A.: Teachers' practices of proactive and reactive classroom management strategies and the relationship to their self-efficacy. *Lang. Teach. Res.* **28**, 2158–2189 (2024)
2. Alzaid, M., Hsiao, I.h.: Effectiveness of reflection on programming problem solving self-assessments. In: *Frontiers in Education Conference (FIE)* (2018)
3. Beal, C.R., Arroyo, I.M., Cohen, P.R., Woolf, B.P.: Evaluation of AnimalWatch: an intelligent tutoring system for arithmetic and fractions. *J. Interact. Online Learn.* **9**, 64–77 (2010)
4. Beal, C.R., Waller, R., Arroyo, I., Woolf, B.P.: On-line tutoring for math achievement testing: a controlled evaluation. *J. Interact. Online Learn.* **6**, 43–55 (2007)
5. Bloom, B.S.: Learning for mastery. *Instruction and curriculum*. Regional education laboratory for the Carolinas and Virginia, topical papers and reprints, number 1. *Evaluation Comment* **1**, n2 (1968)
6. Bonferroni, C.: Teoria statistica delle classi e calcolo delle probabilita. *Pubblicazioni del R istituto superiore di scienze economiche e commerciali di firenze* **8**, 3–62 (1936)
7. Bui, G., Susanto, N., Sibia, N., Zavaleta Bernuy, A., Liut, M., Petersen, A.: Do hints enhance learning in programming exercises? Exploring students' problem-solving and interactions. In: *Proceedings of the Technical Symposium on Computer Science Education (SIGCSE)*, vol. 2 (2024)
8. Choi, H., Jovanovic, J., Poquet, O., Brooks, C., Joksimović, S., Williams, J.J.: The benefit of reflection prompts for encouraging learning with hints in an online programming course. *Internet High. Educ.* **58**, 100903 (2023)
9. Cohen, P.R., Feigenbaum, E.: Planning and problem solving. Stanford University, Department of Computer Science (1982)
10. Denny, P., et al.: Generative AI for education (GAIED): advances, opportunities, and challenges. *CoRR abs/2402.01580* (2024)
11. Ebrahimi, A., Kopec, D., Schweikert, C.: Taxonomy of novice programming error patterns with plan, web, and object solutions. *ACM Comput. Surv.* **38**, 1–24 (2006)
12. Eichmann, B., Goldhammer, F., Greiff, S., Pucite, L., Naumann, J.: The role of planning in complex problem solving. *Comput. Educ.* **128**, 1–12 (2019)
13. Fan, Y., et al.: Beware of metacognitive laziness: effects of generative artificial intelligence on learning motivation, processes, and performance. *Br. J. Educ. Technol.* **56**, 489–530 (2024)
14. Flavell, J.H.: Metacognition and cognitive monitoring: a new area of cognitive-developmental inquiry. *Am. Psychol.* **34**, 906–911 (1979)
15. Gabbay, H., Cohen, A.: Combining LLM-generated and test-based feedback in a Mooc for programming. In: *Proceedings of the Conference on Learning@ Scale (L@S)* (2024)
16. Gunzelmann, G., Anderson, J.R.: Problem solving: increased planning with practice. *Cognit. Syst. Res.* **4**, 57–76 (2003)

17. Hoffman, H.J., Elmi, A.F.: Do students learn more from erroneous code? Exploring student performance and satisfaction in an error-free versus an error-full Sas® programming environment. *J. Stat. Data Sci. Educ.* **29**, 228–240 (2021)
18. Holton, D., Clarke, D.: Scaffolding and metacognition. *Int. J. Math. Educ. Sci. Technol.* **37**, 127–143 (2006)
19. Hurst, A., et al.: GPT-4o system card. Arxiv Preprint [Arxiv:2410.21276](https://arxiv.org/abs/2410.21276) (2024)
20. Kruskal, W.H., Wallis, W.A.: Use of ranks in one-criterion variance analysis. *J. Am. Stat. Assoc.* **47**, 583–621 (1952)
21. Li, R., Che Hassan, N., Saharuddin, N.: College student’s academic help-seeking behavior: a systematic literature review. *Behav. Sci.* **13**, 637 (2023)
22. Liffiton, M., Sheese, B.E., Savelka, J., Denny, P.: CodeHelp: using large language models with guardrails for scalable support in programming classes. In: *Proceedings of the Koli Calling International Conference on Computing Education Research* (2023)
23. Lohr, D., Keuning, H., Kiesler, N.: You’re (not) My type-can LLMs generate feedback of specific types for introductory programming tasks? *J. Comput. Assist. Learn.* **41**, e13107 (2025)
24. Loksa, D., et al.: Metacognition and self-regulation in programming education: theories and exemplars of use. *ACM Trans. Comput. Educ. (TOCE)* **22**, 39 (2022)
25. Ma, B., Chen, L., Konomi, S.: Enhancing programming education with ChatGPT: a case study on student perceptions and interactions in a Python course. In: *Proceedings of the Artificial Intelligence in Education (AIED)* (2024)
26. Mallik, S., Gangopadhyay, A.: Proactive and reactive engagement of artificial intelligence methods for education: a review. *Front. Artif. Intell.* **6**, 1151391 (2023)
27. Mao, P., Cai, Z., Wang, Z., Hao, X., Fan, X., Sun, X.: The effects of dynamic and static feedback under tasks with different difficulty levels in digital game-based learning. *Internet High. Educ.* **60**, 100923 (2024)
28. Marwan, S., Jay Williams, J., Price, T.: An Evaluation of the Impact of Automated Programming Hints on Performance and Learning. In: *Proceedings of the Conference on International Computing Education Research (ICER)* (2019)
29. Mood, A.M.: *Introduction to the Theory of Statistics*. McGraw-hill (1950)
30. Pankiewicz, M., Baker, R.S.: Navigating compiler errors with AI assistance - A study of GPT hints in an introductory programming course. In: *Proceedings of the Innovation and Technology in Computer Science Education (ITiCSE)* (2024)
31. Park, E., Cheon, J.: Exploring debugging challenges and strategies using structural topic model: a comparative analysis of high and low-performing students. *J. Educ. Comput. Res.* **62**, 2104–2126 (2025)
32. parsons, R., Hao, Q., Ding, I.: exploring differences in planning between students with and without prior experience in programming. In: *American Society for Engineering Education Annual Conference & Exposition (ASEE)* (2023)
33. Pearson, K.: X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *London Edinburgh Dublin Philos. Mag. J. Sci.* **50**, 157–175 (1900)
34. Phung, T., et al.: Generative AI for programming education: benchmarking Chatgpt, GPT-4, and human tutors. In: *Proceedings of the Conference on International Computing Education Research (ICER) - Volume 2* (2023)
35. Phung, T., et al.: Automating human tutor-style programming feedback: leveraging GPT-4 tutor model for hint generation and GPT-3.5 student model for hint validation. In: *Proceedings of the International Learning Analytics and Knowledge Conference (LAK)* (2024)

36. Price, T.W., Zhi, R., Barnes, T.: Hint generation under uncertainty: the effect of hint quality on help-seeking behavior. In: *Artificial Intelligence in Education (AIED)* (2017)
37. Rum, S., Ismail, M.A.: Metacognitive support accelerates computer assisted learning for novice programmers. *J. Educ. Technol. Soc.* **20**, 170–181 (2017)
38. Saliba, L., Shioji, E., Oliveira, E., Cohney, S., Qi, J.: Learning with style: improving student code-style through better automated feedback. In: *Proceedings of the Technical Symposium on Computer Science Education (SIGCSE)* (2024)
39. Schraw, G., Moshman, D.: Metacognitive theories. *Educ. Psychol. Rev.* **7**, 351–371 (1995)
40. Shin, Y., Jung, J., Zumbach, J., Yi, E.: The effects of worked-out example and metacognitive scaffolding on problem-solving programming. *J. Educ. Comput. Res.* **61**, 073563312311744 (2023)
41. Stanton, J.D., Sebesta, A.J., Dunlosky, J.: Fostering metacognition to support student learning and performance. *CBE—Life Sci. Educ.* **20**, fe3 (2021)
42. Sun, D., Boudouaia, A., Yang, J., Xu, J.: Investigating students’ programming behaviors, interaction qualities and perceptions through prompt-based learning in ChatGPT. *Humanit. Soc. Sci. Commun.* **11**, 1–14 (2024)
43. Vieira, C., Magana, A.J., Roy, A., Falk, M.: Student explanations in the context of computational science and engineering education. *Cognit. Instruct.* **37**, 201–231 (2019)
44. Volet, S.E., Lund, C.: Metacognitive instruction in introductory computer programming: a better explanatory construct for performance than traditional factors. *J. Educ. Comput. Res.* **10**, 297–328 (1994)
45. Wang, S., Mitchell, J.C., Piech, C.: A large scale RCT on effective error messages in CS1. In: *Proceedings of the Technical Symposium on Computer Science Education (SIGCSE)* (2024)
46. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al: Chain-of-thought prompting elicits reasoning in large language models. In: *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35 (2022)
47. Wiggins, J.B., et al.: Exploring novice programmers’ hint requests in an intelligent block-based coding environment. In: *Proceedings of the Technical Symposium on Computer Science Education (SIGCSE)* (2021)
48. Xiao, R., Hou, X., Stamper, J.: Exploring how multiple levels of GPT-generated programming hints support or disappoint novices. In: *Extended Abstracts Of The Conference On Human Factors In Computing Systems (CHI)* (2024)
49. Karaoglan Yilmaz, F.G., Yilmaz, R.: Learning analytics intervention improves students’ engagement in online learning. *Technol. Knowl. Learn.*, 1–12 (2021). <https://doi.org/10.1007/s10758-021-09547-w>
50. Zamfirescu-Pereira, J., Qi, L., Hartmann, B., DeNero, J., Norouzi, N.: Conversational Programming with LLM-Powered Interactive Support in an Introductory Computer Science Course. *NeurIPS 2023 Workshop on Generative AI for Education (GAIED)* (2023)