

Scripted Reflective Prompts vs. Free Exploration in AI-Assisted Programming Learning: A Quasi-Experimental Study

Ana Paula Siqueira¹, Talita de Paula Cypriano de Souza²,
Rachel C. D. Reis¹, Patricia A. Jaques¹

¹Departamento de Informática, PPGInf, Universidade Federal do Paraná (UFPR)

²Instituto Federal de São Paulo (IFSP), Universidade de São Paulo (USP)

prof.paulasiqueira@gmail.com, talita@ifsp.edu.br
rachel@inf.ufpr.br, patricia@inf.ufpr.br

Abstract. *Large Language Models (LLMs) are increasingly integrated into programming education, raising questions about how much structure such tools require to support learning. This quasi-experimental study investigated whether scripted reflective prompting enhances learning over unstructured exploration in LLM-assisted programming. Twenty-four Brazilian high school students learned Python loops using Google Gemini under guided or free exploration. ANCOVA revealed no significant differences between conditions $F(1,21)=1.097$, $p=.307$, although both groups showed gains. Results suggest scripted scaffolding may not outperform free exploration in brief interventions, and instructional sequencing may influence learning trajectories in AI-assisted contexts.*

1. Introduction

Large Language Models (LLMs) such as GitHub Copilot, ChatGPT, and Google Gemini are rapidly transforming programming education [Wang et al. 2025]. LLMs are generative AI systems that employ deep learning and natural language processing techniques to generate human-like textual content and contextual responses [Mavundla et al. 2025]. While these tools make programming more accessible through instant code generation and explanations, students may engage superficially, i.e., copying code without understanding logic or bypassing productive struggle essential for skill development [Sok and Heng 2023].

This tension raises a fundamental question for educators: should students be allowed to explore LLMs freely, learning through trial and error, or does effective learning require structured guidance that prompts reflection and critical engagement? While scaffolding theory suggests temporary support improves learning [Wood et al. 1976], applying these principles to LLM-assisted programming remains largely unexplored. The challenge is compounded by the fact that LLMs fundamentally alter the learning environment: traditional scaffolding strategies designed for human tutors or static educational software may not transfer directly to conversational AI systems that generate verbose, contextually-adapted responses.

Scaffolding theory [Wood et al. 1976] posits that temporary, structured support enables learners to operate within their zone of proximal development (tackling problems slightly beyond their independent capability). Self-explanation theory [Chi et al. 1989]

further demonstrates that prompting learners to articulate their reasoning during problem-solving improves comprehension and knowledge transfer. These principles suggest that structured guidance during LLM interaction could prevent superficial engagement and promote deeper learning.

Classical studies show that minimally-guided instruction is less effective than explicit guidance for novices, since beginners lack the domain knowledge to direct their own learning [Kirschner et al. 2006]. The emergence of LLMs introduces additional factors to this discussion. LLM access facilitates code generation and debugging [Prather et al. 2023, Kazemitabaar et al. 2023], but unguided use can lead to superficial engagement. In a graduate robotics course, ChatGPT use improved task performance but did not translate into measurable learning: performance predicted learning only among non-users with low-to-medium prior knowledge [Brender et al. 2024]. This result suggests that unstructured AI use decouples task success from conceptual understanding.

A complementary study offered on-demand planning, debugging, and optimization hints to introductory programming students. Students who requested planning hints obtained consistently higher grades, while few students used optimization hints, indicating that novices tend toward reactive, solution-focused interactions unless a system explicitly encourages planning [Phung et al. 2025]. In both studies, however, structuring remained *optional*: students decided whether and how to use the tool. Comparisons of mandatory structuring show its benefits more directly. AI-generated Parsons puzzles produced larger learning gains and less over-reliance on the tool than complete AI-generated solutions [Hou et al. 2024]. Comparing two forms of mandatory scaffolding directly, a Socratic-Guidance tutor produced higher learning gains than a Prompt-Refinement tutor, and this advantage persisted in students' prompting strategies after the scaffolding was removed [Brender et al. 2026]. Controlled comparisons of *scripted* guidance versus free exploration in LLM-assisted programming remain scarce, particularly regarding the influence of pedagogical sequencing.

We operationalized scaffolding through scripted reflective prompts: a protocol integrating metacognitive questions at three critical stages of AI-assisted problem solving: (1) Pre-interaction: *“Analyze the code manually and formulate your hypothesis about the error before consulting the AI”*; (2) During interaction: *“Ask the AI to identify the problem. Compare its explanation with your initial hypothesis. What differences do you notice?”*; (3) Post-interaction: *“Explain the solution in your own words without looking at the AI’s response. How would you solve a similar problem independently?”*

This scripted approach prioritizes experimental control over personalization: all students receive identical prompts at identical timepoints, enabling clean comparison with unstructured exploration [Kazemitabaar et al. 2023, Zhang et al. 2025]. Adaptive scaffolding systems that adjust support based on real-time performance represent an ideal, but scripted protocols offer practical advantages, including simpler implementation, easier replication, and elimination of confounds introduced by algorithmic adaptation decisions. Furthermore, classical instructional design literature suggests that the sequence in which students experience different guidance strategies may influence learning trajectories, with gradual fading of support often recommended over abrupt transitions [Wood et al. 1976, Cheng et al. 2025].

This study addresses two **research questions**. **RQ1:** Does scripted reflective prompting lead to greater learning gains than free LLM exploration in programming? We test this through a controlled comparison of two conditions in Phase 1 of our crossover design. **RQ2:** Do sequence effects (the order in which students experience different guidance strategies) influence learning trajectories? This exploratory question examines whether starting with structured guidance before transitioning to autonomy (Guided→Free) produces different outcomes than the reverse sequence (Free→Guided), drawing on fading scaffolding principles from instructional design literature.

We employed a crossover design for two reasons. First, pedagogical equity: both groups experienced both instructional approaches, ensuring no students were denied potentially beneficial guidance. Second, exploratory investigation: the design enabled examination of whether instructional sequencing (when students receive structured guidance) affects learning trajectories. Given that Phase 1 instruction necessarily influences Phase 2 performance in learning contexts, we restricted confirmatory hypothesis testing to Phase 1 data while conducting exploratory analysis of sequence effects using Phase 2 data.

This study makes two **contributions** to the emerging field of LLM-assisted programming education. First, we address the gap identified above by providing controlled experimental evidence comparing scripted reflective prompting versus unstructured exploration under matched conditions: unlike studies where structuring was voluntary and hint-based [Brender et al. 2024, Phung et al. 2025] or embedded in the mechanics of a narrow puzzle-based task [Hou et al. 2024], our scripted protocol mandates reflection directly within open-ended conversational AI interaction, while holding tool, duration, and researcher support constant across conditions. Second, we present exploratory evidence that pedagogical sequencing may influence learning trajectories: the Guided→Free sequence showed sustained progression, whereas the Free→Guided sequence plateaued after initial gains.

2. Method

This section describes the participants, ethical procedures, experimental design, interventions, measures, and analytical approach employed to address the two research questions. The rationale for the crossover design, pedagogical equity and exploratory investigation of sequence effects, is introduced in the final paragraphs of Section 1 and described in methodological detail in Section 2.3.

2.1. Participants and Context

Twenty-six Brazilian high school students ($M_{\text{age}} = 15.96$ years, $SD = 0.40$; 92% male) from a technical program in Systems Development were initially enrolled in this study. All participants were enrolled in the second year of secondary education at a public technical school in São Paulo, Brazil. Students had completed introductory JavaScript programming but had no prior experience with Python loops, ensuring they possessed basic programming concepts while remaining novices in the target content. Participants were assigned to groups based on pre-existing classroom divisions determined by the institution's alphabetical enrollment system, resulting in a quasi-experimental design with two intact classes rather than randomized group assignment (Group A: $n = 12$; Group B: $n = 14$). Although this quasi-experimental design limits causal inference compared

to randomized trials, the institutional assignment occurred before any knowledge of the study, minimizing selection bias.

Two students from Group A were absent during the Phase 1 intervention session and therefore did not complete the guided-use activity; their data were excluded from all analyses. The final analytic sample thus comprised 24 students with complete data across all timepoints (Group A $n = 10$; Group B $n = 14$), and these are the group sizes reported throughout the remainder of this paper.

2.2. Ethical Considerations

This study was approved by the Research Ethics Committee (CAAE: 75173523.4.3001.0102) in compliance with CNS Resolutions 466/2012 and 510/2016.

2.3. Design and Timeline

This study employed a crossover quasi-experimental design conducted over 10 weeks (May–July 2025). Figure 1 illustrates the experimental design and timeline. The timeline was structured as follows:

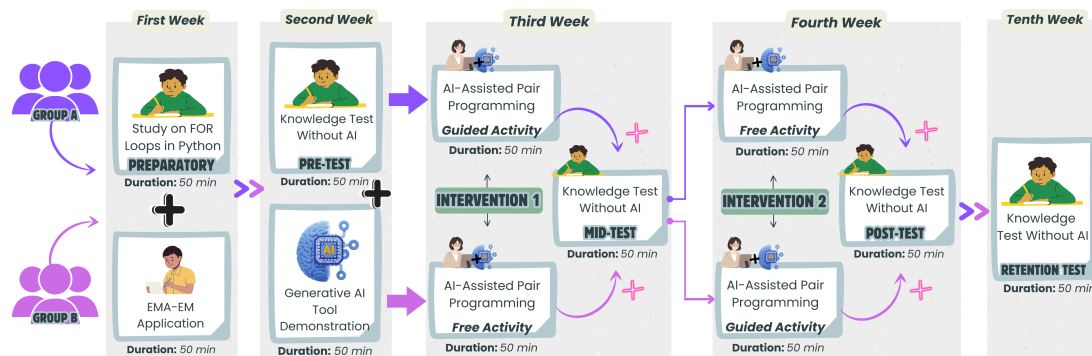


Figure 1. Experimental design and timeline. Group A ($n = 10$) experienced guided use in Phase 1 and free use in Phase 2 (Guided→Free sequence). Group B ($n = 14$) experienced the reverse order (Free→Guided sequence). Confirmatory analyses focus on Phase 1 data only.

Week 1 (Preparatory phase): Students received two 50-minute lessons introducing Python loop structures (`loops`), providing initial familiarity with the concepts and syntax before experimental activities.

Week 2 (Baseline assessment): Pre-test measured initial knowledge of loop structures. Subsequently, all participants attended a demonstration session where the researcher explained how to access and use Google Gemini integrated into Google Colab, including code generation, error explanation, and conceptual questioning.

Week 3 (Phase 1 intervention): Group A ($n=10$) completed programming exercises using guided use with scripted reflective prompts, while Group B ($n=14$) used free LLM exploration. Immediately after the intervention, all participants completed an intermediate test (Mid-test) to assess learning gains.

Week 4 (Phase 2 intervention): Groups switched conditions: Group A received free use and Group B received guided use, followed by a post-test. Sum-of-periods analysis confirmed that learning in Phase 1 carried forward to Phase 2 ($t(22) = 2.74, p = .012$), as expected in educational interventions. We therefore restricted confirmatory hypothesis testing to Phase 1 data, treating the study as a parallel-groups design, while conducting exploratory sequence analyses using Phase 2 data.

2.4. Interventions

Both groups participated in a single 45-minute intervention session using Google Gemini integrated into Google Colab. Two parallel versions of intervention materials were developed with different but equivalently challenging exercises, distributed across groups to prevent practice effects. The interventions differed in the number of exercises and the level of guidance provided for LLM interaction.

Free Use (Control condition - Group B, 4 exercises): Students received minimal instruction: “Use Gemini however you find most helpful to complete the exercises.” They had complete autonomy to formulate prompts, interpret AI responses, and decide when and how to use the tool. The four exercises included: counting consonants in user input, generating temperature conversion tables, creating a temperature guessing game, and implementing access code validation. No structured guidance or metacognitive prompts were provided.

Guided Use (Treatment condition - Group A, 3 exercises): Students completed three exercises, each structured with multiple parts incorporating scripted reflective prompting. This protocol operationalized self-explanation theory [Chi et al. 1989] through systematic metacognitive questioning designed to guide (not prescribe) the reasoning process at three critical stages:

Before AI consultation: Students first analyzed problems independently and formulated hypotheses (e.g., “Trace the code execution mentally. What pattern will it generate? Perform a desk check and draw what you expect to be printed.”).

During AI interaction: Structured prompts guided comparison between student hypotheses and AI explanations (e.g., “Now ask the AI to explain how nested loops work in this code. Compare with your hypothesis. What differences do you notice?”).

After receiving AI response: Students articulated understanding without AI assistance (e.g., “Explain the solution in your own words without looking at the AI’s answer. How would you solve a similar problem independently?”).

The three guided exercises progressively increased in complexity: (1) basic `for` loop concepts with the `range()` function and simple iteration patterns, (2) nested loops for creating two-dimensional patterns (triangles, checkerboards), and (3) code optimization for calculating mathematical series with efficiency considerations. Although the guided condition had fewer exercises (3 vs. 4), each exercise contained multiple parts with extensive reflective prompts, resulting in comparable total engagement time across conditions. This design aimed to prevent superficial copy-paste behavior by requiring cognitive engagement before, during, and after each AI consultation.

To maximize comparability despite differing exercise content, both conditions were standardized on all procedural dimensions: total session duration (50 minutes: ap-

proximately 5 minutes of instructions followed by 45 minutes of activity), physical environment (same computer lab with identical technical configuration), tool (Gemini integrated into Google Colab), and researcher support protocol (technical clarification only, without assistance in solving the programming problems).

Exercise content was designed and reviewed for comparable difficulty and complexity across conditions by the research team, who are experienced programming instructors. The pre/mid-test instruments (Section 2.5) underwent additional independent validation, given their role in measuring the study's primary outcomes.

2.5. Measures

Learning outcomes: Programming performance was assessed through pre-test and intermediate test (Mid-test), each scored on a 0–100 scale. Both tests contained four programming problems focused on loop structures (for loops), conditionals, and accumulators, varying in difficulty across three levels.

Test structure: Each test included two Level 1 exercises (basic iteration tasks, weighted 2 points each), one Level 2 exercise (intermediate complexity with conditional logic, weighted 3 points), and one Level 3 exercise (advanced nested loops or pattern generation, weighted 3 points). Final scores were calculated as weighted averages of the four exercises, allowing more complex problems to have greater influence on total scores.

Scoring rubric: Each exercise was evaluated across three components (input handling, processing logic, and output formatting). Processing logic typically received the highest weight (50–70% of exercise points), with input and output splitting the remainder based on problem requirements. A 10% deduction from the exercise total was applied for each identified error or incomplete implementation.

Parallel versions: Two equivalent test versions were developed with different but equally challenging problems, distributed across groups to prevent practice effects. For example, Group A's pre-test included "ascending count" and "sum of odd numbers," while Group B's pre-test featured "even number sequence" and "average calculation" (conceptually similar tasks requiring identical programming constructs). All instruments were validated by three experienced programming instructors to ensure appropriate difficulty and equivalence across versions.

2.6. Analysis

Primary analysis (RQ1): We conducted ANCOVA with Mid-test as the dependent variable, Pre-test as covariate, and Group (Guided vs. Free) as the independent variable. This approach isolated the intervention effect while controlling for baseline differences. ANCOVA assumptions were verified: homogeneity of regression slopes (interaction term: $F < 1$, $p = .89$), normality of residuals (Shapiro-Wilk test: non-significant for both groups), and homogeneity of variance (Levene's test: non-significant). Effect sizes were calculated using Cohen's d for between-group comparisons and partial η^2 (η_p^2) for ANCOVA effects.

Independent samples t -tests verified baseline equivalence between groups on Pre-test scores, ensuring observed Mid-test differences could be attributed to the intervention. Paired t -tests examined within-group learning gains from Pre-test to Mid-test.

Exploratory analysis (RQ2): Trajectory plots visualized learning progressions across all three timepoints (Pre-test→Mid-test→Post-test) separated by sequence: Guided→Free versus Free→Guided. An exploratory ANCOVA on Post-test controlling for Mid-test examined sequence effects. Given the identified carry-over, these Phase 2 findings are hypothesis-generating rather than conclusive. Significance level: $\alpha = .05$ for all tests.

3. Results

This section first establishes baseline equivalence between groups and reports within-group learning gains, before presenting the primary confirmatory analysis addressing RQ1 and the exploratory sequence analysis addressing RQ2.

3.1. Baseline Equivalence

Groups were equivalent at baseline on programming knowledge. Pre-test means were: Guided group ($M = 53.30$, $SD = 24.92$, $n = 10$) and Free group ($M = 35.07$, $SD = 29.04$, $n = 14$). An independent samples t -test showed no significant difference between groups, $t(22) = -1.65$, $p = .114$, Cohen's $d = 0.67$. This non-significant result, combined with ANCOVA's statistical control for baseline differences, supports valid comparison of intervention effects.

Both groups showed within-group improvement from pre-test to intermediate test. The Free group demonstrated significant learning gains (paired t -test: $t(13) = -1.87$, $p = .042$ one-tailed, Cohen's $d = 0.37$). The Guided group showed numerically similar gains that did not reach statistical significance ($t(9) = -0.88$, $p = .201$ one-tailed, Cohen's $d = 0.29$), likely due to the smaller sample size and lower statistical power. Table 1 presents the descriptive statistics for both groups across these testing phases.

Table 1. Descriptive statistics for Pre-test and Mid-test scores by condition.

Condition	Pre-test M (SD)	Mid-test M (SD)	n
Free Exploration	35.07 (29.04)	44.07 (17.82)	14
Scripted Prompting	53.30 (24.92)	60.40 (24.49)	10

3.2. RQ1: Phase 1 Effect (Primary Analysis)

ANCOVA results revealed no significant difference between guided and free LLM use on intermediate test performance (Table 2).

Table 2. ANCOVA Results for Phase 1 (Mid-test as Dependent Variable).

Source	F	df	p	η_p^2
Pre-test (covariate)	14.27	(1,21)	.001	.40
Group	1.10	(1,21)	.307	.050

The covariate (Pre-test) significantly predicted Mid-test scores, confirming that prior knowledge strongly influences subsequent performance. However, the Group effect was not significant ($p = .307$), indicating no reliable difference between instructional conditions.

Adjusted means (controlling for pre-test): Free = 47.74 (SE = 4.50), Guided = 55.27 (SE = 5.37). The mean difference favoring guided use was 7.53 points, 95% CI $[-6.2, 21.3]$, representing a small effect size: Cohen's $d = 0.46$, 95% CI $[-0.36, 1.28]$.

Interpretation: The scripted reflective prompting intervention did not produce significantly greater learning gains than free exploration. **Hypothesis 1 was not supported.** The small sample size ($N = 24$) limits the statistical power to detect small effects, though the confidence interval suggests any undetected effect is likely modest.

3.3. RQ2: Sequence Effects (Exploratory)

The crossover design enabled exploratory investigation of sequence effects beyond the primary Phase 1 comparison. Because Phase 1 learning necessarily carries forward to Phase 2 in educational interventions, we treat these analyses as hypothesis-generating rather than confirmatory.

Visual inspection of learning trajectories across all three timepoints revealed divergent patterns depending on instructional sequence (Figure 2). Students who began with guided use and transitioned to free exploration (Guided→Free sequence) demonstrated consistent upward progression across all three assessments: Pre-test ($M = 53.3$) → Mid-test ($M = 60.4$) → Post-test ($M = 64.6$). In contrast, students following the reverse sequence (Free→Guided) showed initial gains from Pre-test ($M = 35.1$) to Mid-test ($M = 44.1$), but then declined in Post-test ($M = 41.9$).

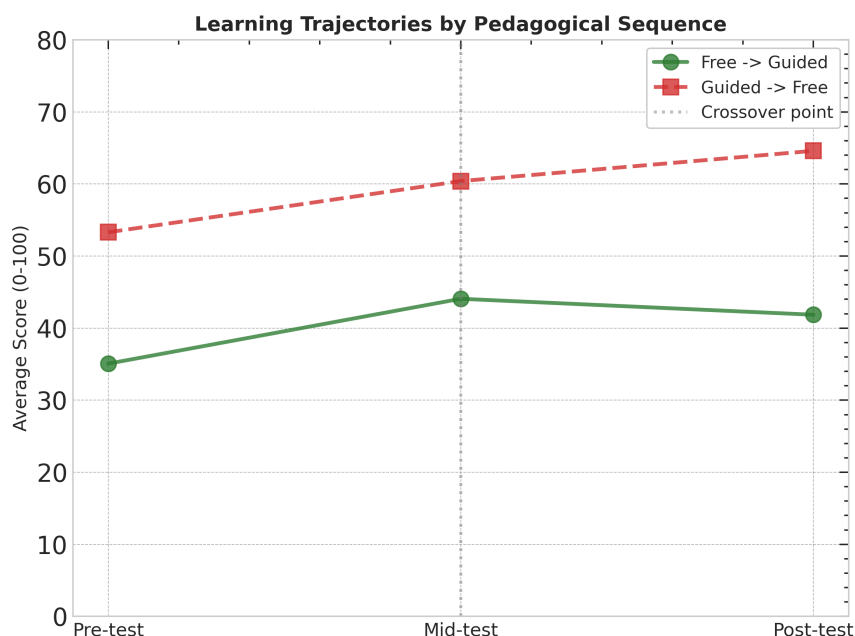


Figura 2. Learning trajectories by pedagogical sequence across three timepoints. The Guided→Free sequence (red, dashed line) demonstrated consistent upward progression, while the Free→Guided sequence (green, solid line) showed initial gains followed by decline. Vertical dashed line indicates the crossover point where groups switched conditions.

This descriptive pattern suggests that starting with structured guidance before transitioning to autonomous exploration may support more sustained learning gains than the

reverse sequence. However, these Phase 2 observations must be interpreted with extreme caution for two critical reasons:

First, the detected carry-over effect ($t(22) = 2.74, p = .012$) means Phase 2 performance was contaminated by Phase 1 learning, violating the independence assumption necessary for causal inference in crossover designs. The Guided→Free group entered Phase 2 with substantially higher Mid-test scores ($M = 60.4$) than the Free→Guided group ($M = 44.1$), creating unequal starting points.

Second, the small sample size ($n = 10\text{--}14$ per sequence) provided insufficient statistical power to reliably detect sequence effects even if they exist. Formal statistical tests on Phase 2 data yielded non-significant results (all $p > .05$), which could reflect either true absence of effects or Type II error due to low power.

Interpretation: These trajectory patterns are presented as **hypothesis-generating observations**. The visual divergence is consistent with “fading scaffolding” theory (the principle that starting with structured support and gradually releasing it promotes better learning than beginning with full autonomy). Confirmation requires replication with larger samples using between-subjects designs or adequate washout periods.

4. Discussion

This section interprets the null primary finding in light of theoretical and methodological explanations, discusses the exploratory sequence effects and their implications for instructional design, and situates these findings within the broader literature on AI-assisted programming education.

4.1. What Explains the Null Effect of Scripted Scaffolding?

The null finding contradicts common assumptions about scaffolding in educational technology. We consider four non-mutually exclusive explanations. First, LLMs generate verbose, information-dense responses; adding structured metacognitive prompts may have exceeded students’ working memory capacity, imposing extraneous cognitive load rather than facilitating deeper processing [Kirschner et al. 2006, Prather et al. 2023]. Second, our one-size-fits-all scripted approach prioritized experimental control over adaptive support, likely creating mismatches where high-knowledge students found prompts redundant while struggling students needed more targeted guidance [Kirschner et al. 2006, Jin et al. 2024]. In contrast, free exploration enabled implicit self-regulation based on immediate feedback.

Third, implementation fidelity was not directly measured, which limits how the null finding can be interpreted. Students may have engaged with the reflective prompts to different degrees, and without process data, such as interaction logs or observation of student behavior during the session, we cannot establish how consistently students in the guided condition engaged with the metacognitive questions rather than completing them with minimal effort. This distinction is relevant, since self-explanation effects depend on the quality of self-generated elaborations [Chi et al. 1989]: if some students did not fully internalize the reflective prompts, the guided condition may have been less distinct from free exploration than the design intended. Future work should incorporate systematic process measures to confirm that scripted interventions are implemented as intended before attributing null findings to the scaffolding design itself.

Fourth, with $N = 24$, the small sample limits ability to detect medium effects ($d = 0.5$), raising Type II error risk. The effect estimate (Cohen's $d = 0.46$, 95% CI $[-0.36, 1.28]$) suggests any true effect may range from small-negative to large-positive, requiring larger samples for precision.

These findings diverge from studies documenting benefits associated with AI-related structuring: puzzle-based scaffolding embedded directly into task mechanics reduced over-reliance and improved learning gains [Hou et al. 2024], while unstructured or purely optional engagement with AI assistance was associated with weaker translation from performance to learning [Brender et al. 2024, Phung et al. 2025].

A key methodological distinction may explain this divergence: those studies either embedded structure into mechanics that mechanically prevent full-solution copying (puzzle formats), or measured spontaneous, self-selected engagement with optional support, whereas our scripted protocol relied on voluntary compliance with reflective prompts layered on top of an otherwise unconstrained conversational interaction. Structural constraints that make superficial engagement impossible, or measures of who chooses to engage deeply, may be more informative than metacognitive prompts students can complete perfunctorily, particularly in brief, single-session interventions where the incentive to engage deeply with mandated-but-unenforced reflection is low. This interpretation aligns with, and extends, our third explanation above regarding unmeasured implementation fidelity.

4.2. What Do Sequence Effects Suggest?

Exploratory trajectory analysis revealed a distinct pattern despite the null primary finding: Guided→Free sequences showed consistent upward progression, while Free→Guided sequences declined after initial gains. This pattern is consistent with *fading scaffolding* principles, according to which starting with structured support and gradually releasing it produces better learning than beginning with full autonomy [Wood et al. 1976, Cheng et al. 2025]. Guided use may teach productive AI interaction strategies, such as clear prompting and critical evaluation, that transfer to subsequent autonomous learning, while starting with free exploration may establish superficial copy-paste habits that are harder to correct later [Jin et al. 2024, Paludo and Montresor 2024]. This interpretation is consistent with recent evidence that guided AI use shapes subsequent independent use: students who received Socratic-Guidance scaffolding in a graduate robotics course were more likely to adopt understanding-driven prompting strategies once the scaffolding was removed, compared to students who received a less reflective form of guidance [Brender et al. 2026]. However, the carry-over effect detected in Phase 2 ($t = 2.74$, $p = .012$) indicates that Phase 2 data are contaminated, which prevents strong causal claims. These observations are hypothesis-generating and require replication with a between-subjects design.

5. Conclusion

This quasi-experimental study ($N = 24$) found no significant difference between scripted reflective prompting and free exploration in LLM-assisted programming (ANCOVA: $F(1, 21) = 1.10$, $p = .307$, $d = 0.46$, 95% CI $[-0.36, 1.28]$), yet both groups demonstrated within-group learning gains (Free: $d = 0.37$, $p = .042$; Guided: $d = 0.29$,

$p = .201$). Exploratory trajectory analysis revealed that students who began with guided instruction before transitioning to autonomous use maintained consistent upward progression across all timepoints, while the reverse sequence showed decline after initial gains. This pattern aligns with fading scaffolding principles, though the detected carry-over effect ($t(22) = 2.74, p = .012$) necessitates interpreting Phase 2 findings as hypothesis-generating rather than conclusive.

The null primary finding likely reflects insufficient statistical power ($N = 24$) to detect medium effects, as suggested by the wide confidence interval spanning small-negative to large-positive values. Alternative explanations include cognitive overload from combining verbose LLM responses with structured prompts, mismatches between scripted support and individual learner needs, and implementation fidelity issues.

These findings challenge assumptions that structured guidance automatically outperforms exploratory learning with generative AI. Traditional programming pedagogies may not transfer directly to AI-assisted contexts where instant, verbose explanations fundamentally alter the scaffolding landscape. When an LLM can provide detailed, contextually adapted responses on demand, the added cognitive demand of structured metacognitive prompts may compete with rather than complement the learning process, particularly in brief single-session interventions with novice learners.

5.1. Limitations and Future Directions

Several limitations constrain the interpretation of these findings. The small sample size limited power to detect effects in the small-to-medium range suggested by the confidence interval, and non-randomized assignment based on intact classrooms limits causal inference. Generalizability is restricted to similar contexts: a single public technical school, a predominantly male student population, one programming topic (Python loops), and a single intervention session. Implementation fidelity was not formally measured, so the extent to which students in the guided condition engaged with the reflective prompts remains uncertain. Finally, a carry-over effect contaminated Phase 2 data, restricting confirmatory analyses to Phase 1.

Future work should address these limitations through several avenues. Replication with larger samples using between-subjects designs would enable confirmatory sequence-effect analyses without carry-over contamination. Process data, including screen recordings, interaction logs, and prompt quality assessment, would clarify whether implementation fidelity explains the null finding. Investigating cognitive load mechanisms, for instance through dual-task paradigms or physiological measures, could determine whether verbose LLM responses combined with scripted prompts overwhelm working memory in novice learners. Finally, adaptive scaffolding systems that adjust support dynamically based on individual learner knowledge and interaction patterns represent a promising avenue beyond the uniform scripted approach employed here.

Declaration on the Use of Artificial Intelligence

Claude (Anthropic) was used as an auxiliary tool for text revision, style improvement, translation into English, and formatting support in Overleaf. All AI-assisted text was reviewed and validated by the authors, who assume full responsibility for the final content.

Referências

- Brender, J., El-Hamamsy, L., Mondada, F., and Bumbacher, E. (2024). Who's helping who? when students use ChatGPT to engage in practice lab sessions. In *Proceedings of the International Conference on Artificial Intelligence in Education (AIED)*, pages 235–249, Cham, Switzerland. Springer Nature.
- Brender, J., El-Hamamsy, L., Uittenhove, K., Perez, A., Jermann, P., Mondada, F., and Bumbacher, E. (2026). Reflective dialogue or prompt refinement? effects of tutor scaffolding on students' independent llm use for programming. In Blanchard, E. G., Chen, G., Chi, M., and Isotani, S., editors, *Artificial Intelligence in Education. AIED 2026*, volume 16584 of *Lecture Notes in Computer Science*. Springer, Cham.
- Cheng, G. et al. (2025). Integrating a scaffolding-based, LLM-driven chatbot into programming education: A university case study. In *Proceedings of the 2025 International Symposium on Educational Technology (ISET)*, pages 1–5. IEEE.
- Chi, M. T., Bassok, M., Lewis, M. W., Reimann, P., and Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, 13(2):145–182.
- Hou, X., Wu, Z., Wang, X., and Ericson, B. J. (2024). Codetailor: Llm-powered personalized parsons puzzles for engaging support while learning programming. In *Proceedings of the Eleventh ACM Conference on Learning@ Scale*, pages 51–62.
- Jin, H. et al. (2024). Teach AI how to code: Using large language models as teachable agents for programming education. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI)*, pages 1–28.
- Kazemitabaar, M. et al. (2023). Studying the effect of AI code generators on supporting novice learners in introductory programming. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI)*, pages 1–23.
- Kirschner, P. A., Sweller, J., and Clark, R. E. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist*, 41(2):75–86.
- Mavundla, K., Abayomi, A., Adetiba, E., Olaitan, O., and Thakur, S. (2025). The transformative influence of generative AI on teaching and learning. In González Vallejo, R., Moukhliiss, G., Schaeffer, E., and Paliktzoglou, V., editors, *The Second International Symposium on Generative AI and Education (ISGAIE 2025)*, volume 262 of *Lecture Notes on Data Engineering and Communications Technologies*, Cham, Switzerland. Springer.
- Paludo, G. and Montresor, A. (2024). Fostering metacognitive skills in programming: Leveraging AI to reflect on code. In *Proceedings of the 2nd International Workshop on Artificial Intelligence Systems in Education (AIxEDU)*, volume 3879 of *CEUR Workshop Proceedings*, Bolzano, Italy.
- Phung, T., Choi, H., Wu, M., Singla, A., and Brooks, C. (2025). Plan more, debug less: Applying metacognitive theory to AI-assisted programming education. In *Proceedings of the International Conference on Artificial Intelligence in Education (AIED)*, pages 3–17, Cham, Switzerland. Springer Nature.

- Prather, J. et al. (2023). The robots are here: Navigating the generative AI revolution in computing education. In *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE-WGR)*, pages 108–159.
- Sok, S. and Heng, K. (2023). ChatGPT for education and research: A review of benefits and risks. *SSRN Electronic Journal*.
- Wang, Z., Zou, D., Zhang, R., Lee, L.-K., Xie, H., and Wang, F. L. (2025). ChatGPT-enhanced self-regulated learning in programming education: Impacts on motivation, self-efficacy, and learning outcomes. *Interactive Learning Environments*, 34(5):3041–3066.
- Wood, D., Bruner, J. S., and Ross, G. (1976). The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry*, 17(2):89–100.
- Zhang, L. et al. (2025). Effects of ChatGPT-based human–computer dialogic interaction programming activities on student engagement. *Journal of Educational Computing Research*, 63(4):988–1023.