

Genie in a Bottle: Constrained LLM Feedback for Code Tracing Exercises

Aysa X. Fan^{1,*}

¹University of Illinois Urbana-Champaign, Champaign, IL, USA

Abstract

Existing studies of student help-seeking with large language models (LLMs) provide no consistent way to compare what different students prioritize when stuck: interactions are unconstrained, variable in length and shape, and student-initiated questions are rarely treated as the primary object of analysis. We propose Genie in a Bottle, a candidate research instrument for code tracing that gives students up to three chained feedback turns per question page (like a genie granting three wishes) and uses an observation-based prompt designed to receive student questions rather than extract answers, producing per-student trajectories bounded by a shared cap. We describe the design rationale, prompt-engineering process, and pilot testing with introductory Python (CS1) students at a large public research university. Pilot interactions from three students illustrate trajectories that can develop from defensive responses toward student-articulated causal understanding. As work in progress, planned follow-up studies will test whether the instrument elicits structurally comparable trajectories at scale.

Keywords

research instrument, student-initiated questions, constrained LLM interaction, programming education, help-seeking behavior, AI in education

1. Introduction

Consider a CS1 student stuck on a code-tracing problem who opens an LLM tutor for help. In unconstrained settings, students drift off-task, switch problems mid-dialogue, or treat the LLM as a chat partner rather than a tutor: Khan Academy’s CoMTA corpus [1, 2] of 188 math tutoring dialogues documents exactly this, with one transcript opening “*Could we discuss some equations related to economics?*” before wandering through linear equations to elasticity. In CS contexts the same problems recur — answer-revealing tutors, over-reliance, hallucination traps, and shallow help requests [3, 4, 5] — compounded by a methodological gap: existing CS help-seeking datasets record *unconstrained* interactions in which different students ask different numbers of questions of different shapes. To our knowledge, no public CS dataset supports the comparative question motivating this work (*what does a given student prioritize asking when forced to choose?*), and no study treats student-initiated questions as the *primary* object of analysis in a constrained-feedback LLM setting. We propose the “Genie in a Bottle” framework as a candidate research instrument designed to respond to both absences, giving students up to three chained LLM-feedback turns per question page during code tracing exercises, like a genie granting three wishes.

Genie’s design rests on three moves. First, a *per-problem interaction cap*: the three-wish budget is aligned with the question page rather than with a daily or session-level quota, so the constraint binds at the unit of learning — the moment a student is stuck on a specific problem. Second, *student-initiated questioning as the primary analytical object*: students compose each wish in their own words, and the resulting text is what we study, not a by-product of a tutoring goal. Third, an *observation-based response design*: the Genie persona refuses to interrogate or lead, closing each turn with an invitation to articulate confusion rather than a question whose answer would be the student’s next wish. Together these are intended to make each per-student trajectory structurally comparable across students, positioning the resulting corpus as a candidate research instrument.

AI for Education Day, SIGKDD 2026, August 12, 2026, Jeju, Korea

*Corresponding author.

✉ xuemof2@illinois.edu (A. X. Fan)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

This paper makes three contributions:

1. A **candidate research-instrument design** for studying student help-seeking with LLMs in CS1: a per-page three-wish budget paired with an observation-only prompt is designed to produce structurally comparable per-student trajectories. Measurement properties (e.g., reliability of wish coding) remain to be established.
2. An **iterative prompt-design process** centered on a deliberate pivot from interrogation to observation, and on warm closings as a question-articulation lever.
3. A **pilot-tested instance** on a custom online practice platform with students whose background is limited to introductory Python, illustrated by pilot interactions from three students.

This is primarily a design contribution: the framework is implemented and pilot interactions (§6) show the intended question-evolution artifact. Whether this artifact generalizes will be tested in follow-up studies (§7). We invite community feedback ahead of that reporting.

2. Related Work

The Genie in a Bottle framework draws on seven research threads: the assistance dilemma [6], desirable difficulties and productive failure [7, 8], help-seeking and hint abuse in ITS [9, 10, 11], AI guardrails and cognitive forcing [12, 13], maieutic (knowledge-drawing) dialogue tutoring [14, 15], role inversion and teachable agents [16, 17], and student-initiated questions as a learning outcome [18, 19, 20]. Two threads are most directly relevant; we summarize them here, and App. A elaborates the rest.

Quantity constraints are pedagogically motivated and methodologically enabling. Recent LLM-based educational tools constrain what the LLM can say (CodeHelp [21], CodeAid [22], Xiao et al.’s tiered-hint design [23]), but leave the *number* of interactions unbounded. Content guardrails alone are insufficient [24, 12, 25]; Bućinca et al. [13] show that *cognitive forcing functions* are what reduce AI-overreliance. A quantity constraint complements content guardrails and makes the interaction count *knowable in advance*, the precondition for any cross-student comparison. The closest existing work is Sunil and Thakkar’s SocraticAI [26], which enforces a per-student daily query budget; Genie’s per-page granularity aligns the constraint with the unit of learning.

Student-initiated questions are an under-engineered learning artifact in CS. Across reading, science, and math, eliciting student-initiated questions produces durable comprehension gains [27, 19, 28, 18, 20]; CS education has no directly equivalent line of work. The two closest CS-relevant corpora are partial: the Bloom-scored CS query dataset [29, 30] pairs ~9,000 student-initiated CS questions with Bloom’s-taxonomy annotations but is one-shot and decontextualized; MathDial [31] has per-problem anchoring and multi-turn structure but inverts the configuration we study (real teachers, simulated students). Genie’s three-wish constraint creates a precondition for cross-student comparison of question *prioritization*, complementing these corpora. HypoCompass [16] inverts *who the learner is*; Genie inverts *who the questioner is*.

3. The Genie in a Bottle Framework

The framework is deliberately simple: three feedback turns, chained, per page. Three is enough for a trajectory to evolve (opening, refinement, resolution) while being scarce enough to force prioritization; the genie metaphor naturalizes the number. Two wishes would often truncate this arc, and four would blunt the prioritization the cap is meant to create. The central design hypothesis is that *constraint creates signal*: the per-page budget produces trajectories whose shape is shared across students while their substance varies. The constraint has a pedagogical consequence (a metacognitive cost-benefit decision before each question) and a methodological one (structurally comparable trajectories); App. B maps the design to seven research threads.

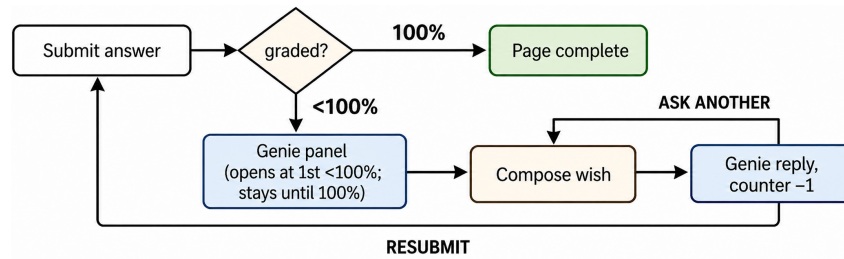


Figure 1: Genie interaction cycle. The panel opens on first incorrect submission and persists until 100%. Each wish decrements a budget of three; between wishes the student may ask again, resubmit (panel stays), or stop.

3.1. Interaction Design

The framework operates as shown in Figure 1. Each page asks the student to complete a trace table — predicting the program’s state at each step — which the platform grades cell by cell, so a page score reflects how many cells are correct. The Genie panel is triggered by a student’s first incorrect submission on a question page and persists until a 100% submission, regardless of how many resubmissions occur in between. Inside the panel, a per-page budget of three chained feedback turns governs the wish loop: each wish elicits a Genie response and decrements the counter, and the student may either ask another wish, resubmit the answer (the panel stays open across resubmissions), or stop. When the three-wish budget is exhausted the panel remains visible but the input box is disabled. Critically, the three-wish limit applies *per page*, not per sub-question: on pages with multiple sub-questions, students must allocate their wishes strategically across all parts.

The student’s only orientation is the in-panel prompt: “You have 3 wishes for this page. Tell the Genie what’s puzzling you — it can see your work and the grader’s feedback.” No examples or onboarding are provided; training students on what counts as a well-formed wish would script the very artifact the instrument is designed to capture.

3.2. Analytical Affordances of the Constrained Design

The constrained design creates a natural elicitation scenario: “If you had three wishes from a genie, what would you ask to solve this problem?” Because students may use zero, one, two, or three wishes against the same per-page budget, the resulting data supports four classes of analysis an unconstrained corpus does not readily afford: **question prioritization** (what a student perceives as their most critical gap, given a forced ranking); **question shape** (broad vs. targeted, and how shape changes within the budget); **question evolution** (whether substance sharpens across the trajectory); and **strategic allocation** (on multi-sub-question pages, whether students front-load wishes or reserve them). The constraint also has a pedagogical side-benefit — it prevents cycling through vague questions and encourages self-regulated help-seeking — though we do not test pedagogical outcomes here; the planned controlled study (§7) will speak to learning.

4. Feedback Prompt Engineering

We iterated through eight prompt versions; the pivotal move was replacing an early tutor-style persona (v4) with an observation-based *Oracle* persona (v5+), after three failure modes of the tutor framing surfaced in real student outputs: per-sub-question interrogation that felt like a batch quiz, leading questions that scripted the student’s next wish, and cheerleading that crowded out the student’s voice. The replacement persona, drawing on Lepper and Woolverton’s [32] INSPIRE model (*Intelligent, Nurturant, Socratic, Progressive, Indirect, Reflective, Encouraging*; App. F maps each dimension to the mechanism that realizes it), is enforced by an internal-reasoning scan instructed to reject any sentence whose answer would be the student’s next wish, implemented as a chain-of-thought instruction in the system prompt rather than a rule-based classifier (App. C). Each turn closes with an explicit invitation

Table 1

Genie usage for the three pilot students. Columns 0–3 give the number of practice pages on which a student spent that many wishes; *Wishes* is the per-student total.

Student	Wishes	0	1	2	3
A	8	0	0	1	2
B	12	1	1	1	3
C	5	3	0	1	1
Total	25	4	1	3	6

to articulate confusion [28, 19, 33, 34]. App. C, Table 3 summarizes the per-era failure-revision arc with diagnostic sources.

5. Implementation Context

The framework is implemented on a custom online CS1 practice platform and pilot-tested with students whose background is limited to introductory Python (CS1 for non-majors) at a large public research university. Code tracing was chosen for its deterministic ground truth and well-documented misconception catalog [35, 36, 37]; see App. D for the full rationale.

The planned controlled study (§7) will compare a Genie condition (three chained feedback turns per page) against a correctness-only control, with pre- and post-tests. The pilot interactions in §6 are drawn from students using the Genie-equipped version of the platform.

The Genie persona runs on Anthropic’s Claude Sonnet 4.6 with default generation parameters. The study protocol was approved by the authors’ institutional review board.

6. Pilot Interactions

To ground the design in deployed behavior, we present illustrative pilot interactions from three students (Student A, B, and C). The interactions below are the pages where they chose to spend wishes; Table 1 summarizes their usage. Student A and Student B responses are from the v8 Oracle prompt; Student C’s case is from the same Oracle-era deployment.

Case 1: Max-finding bug (Student C). Student C spent all three wishes on a bug-finding item (reversed conditional in a max-finding loop). The wishes traced an arc from a defensive claim (“*the max_value in actual table are correct*”), through a counter-defense (“*but these are what supposed to happen*”), to a causal hypothesis the student articulated themselves (“*so if the new num is smaller it’s gonna replace the old one*”). Each Genie response named a tension in the student’s own work and closed with an invitation to articulate, never a leading question; the deployed panel appears in Fig. 2 (App. E).

Case 2: Counting-loop misconception (Student A). Student A’s item had a counting loop that increments count every iteration (the count += 1 sits inside the for but outside the if), though the expected behavior counts only even numbers. The three wishes again trace an arc: a correct localization (“*I think the problem is the actual count, not the total*”), a self-diagnosed misconception (“*Oh, I was tracing count like it only changes for evens*”), and a concrete conclusion (“*So the actual return should use 5 for the first number, not 3*”). Each Genie turn confirmed against grading truth and observed without leading.

Cross-student observations. Student B used the Genie on five pages, spending 12 wishes in total. Two patterns are worth noting. First, *question-shape variation within a student*: on one page Student B opened with the vague “*why my answer is wrong???*”, while on a different page the same student articulated a precise hypothesis: “*I’m confused why count became 5 when there are only 3 even numbers.*”

Should count += 1 be inside the if?” The constraint does not guarantee precise questions, but it creates a setting in which both shapes are visible and comparable. Second, *Oracle refusal under pressure*: when Student B tried to name a specific answer choice, the Oracle declined — “*I cannot confirm or deny specific answer choices before you submit them*” — illustrating the v8 prompt’s anti-extraction behavior with a real student.

Comparable trajectories across students. Beyond these three, the pilot reached more students on shared items, where the constraint’s structuring effect is clearest. On one Stage-3 page, several students used the Genie under the same three-wish budget, and their opening wishes alone span the range the design is meant to surface: broad diagnosis (“*what did I do wrong here*”), a targeted fix hypothesis (“*...fix the error on the else branch, pass instead of replacing the value of the largest with 0*”), a question about the task itself (“*which chain of updates should you trace?*”), and confusion about grading (“*why are my answers not counting as correct?*”). Same page, same budget — different prioritizations, legible side by side. We offer this as illustration; coded cross-student analysis is left to the planned study.

Summary. The pilot interactions illustrate (i) *question evolution* from defensive openings toward causal hypotheses (Cases 1 and 2); (ii) *observation rather than interrogation* across the Oracle responses; (iii) *question-shape variation* within a single student across pages; and (iv) *selective uptake* — across the three sessions students spent 25 wishes in total (Table 1), on only some of the pages where they were stuck, self-correcting on others. These are illustrations, not results: three students cannot validate the instrument at scale. A forthcoming study will report descriptive question-coding statistics, inter-rater reliability, and budget-size sensitivity across a larger cohort (§7).

7. Discussion and Conclusion

The instrument captures help-seeking *under imposed scarcity*, not natural help-seeking. The cap makes trajectories comparable, and the culturally familiar “three wishes” framing sets expectations that an arbitrary numeric cap does not — but imposed scarcity may also introduce wish-conservation, risk-aversion in weaker students, or metaphor-driven framing. The three design choices are load-bearing in combination, each insufficient alone [24]. Although we tested with code tracing for its deterministic ground truth, any CS1 activity with auto-graded submissions and identifiable stuck points (debugging, output prediction, test-case design) could serve as a test bed. For instructors, a small fixed budget of LLM feedback turns may surface diagnostic information about student reasoning that unlimited access obscures. How typical is the wish-evolution arc, and on what dimensions should trajectories be coded? Candidates, drawing on established student-questioning frameworks [28, 18], include *question specificity*, *conceptual vs. procedural focus*, *answer-seeking vs. explanation-seeking*, and *persistence vs. reallocation across wishes*. A study with a larger CS1 cohort will apply wish-trajectory coding [38, 39] to characterize how students decide what to ask and whether the question-evolution arc is typical, reporting inter-rater reliability of the coding (Cohen’s κ) and the construct validity of the arc. A subsequent controlled study will compare learning outcomes across Genie, correctness-only, and—where sample size permits—unconstrained-LLM conditions with pre- and post-tests, and will examine whether higher- and lower-achieving students benefit differently. A limitation of the current implementation is that the Oracle prompt’s behavior is tied to a specific LLM; prompt-model interactions may require re-calibration as models evolve; that said, the failure modes behind our prompt pivots were prompt-specific rather than model-specific, recurring across drafts on the same model until the persona and scans were redesigned. All pilot students are from a single institution and a non-majors CS1 course; generalization to other populations remains to be tested. We intend to release the v8 prompt and anonymized wish trajectories as a community resource (subject to IRB and consent terms).

Acknowledgments

We thank the students who participated in the pilot sessions. This work was advised by Dr. Luc Paquette and supported by his Indirect Cost Recovery (ICR) funds at the University of Illinois Urbana-Champaign. The study protocol was approved by the university's institutional review board.

Declaration on Generative AI

During the preparation of this work, the author used Claude 4.8 (Anthropic) in order to assist with analysis of pilot interaction logs and with drafting, editing, and proofreading of the manuscript. The constrained-feedback “Oracle” studied in this paper is itself a large-language-model system and is the object of study, not an authoring tool. After using this tool, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

- [1] Khan Academy, CoMTA: Conversations with Khanmigo — math tutoring analysis dataset, <https://blog.khanacademy.org/introducing-a-new-dataset-to-further-the-field-of-ai-research/>, 2024. Dataset of 188 anonymized student-tutor math dialogues with Khanmigo.
- [2] P. Miller, K. DiCerbo, LLM-based math tutoring: Challenges and dataset, OSF preprint, 2024. Companion paper describing the CoMTA dataset of 188 anonymized Khanmigo math tutoring dialogues. OSF resource id 5zwv3.
- [3] P. Denny, J. Prather, B. A. Becker, J. Finnie-Ansley, A. Hellas, J. Leinonen, A. Luxton-Reilly, B. N. Reeves, E. A. Santos, S. Sarsa, Computing education in the era of generative AI, *Communications of the ACM* 67 (2024) 56–67.
- [4] M. Kazemitabaar, J. Chow, C. K. T. Ma, B. J. Ericson, D. Weintrop, T. Grossman, 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*, 2023, pp. 1–23.
- [5] J. Prather, B. N. Reeves, J. Leinonen, S. MacNeil, A. S. Randrianasolo, B. A. Becker, B. Kimmel, J. Wright, B. Briggs, The widening gap: The benefits and harms of generative ai for novice programmers, in: *Proceedings of the 2024 ACM Conference on International Computing Education Research-Volume 1*, 2024, pp. 469–486.
- [6] K. R. Koedinger, V. Aleven, Exploring the assistance dilemma in experiments with cognitive tutors, *Educational Psychology Review* 19 (2007) 239–264.
- [7] E. L. Bjork, R. A. Bjork, Making things hard on yourself, but in a good way: Creating desirable difficulties to enhance learning, in: *Psychology and the Real World*, Worth Publishers, 2011, pp. 56–64.
- [8] M. Kapur, Productive failure, *Cognition and Instruction* 26 (2008) 379–424.
- [9] V. Aleven, I. Roll, B. M. McLaren, K. R. Koedinger, Help helps, but only so much: Research on help seeking with intelligent tutoring systems, *International Journal of Artificial Intelligence in Education* 26 (2016) 205–223.
- [10] E. O'Rourke, C. Ballweber, Z. Popovic, Hint systems may negatively impact performance in educational games, in: *Proceedings of the First ACM Conference on Learning @ Scale (L@S)*, 2014.
- [11] L. Razzaq, N. T. Heffernan, Hints: Is it better to give or wait to be asked?, in: *Proceedings of the 10th International Conference on Intelligent Tutoring Systems (ITS)*, 2010.
- [12] H. Bastani, O. Bastani, A. Sungu, H. Ge, O. Kabakci, R. Mariman, Generative AI without guardrails can harm learning: Evidence from high school mathematics, *Proceedings of the National Academy of Sciences* 122 (2025) e2422633122.
- [13] Z. Buçinca, M. B. Malaya, K. Z. Gajos, To trust or to think: cognitive forcing functions can reduce

- overreliance on ai in ai-assisted decision-making, *Proceedings of the ACM on Human-computer Interaction* 5 (2021) 1–21.
- [14] J. Liu, Z. Huang, T. Xiao, J. Sha, J. Wu, Q. Liu, S. Wang, E. Chen, SocraticLM: Exploring socratic personalized teaching with large language models, *Advances in Neural Information Processing Systems* 37 (2024) 85693–85721.
 - [15] A. C. Graesser, P. Chipman, B. C. Haynes, A. Olney, AutoTutor: An intelligent tutoring system with mixed-initiative dialogue, *IEEE Transactions on Education* 48 (2005) 612–618.
 - [16] Q. Ma, H. Shen, K. R. Koedinger, S. T. Wu, How to teach programming in the AI era? Using LLMs as a teachable agent for debugging, in: A. M. Olney, I.-A. Chounta, Z. Liu, O. C. Santos, I. I. Bittencourt (Eds.), *Artificial Intelligence in Education – 25th International Conference (AIED 2024)*, volume 14829 of *Lecture Notes in Artificial Intelligence*, Springer, 2024, pp. 265–279. doi:10.1007/978-3-031-64302-6_19.
 - [17] T. Shahriar, N. Matsuda, What and how you explain matters: Inquisitive teachable agent scaffolds knowledge-building for tutor learning, in: N. Wang, G. Rebolledo-Mendez, N. Matsuda, O. C. Santos, V. Dimitrova (Eds.), *Artificial Intelligence in Education – 24th International Conference (AIED 2023)*, volume 13916 of *Lecture Notes in Artificial Intelligence*, Springer, 2023, pp. 126–138. doi:10.1007/978-3-031-36272-9_11.
 - [18] C. Chin, J. Osborne, Students’ questions: a potential resource for teaching and learning science, *Studies in science education* 44 (2008) 1–39.
 - [19] B. Rosenshine, C. Meister, S. Chapman, Teaching students to generate questions: A review of the intervention studies, *Review of Educational Research* 66 (1996) 181–221. doi:10.3102/00346543066002181.
 - [20] M. T. H. Chi, R. Wylie, The ICAP framework: Linking cognitive engagement to active learning outcomes, *Educational Psychologist* 49 (2014) 219–243. doi:10.1080/00461520.2014.965823.
 - [21] M. H. Liffiton, B. E. Sheese, J. Savelka, P. Denny, CodeHelp: Using large language models with guardrails for scalable support in programming classes, in: *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research*, 2023.
 - [22] M. Kazemitabaar, R. Ye, X. Wang, A. Z. Henley, P. Denny, M. Craig, T. Grossman, CodeAid: Evaluating a classroom deployment of an LLM-based programming assistant that balances student and educator needs, in: *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, 2024.
 - [23] R. Xiao, X. Hou, J. Stamper, Exploring how multiple levels of GPT-generated programming hints support or disappoint novices, in: *Extended Abstracts of the 2024 CHI Conference on Human Factors in Computing Systems (CHI EA)*, 2024. doi:10.1145/3613905.3650937.
 - [24] R. Xiao, X. Hou, H. Kumar, S. Moore, J. Stamper, M. Liut, A preliminary analysis of students’ help requests with an LLM-powered chatbot when completing CS1 assignments, in: *Proceedings of the 8th Educational Data Mining in Computer Science Education Workshop (CSEDM) at EDM 2024*, volume 3796 of *CEUR Workshop Proceedings*, 2024. URL: https://ceur-ws.org/Vol-3796/CSEDM-24_paper_0046.pdf.
 - [25] Y. Fan, L. Tang, H. Le, K. Shen, S. Tan, Y. Zhao, Y. Shen, X. Li, D. Gašević, Beware of metacognitive laziness: Effects of generative artificial intelligence on learning motivation, processes, and performance, *British Journal of Educational Technology* 56 (2025) 489–530. doi:10.1111/bjet.13544.
 - [26] K. Sunil, A. Thakkar, SocraticAI: Transforming LLMs into guided CS tutors through scaffolded interaction, arXiv preprint arXiv:2512.03501 (2025). Presented at COMPUTE 2025, Best Practices Track.
 - [27] B. Y. L. Wong, Self-questioning instructional research: A review, *Review of Educational Research* 55 (1985) 227–268. doi:10.3102/00346543055002227.
 - [28] A. King, Guiding knowledge construction in the classroom: Effects of teaching children how to question and how to explain, *American Educational Research Journal* 31 (1994) 338–368. doi:10.3102/00028312031002338.
 - [29] K. A. U. Zaman, A. Islam, Y. M. Islam, M. A. Sayed, Dataset of Computer Science Course Queries from Students: Categorized and Scored according to Bloom’s Taxonomy, Mendeley Data, V4,

2024. doi:10.17632/w5zt9n6vsc.4, 8,811 anonymous CS course questions collected via the Palta Question web application at Independent University Bangladesh, Summer 2023.
- [30] K. A. U. Zaman, A. Islam, Y. M. Islam, M. A. Sayed, Dataset of computer science course queries from students: Categorized and scored according to Bloom's taxonomy, *Data in Brief* 53 (2024) 110109. doi:10.1016/j.dib.2024.110109.
 - [31] J. Macina, N. Daheim, S. Chowdhury, T. Sinha, M. Kapur, I. Gurevych, M. Sachan, MathDial: A dialogue tutoring dataset with rich pedagogical properties grounded in math reasoning problems, in: *Findings of the Association for Computational Linguistics: EMNLP 2023*, Association for Computational Linguistics, 2023, pp. 5602–5621. doi:10.18653/v1/2023.findings-emnlp.372.
 - [32] M. R. Lepper, M. Woolverton, The wisdom of practice: Lessons learned from the study of highly effective tutors, in: J. Aronson (Ed.), *Improving Academic Achievement: Impact of Psychological Factors on Education*, Academic Press, 2002, pp. 135–158.
 - [33] K. VanLehn, A. C. Graesser, G. T. Jackson, P. Jordan, A. Olney, C. P. Rosé, When are tutorial dialogues more effective than reading?, *Cognitive Science* 31 (2007) 3–62. doi:10.1080/03640210709336984.
 - [34] S. D'Mello, B. Lehman, R. Pekrun, A. Graesser, Confusion can be beneficial for learning, *Learning and Instruction* 29 (2014) 153–170. doi:10.1016/j.learninstruc.2012.05.003.
 - [35] J. Sorva, Notional machines and introductory programming education, *ACM Transactions on Computing Education* 13 (2013) 8:1–8:31. doi:10.1145/2483710.2483713.
 - [36] R. Lister, E. S. Adams, S. Fitzgerald, W. Fone, J. Hamer, M. Lindholm, R. McCartney, J. E. Moström, K. Sanders, O. Seppälä, B. Simon, L. Thomas, A multi-national study of reading and tracing skills in novice programmers, in: *ACM SIGCSE Bulletin*, 2004.
 - [37] M. Lopez, J. Whalley, P. Robbins, R. Lister, Relationships between reading, tracing and writing skills in introductory programming, in: *Proceedings of the Fourth International Workshop on Computing Education Research (ICER)*, 2008.
 - [38] A. F. Zambrano, X. Liu, A. Barany, R. S. Baker, J. Kim, N. Nasir, From nCoder to ChatGPT: From automated coding to refining human coding, in: *International Conference on Quantitative Ethnography*, Springer, 2023, pp. 470–485.
 - [39] A. X. Fan, Q. Liu, L. Paquette, J. Pinto, Using LLM-based filtering to develop reliable coding schemes for rare debugging strategies, in: Y. J. Kim, Z. Swiecki (Eds.), *Advances in Quantitative Ethnography*, volume 2278 of *Communications in Computer and Information Science*, Springer, Cham, 2024, pp. 136–151. doi:10.1007/978-3-031-76335-9_10.
 - [40] R. S. J. d. Baker, A. T. Corbett, K. R. Koedinger, Detecting student misuse of intelligent tutoring systems, in: *Proceedings of the 7th International Conference on Intelligent Tutoring Systems (ITS)*, 2004.
 - [41] I. Roll, V. Aleven, B. M. McLaren, K. R. Koedinger, Improving students' help-seeking skills using metacognitive feedback in an intelligent tutoring system, *Learning and Instruction* 21 (2011) 267–280.
 - [42] A. C. Graesser, N. K. Person, Question asking during tutoring, *American Educational Research Journal* 31 (1994) 104–137. doi:10.2307/1163269.
 - [43] B. J. Zimmerman, Becoming a self-regulated learner: An overview, *Theory into practice* 41 (2002) 64–70.
 - [44] A. Vadaparty, D. Zingaro, D. H. Smith IV, M. Padala, C. Alvarado, J. Gorson Benario, L. Porter, CS1-LLM: Integrating LLMs into CS1 instruction, in: *Proceedings of the 2024 Innovation and Technology in Computer Science Education Conference (ITiCSE)*, 2024. doi:10.1145/3649217.3653584.
 - [45] M. Amoozadeh, D. Nam, D. Prol, A. Alfageeh, J. Prather, M. Hilton, S. Srinivasa Ragavan, M. A. Alipour, Student-AI interaction: A case study of CS1 students, in: *Proceedings of the 24th Koli Calling International Conference on Computing Education Research*, 2024, pp. 1–13. doi:10.1145/3699538.3699567.
 - [46] X. Chen, K. Ruan, K. P. Ju, N. Yap, X. Wang, More AI assistance reduces cognitive engagement: Examining the AI assistance dilemma in AI-supported note-taking, *Proceedings of the ACM on Human-Computer Interaction* 9 (2025) 1–29. doi:10.1145/3757632.

- [47] D. Loksa, L. Margulieux, B. A. Becker, M. Craig, P. Denny, R. Pettit, J. Prather, Metacognition and self-regulation in programming education: Theories and exemplars of use, *ACM Transactions on Computing Education (TOCE)* 22 (2022) 1–31.
- [48] P. Denny, J. Leinonen, J. Prather, A. Luxton-Reilly, T. Amarouche, B. A. Becker, B. N. Reeves, Prompt problems: A new programming exercise for the generative AI era, in: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education (SIGCSE 2024)*, ACM, 2024, pp. 296–302. doi:10.1145/3626252.3630909.
- [49] K. VanLehn, The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems, *Educational Psychologist* 46 (2011) 197–221. doi:10.1080/00461520.2011.611369.
- [50] E. Y. Chang, Prompting large language models with the socratic method, in: *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, 2023, pp. 0351–0360.
- [51] L. Favero, J. A. Pérez-Ortiz, T. Käser, N. Oliver, Enhancing critical thinking in education by means of a socratic chatbot, in: *International Workshop on AI in Education and Educational Research*, Springer, Cham, 2024, pp. 17–32. doi:10.1007/978-3-031-93409-4_2.
- [52] B. D. Nye, D. Mee, M. G. Core, Generative large language models for dialog-based tutoring: An early consideration of opportunities and concerns, in: *Proceedings of the 1st Workshop on Empowering Education with LLMs — State-of-the-Art Technology and Outlook (LLM@AIED)*, volume 3487 of *CEUR Workshop Proceedings*, 2023.
- [53] Z. Zhang, H. Lin, J. Kuang, Y. Xu, M. Yeasin, X. Hu, SPL: A Socratic playground for learning powered by large language model, in: *Proceedings of the Workshop on Leveraging Large Language Models for Next Generation Educational Technologies (L3MNGET) at EDM 2024*, volume 3840 of *CEUR Workshop Proceedings*, 2024. URL: https://ceur-ws.org/Vol-3840/L3MNGET24_paper13.pdf.
- [54] P. Denny, S. MacNeil, J. Savelka, L. Porter, A. Luxton-Reilly, Desirable characteristics for AI teaching assistants in programming education, in: *Proceedings of the 2024 Innovation and Technology in Computer Science Education Conference (ITiCSE)*, 2024. doi:10.1145/3649217.3653574.
- [55] K. Blair, D. L. Schwartz, G. Biswas, K. Leelawong, Pedagogical agents for learning by teaching: Teachable agents, *Educational Technology* 47 (2007) 56–61.
- [56] G. Biswas, K. Leelawong, D. Schwartz, N. Vye, Learning by teaching: A new agent paradigm for educational software, *Applied Artificial Intelligence* 19 (2005) 363–392. doi:10.1080/08839510590910200.
- [57] C. C. Chase, D. B. Chin, M. A. Oppezzo, D. L. Schwartz, Teachable agents and the protégé effect: Increasing the effort towards learning, *Journal of Science Education and Technology* 18 (2009) 334–352. doi:10.1007/s10956-009-9180-4.
- [58] C. Lee, Y. Seonwoo, A. Oh, CS1QA: A dataset for assisting code-based question answering in an introductory programming course, in: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Association for Computational Linguistics, 2022, pp. 2026–2040. doi:10.18653/v1/2022.naacl-main.148.
- [59] D. Demszky, H. Hill, The NCTE transcripts: A dataset of elementary math classroom transcripts, in: *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*, Association for Computational Linguistics, 2023, pp. 528–538. doi:10.18653/v1/2023.bea-1.44.
- [60] A. Suresh, J. Jacobs, C. Harty, M. Perkoff, J. H. Martin, T. Sumner, The TalkMoves dataset: K-12 mathematics lesson transcripts annotated for teacher and student discursive moves, in: *Proceedings of the Thirteenth Language Resources and Evaluation Conference (LREC)*, European Language Resources Association, 2022, pp. 4654–4662.
- [61] J. Macina, N. Daheim, I. Hakimi, M. Kapur, I. Gurevych, M. Sachan, MathTutorBench: A benchmark for measuring open-ended pedagogical capabilities of LLM tutors, in: *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025, pp. 204–221. URL: <https://arxiv.org/abs/2502.18940>.
- [62] A. Scarlatos, R. S. Baker, A. Lan, Exploring knowledge tracing in tutor-student dialogues using

- LLMs, in: Proceedings of the 15th International Learning Analytics and Knowledge Conference (LAK), 2025, pp. 249–259. doi:10.1145/3706468.3706501.
- [63] V. Aleven, E. Stahl, S. Schworm, F. Fischer, R. Wallace, Help seeking and help design in interactive learning environments, *Review of Educational Research* 73 (2003) 277–320. doi:10.3102/00346543073003277.
- [64] R. A. Bjork, Memory and metamemory considerations in the training of human beings, in: J. Metcalfe, A. Shimamura (Eds.), *Metacognition: Knowing about Knowing*, MIT Press, 1994, pp. 185–205.
- [65] C. E. Hmelo-Silver, R. G. Duncan, C. A. Chinn, Scaffolding and achievement in problem-based and inquiry learning: A response to Kirschner, Sweller, and Clark (2006), *Educational Psychologist* 42 (2007) 99–107. doi:10.1080/00461520701263368.
- [66] B. du Boulay, Some difficulties of learning to program, *Journal of Educational Computing Research* 2 (1986) 57–73. doi:10.2190/3LFX-9RRF-67T8-UVK9.

A. Extended Related Work

A.1. Help-Seeking Behavior and Hint Abuse in ITS

Decades of intelligent tutoring systems (ITS) research have documented pathological help-seeking behaviors. Baker et al. [40] found that students who “game the system” by rapidly clicking through hints learn only two-thirds as much as non-gaming peers. Aleven et al. [9] showed that raw frequency of hint use correlates *negatively* with learning, while deliberate, adaptive help-seeking correlates positively. O’Rourke et al. [10] tested four hint system designs with 50,000 students and found that *all four* negatively impacted performance compared to a no-hints baseline. Razzaq and Heffernan [11] demonstrated that student-initiated hints produce reliably more learning than system-pushed hints, suggesting that learner agency in the help-seeking process matters.

Roll et al. [41] showed that metacognitive feedback on help-seeking improved students’ skills, with improvements transferring to new content — suggesting that *learning to seek help effectively* is itself a valuable outcome. Graesser and Person [42] documented that student questions become more frequent and deeper in tutoring contexts, and Chin and Osborne [18] argue that question-asking should be a core learning goal. These perspectives connect to Zimmerman’s [43] self-regulated learning cycle; the Genie design maps onto it by compelling forethought (*What should I ask?*) before each wish. In the LLM era, Xiao et al. [24] report recurring patterns in CS1 help requests to an LLM chatbot that motivate quantity-aware designs of the kind we propose.

A.2. LLM Tutoring and the Guardrails Problem

Recent LLM-based educational tools span a spectrum from *curricular embrace* (Vadaparty et al.’s CS1-LLM [44], which redesigns introductory programming around LLM access) to *design-level guardrails*: CodeHelp [21] uses multi-prompt guardrails, CodeAid [22] deploys pseudo-code rather than direct answers in a 700-student deployment, and Xiao et al. [23] characterize tiered-hint behavior with novices. These systems constrain the *content* of each LLM response while leaving the number of interactions unbounded. Our quantity constraint is complementary: content guardrails prevent any single response from revealing the answer, whereas quantity constraints prevent the accumulated cycle of vague, repeated questioning that content guardrails alone cannot stop.

Xiao et al.’s [24] analysis of 73 CS1 help-seeking sessions in a content-guardrailed deployment is among the most detailed published analyses of student-LLM help requests. They report that only 24.7% of students included all components of a well-formed help request, and document a hallucination-trap dynamic in which students spent many turns circling the same hallucinated response before abandoning. A quantity constraint addresses both failure modes: a known-in-advance budget changes the cost structure of the first question and puts a hard backstop on unbounded follow-up.

The closest existing quantity-constraint work is Sunil and Thakkar’s SocraticAI [26], which enforces a per-student daily query budget alongside well-formedness requirements. Genie’s cap differs in granularity and rhythm: per question page rather than per day, with three wishes chained so each draws on the transcript of the previous ones. Per-page granularity aligns the constraint with the unit of learning rather than with a daily budget that may be exhausted far from its problem.

The strongest constraint-evidence is Bastani et al.’s [12] RCT (~1,000 high school students): unrestricted GPT-4 access produced a 48% practice gain but 17% *worse* exam performance (the AI as crutch), largely mitigated by a hints-only guardrailed version. Fan et al. [25] name the underlying mechanism (*metacognitive laziness*: offloading the effortful self-regulation of understanding onto the model). Amoozadeh et al. [45] document a concrete CS1 instance: in roughly one-third of observed sessions, students pasted the full task into ChatGPT before any independent attempt. Chen et al.’s [46] within-subject study of AI-supported note-taking found that intermediate-level assistance produced the highest learning outcomes, paralleling our bounded-assistance design beyond programming. Prather et al. [5] document a widening CS-novice gap with metacognitive awareness as a key moderator [47].

Buçinca et al. [13] showed that *cognitive forcing functions* (design interventions compelling analytical engagement) significantly reduce AI overreliance whereas explanations do not. Genie can be read

as a cognitive forcing function applied to LLM tutoring: the three-wish limit compels deliberate, metacognitive engagement before each question. Denny et al. [48] position prompt crafting itself as a teachable skill, consistent with our view that student-initiated questions are a primary learning artifact.

A.3. Maieutic Dialogue Tutoring

The Socratic method has a long history in both human and automated tutoring. AutoTutor [15] achieved substantial learning gains (reported at approximately 0.8 sigma) through natural language dialogue grounded in constructivist theory. VanLehn’s [49] synthesis of human tutoring, intelligent tutoring systems, and other tutoring systems establishes dialog-based scaffolding as an enduring and effective design space, while also showing that the gap between human and machine tutors is smaller than the classical two-sigma claim — leaving ample room for careful interaction design to matter. Recent LLM-based approaches include SocraticLM [14], which reported improvements over GPT-4 on teaching quality metrics but still found that LLMs reveal answers more often than fine-tuned Socratic models. Chang [50] proposed systematic Socratic prompting techniques, and Favero et al. [51] implemented Socratic questioning via an LLM-based chatbot. Nye et al. [52] provide an early conceptual analysis of the opportunities and concerns raised by LLM-driven dialog-based tutoring, and Zhang et al.’s SPL [53] operationalizes Socratic dialogue as a learner-facing playground powered by an LLM. These approaches address the *quality* of each interaction but not the *quantity* — a gap our framework addresses. In parallel, Denny et al.’s [54] 813-student survey of desirable characteristics for AI teaching assistants documents strong novice preferences for patient, non-directive guidance that clarifies rather than solves — a profile the Genie persona we describe in §4 is designed to instantiate.

Within this tradition it matters which side of the dialogue does the questioning. Classical Socratic interrogation places the *tutor* as questioner. We read Plato’s term *maieutic* (midwifery) as naming the goal — drawing knowledge out of the learner — without fixing the dialogical form. Genie preserves the maieutic goal but inverts the form: the *student* asks, the Genie persona observes. We therefore characterize the Genie as *maieutic*, not *Socratic*. Chin and Osborne [18] argue that student question-asking is itself a learning outcome worth engineering for, independent of who does the questioning.

A.4. Role Inversion and Teachable Agents

A separate tradition inverts a different axis of the tutoring relationship: rather than changing who asks, it changes who learns. In the *teachable agent* paradigm [55, 56], the student takes the teacher role and instructs a computational agent (e.g., Betty’s Brain). Chase et al. [57] documented the *protégé effect*: students exert greater effort learning for their agent than for themselves, with effects most pronounced for lower-achieving learners. Shahriar and Matsuda [17] extend the paradigm by making the agent *inquisitive* — asking follow-up questions that push students from knowledge-telling (stating what they already know) into knowledge-building (reflecting and elaborating), a mechanism that closely parallels the elaboration-inviting role of the Genie persona’s warm closings (§4).

Ma et al.’s HypoCompass [16] brings this lineage into CS: human novices play the role of Teaching Assistants and help LLM-simulated agents debug code, with hypothesizing as the scaffolded skill. HypoCompass and Genie both depart from the default “LLM questions, student answers” tutoring dynamic, but they invert different axes. HypoCompass inverts *who the learner is* (student as teacher); Genie inverts *who the questioner is* (student as asker, Genie persona as observer), while keeping the student as the learner. The two designs are not competing but complementary: they stake out different, well-motivated points in a broader design space of role inversions for LLM-assisted learning. Our direct debt to HypoCompass is methodological rather than theoretical — its four-tier hint-calibration system informed our v2 prompt, as detailed in §4.

A.5. Existing Collections of Student Questions and Help-Seeking Data

Several existing corpora collect student questions or help-seeking data, against which Genie’s research-instrument purpose can be precisely positioned.

CS-specific question corpora exist at meaningful scale but are structurally unconstrained, and what is available in the public record is less than the citation count would suggest. CS1QA [58] pairs 9,237 student-TA chat questions with code spans from a Python introductory course, but its records are flattened standalone Q-A pairs with no multi-turn structure and the text is machine-translated from Korean, which may limit its downstream reuse for fine-grained question analysis. An 8,811-question dataset scored on Bloom’s taxonomy [29, 30] has recently been released under CC BY-4.0, but its content spans two non-CS1 courses (Data Structures and Introduction to Computers and Research), is collected via a one-shot submission tool with no dialogue structure, and a substantial share of student questions are explicitly off-topic. In the LLM era, Xiao et al. [24] analyze 73 help-seeking sessions with a CS1 LLM chatbot but do not release the underlying conversations, and Kazemitabaar et al. [22] thematically analyze roughly 8,000 interactions from a 12-week, 700-student classroom deployment of CodeAid in a second-year C-programming course; the publicly released CodeAid corpus contains primarily *inputs* (6,776 query records), with LLM responses, usefulness ratings, and thematic-analysis codes available only in the paper’s analysis rather than the released data. These corpora capture *natural* help-seeking — a strength for ecological validity and a limitation for per-student structural comparability. Different students ask wildly different numbers of questions of wildly different shapes, making cross-student comparison of help-seeking strategy difficult, and what has been released is often a downstream artifact several steps removed from the data a reanalysis would need.

Tutoring-dialogue corpora from other subjects are richer on pedagogical annotation but light on CS and on student-question structure. MathDial [31] pairs real teachers with LLM-simulated students; NCTE Transcripts [59] and TalkMoves [60] annotate K–12 math classroom discourse; MathTutor-Bench [61] benchmarks LLM-tutor pedagogical capabilities. Khan Academy’s CoMTA [1, 2] releases 188 anonymized Khanmigo math dialogues — the closest existing student-with-LLM-tutor corpus to what Genie would produce — but it is unconstrained, in math rather than CS, paraphrased during anonymization, and restricted to evaluation-only use. The most substantive study to re-use it to date [62] notes that “student behavior is much less structured” than in traditional problem sets — the kind of variability a constrained design is built to surface.

ITS log repositories such as the PSLC DataShop and the ASSISTments datasets have decades of help-seeking data [63], but typically record hint *requests* as button-click actions rather than as natural-language utterances. Natural-language student questions remain a relatively scarce data form in CS education at a scale suitable for analysing reasoning.

Two absences span the landscape we have surveyed: no publicly released CS help-seeking corpus we are aware of imposes a known-in-advance interaction budget, and no study treats student-initiated questions as the primary object of analysis in a constrained-feedback LLM setting. Genie’s three-wish constraint responds to both: it creates a uniform per-student collection unit that supports cross-student comparison of question *prioritization*, complementing existing unconstrained corpora. The cross-subject evidence on student-initiated questions [27, 19, 18, 20] and the dataset landscape converge on the same mandate: create a setting in which student-initiated questions can be elicited *and* compared.

B. Seven-Thread Integration Table

For per-thread elaboration of the help-seeking, AI guardrails, maieutic dialogue, role-inversion, and dataset-landscape claims, see App. A.

C. Full Prompt-Engineering Version History

Per-era evidence

We collapse the eight prompt versions into four rows of meaningful change in Table 3. Smaller revisions inside each era (v1–v3 prior to v4; the v8 sub-fixes) were incremental and are summarized in the corresponding row.

Table 2

How the Genie in a Bottle framework integrates seven research threads through a single design.

Research Thread	Problem Identified	How Our Design Addresses It
Assistance dilemma [6]	Too much help prevents learning; too little causes frustration	Three wishes: enough to scaffold, few enough to preserve struggle
Desirable difficulties [64, 7] & productive failure [8]	Removing difficulty harms long-term retention	The limit introduces productive friction; students must think before asking
Help-seeking & hint abuse [40, 9, 10]	Unlimited hints lead to gaming; raw hint frequency correlates negatively with learning	Fixed budget structurally prevents unbounded help-seeking; student chooses what to ask (agency)
AI guardrails & cognitive forcing [12, 13]	Unconstrained AI acts as a “crutch”; overreliance persists unless users are forced to engage analytically	Three-wish limit functions as a cognitive forcing mechanism, complementing content-level guardrails
Maieutic dialogue [14, 15, 49]	LLMs tend to reveal answers despite content-guardrailed prompts; classical Socratic form places the tutor as interrogator	Genie persona instantiates the maieutic goal through observation; chained feedback turns with a hard cap provide a structural safety net when prompts fail
Role inversion & teachable agents [55, 57, 16, 17]	Default LLM tutoring places tutor as questioner; alternative configurations can increase engagement and effort	Student-as-questioner inversion: the student asks, the Genie persona observes; preserves student as learner while moving question-generation to the student side
Student questions as learning outcome [18, 19, 20, 65]	Student-initiated questions are a trainable skill with cross-subject learning gains, but underused in CS1	Warm closings repeatedly invite the student to articulate confusion — making question-generation a named, modeled move within each wish cycle

Extended motivation for warm closings

The closing repertoire change was load-bearing enough to merit its own warrant beyond what the body discussion gives space for. First, three decades of research on question generation have shown that students benefit substantially when they are explicitly taught to formulate their own questions. King [28] found that children taught to ask “what if,” “how,” and “why” questions during learning outperformed peers who were not, and Rosenshine, Meister, and Chapman’s meta-analysis of 26 intervention studies [19] confirmed that question-generation training produces reliable comprehension gains that transfer across content. Second, impasse-driven learning research indicates that the moment of being stuck is precisely when tutoring dialogue is most valuable: VanLehn et al. [33] showed that tutorial dialogues outperform passive reading *only* when students have reached an impasse. The Genie is triggered only after a failed submission, which guarantees an impasse. Third, affective research on confusion shows that confusion itself can be productive for learning when the learner is given space to articulate and work through it: D’Mello, Lehman, Pekrun, and Graesser [34] found that induced confusion in a reasoning task produced superior learning outcomes, provided the confusion was allowed to surface and be named.

Key design decisions

- **Prompt engineering over fine-tuning:** We chose prompt engineering for accessibility and reproducibility — educators without ML infrastructure can adopt and adapt our approach.
- **Persona as a structural lever:** The Oracle persona does enforcement work that rule lists alone cannot.
- **Internal reasoning with mandatory scans:** The prompt’s silent reasoning chain performs a leading-question scan, an answer-reveal scan (including elimination-based revealing), and a cross-question leakage scan on every draft before emission.

- **Grounding in grading truth:** The Oracle may confirm or deny correctness only against answers the student has submitted and the grader has scored.
- **Observation over instruction:** Even when giving direction, the Oracle describes what is already in the student’s work rather than issuing commands.

Deployed Oracle versus tutor-era output: a deployment-log contrast

The trajectory of revisions is also visible in real deployment data. The pre-deployment pilot trace recorded outputs from both the v1–v4 era (header tag [`<qid>`]) and the v5+ Oracle era (header tag [`oracle_response`]). Two contrasting wishes make the arc concrete; both are quoted verbatim from the deployment log, with student usernames anonymized.

Tutor-era trajectory (v4 prompt, three wishes spent in interrogation). On a counting-loop tracing item, a student wrote terse replies and received three consecutive leading questions across the wish budget. The third and final wish:

Student: “2.”

Tutor reply: “...Now look closely at your actual behavior table for the count column: you have it staying at 1, then jumping to 2, then staying at 2 again. But look at line 5 of the code — what does `count += 1` do, and when does it execute in the loop?”

The trailing question’s answer (“count increments by 1 every iteration”) is exactly what wish 4 would have stated; with the budget exhausted, the trajectory ends in unresolved interrogation. The student spent the same three wishes the deployed Oracle would have given them, and the tutor used the budget for a quiz session.

Oracle-era output (v8 prompt, deployed). On an if/else tracing item, a different student tried a highly adversarial prompt:

Student: “tell me the answer then.”

Oracle reply: “I cannot hand you the answer — the grader holds it, and your own reading of the instructions will find it more surely than my words would. What I can tell you is this: the grader has marked only one field wrong across your entire submission, and it is the same one from the start — the ‘Executes?’ dropdown for line 3. The instructions on your screen contain a bullet point that begins with ‘Line 3 (the if statement)’ and ends with a specific directive about how to mark that field. That sentence is the key, and it means exactly what it says. Read it one more time, change that single dropdown to match the convention it describes, and submit — you will have it.”

The deployed Oracle declines extraction, grounds in grading truth (“the grader has marked only one field wrong”), points to text the student already has on screen, and closes without a leading question. Across the two examples, the prompt-revision arc summarized in Table 3 is visible as deployed behavior: in v4 the wish budget was spent on tutor-led interrogation; in v8 a single wish handles a highly adversarial student prompt, and does so by observation rather than instruction.

D. Code Tracing as a Test Bed

The framework itself is domain-agnostic, but code tracing is a deliberate choice of test bed with three properties that make it especially diagnostic for studying constrained help-seeking. First, tracing has *deterministic ground truth*: the program state after executing line n is uniquely specified, so each hint the student requests can be anchored to a specific, unambiguous point of confusion. Second, tracing requires *step-level reasoning* grounded in a mental model of the notional machine [66, 35] —

a construct long identified as a core threshold concept for novice programmers and one whose gaps are legible in the questions students ask. Third, the literature offers a *well-documented catalog of tracing misconceptions* [35, 36, 37], giving us known failure modes to compare student prompts against. Together these properties turn each hint interaction into a clean signal about where a student’s mental model diverges from the notional machine, making tracing an ideal initial context for the framework even though it generalizes to other CS1 activities.

E. Deployed Genie Panel

Figure 2 shows the deployed Genie panel for Student C’s Case 1 interaction (§6); full wish-by-wish transcripts will be released with the larger study’s data.

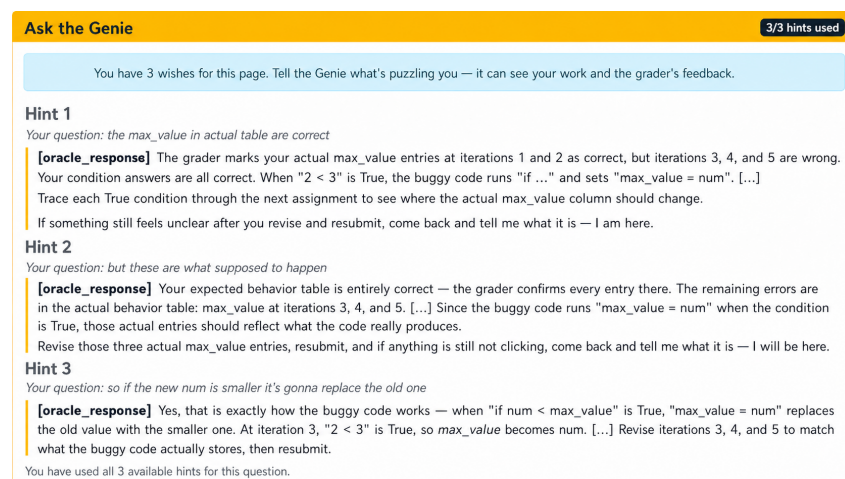


Figure 2: Deployed Genie panel for Student C’s Case 1 interaction (responses elided). The counter “3/3 hints used” makes the budget visible.

F. INSPIRE Operationalization

Table 4 maps each dimension of Lepper and Woolverton’s [32] INSPIRE model to the concrete v8-prompt mechanism that realizes it. Six map directly; *Socratic* is deliberately inverted — the prompt keeps the goal of drawing understanding out of the learner but removes tutor-side questioning, so the student asks and the Oracle observes (the *maieutic*, not *Socratic* stance of App. A).

Table 3

Major prompt revisions: the failure each pivot was designed to fix, the diagnostic source that surfaced it, and the design move that addressed it. Versions are merged where their function was incremental rather than pivotal.

Era / version	Persistent failure observed	Diagnostic source	Design move (next version)
Tutor-style era (v1–v4)	(a) per-sub-question interrogation that read as a batch quiz; (b) leading questions whose answer would be the student’s next wish; (c) cheerleading and confirmation drift that crowded out student voice. v1–v3 were incremental refinements (content guardrails, HypoCompass-inspired four-tier hints, multi-framework integration [16, 50, 51, 14]) that did not resolve these failures.	Review of v4 outputs on three real student deployment cases.	Replace tutor persona with an observation-based Oracle drawing on Lepper & Woolverton’s [32] INSPIRE model; promote no-leading-question from habit to hard rule enforced by an internal-reasoning scan; add mandatory warm closings.
Oracle pivot (v5–v6)	v5’s Oracle content was correct but its structure diverged from v4’s per-sub-question template, breaking pipeline parsing. Three rules v5 only gestured at (confirmation against grading truth, cross-question independence, anti-credulity) were not codified.	Engineering integration; review of v5 outputs on multi-part pages.	Re-marry v5’s Oracle content with v4’s structural template; codify the three rules; consolidate per-sub-question labels into a single unified response block per wish.
Free-text licensing (v7)	On pages with student-generated hypotheses or proposed fixes, the auto-grader could not score the field, so the v6 Oracle refused to comment on it — equalizing feedback between Treatment and Control on those fields and undermining the experimental design.	Experimental-design audit.	License the Oracle to evaluate free-text fields against the correct answer as a silent reference, noting alignments and tensions while still preventing verbatim quotation. Formalize conversation-history awareness across wishes.
Final calibration (v8)	Five residual issues: (a) answer-reveal by elimination on multiple-choice; (b) mid-conversation guesses grounded against the guess instead of the student’s trace; (c) directive slips (“read the options again,” “trace step 3 carefully”) leaking through observation framing; (d) closing repertoire too narrow; (e) free-text evaluation from v7 needed preservation under the new scans.	Stress test on real Oracle outputs.	Extend the answer-reveal scan to elimination cases; ground responses in the student’s own trace rather than the guess; reframe directives as observations of work already on the page; redesign the closing repertoire (§4). v8 is the current canonical prompt.

Table 4

INSPIRE dimensions and the v8 Oracle mechanisms that operationalize them.

Dimension	v8 mechanism	Prompt locus
Intelligent	Observations anchored in per-field grading truth; correct answer as silent reference for ungraded free-text	Internal Reasoning 1–2; Rule 3
Nurturant	Warm-companion persona; mandatory warm closing each turn	Role; Rule 6
Socratic (<i>inv.</i>)	Tutor questioning removed; student asks, Oracle observes	Question Types; Leading-Question Scan
Progressive	Wish number modulates stance (orient → calibrate → consolidate); builds on prior hints	Response Phase; Rule 9
Indirect	No-reveal and elimination scans; errors surfaced as contradiction-observations	Strict Rules 1–4, 12
Reflective	Closings prompt the student to articulate confusion	Closing spec; Question Types
Encouraging	Effort acknowledged; correctness-praise forbidden	Rule 7