POSTER

Overcoming Illusionary Difficulty in Novice Programming through Metacognitive Skill Enhancement

ZHEN JIANG, West Chester University, West Chester, PA, United States

JIE WU, Temple University, Philadelphia, PA, United States

NORA JIANG, Swarthmore College, Swarthmore, PA, United States

Published: 18 February 2025

SIGCSE TS 2025: The 56th ACM Technical Symposium on Computer Science Education
February 26 - March 1, 2025
PA, Pittsburgh, USA

Conference Sponsors:
SIGCSE

https://doi.org/10.1145/3641555.3705230
ISBN: 9798400705328

Overcoming Illusionary Difficulty in Novice Programming through Metacognitive Skill Enhancement

Zhen Jiang
Computer Science
West Chester University
West Chester, PA, USA
zjiang@wcupa.edu

Jie Wu
Computer & Information Sciences
Temple University
Philadelphia, PA, USA
jiewu@temple.edu

Nora Jiang
Swarthmore College
Swarthmore, PA, USA
njiang1@swarthmore.edu

Abstract

This paper investigates the illusionary difficulty novice programmers face, where students struggle with new tasks despite feeling well-prepared. The study explicitly shows that traditional assessments, such as self-reports, often fail to identify the misconceptions underlying this challenge. Using Veenman's metacognitive cueing with scaffolded self-check questions, we help students – without introducing additional materials – to refocus from their current understanding to uncovering unconscious knowledge gaps. This approach encourages students to step out of their comfort zones, engage in deeper self-exploration, persist through real difficulties, accommodate new concepts, and resolve misconceptions in their grasp. Pilot experiments reveal that while students initially overestimate their understanding, applying this cueing approach during test preparation significantly improves their actual test performance, as opposed to cumulative performance that relies on feedback and reinforcement through practices. The results underscore the importance of addressing misconceptions in problem-solving training using non-identical cues (as opposed to code evaluation training, which follows a predetermined solution path) to foster long-term self-efficacy, even if it temporarily reduces self-esteem.

ACM Reference Format:
Zhen Jiang, Jie Wu, and Nora Jiang. 2025. Overcoming Illusionary Difficulty in Novice Programming through Metacognitive Skill Enhancement. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE TS 2025), February 26-March 1, 2025, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3641555.3705230

1 Further Information

Programming is not just about code evaluation or comprehension; it's about finding a solution from scratch to tackle a range of similar tasks [2]. The former involves following a predetermined path set by computer logic, while the latter requires using a limited set of language syntax to meet the specific needs of a problem, applying code to new and unfamiliar situations. To determine whether an instruction fits a particular task, rather than merely recalling stored knowledge, it's crucial to explore new examples and expand the understanding of how this instruction can be applied.

SIGCSE TS 2025, February 26-March 1, 2025, Pittsburgh, PA, USA

ACM ISBN 979-8-4007-0532-8/25/02
https://doi.org/10.1145/3641555.3705230

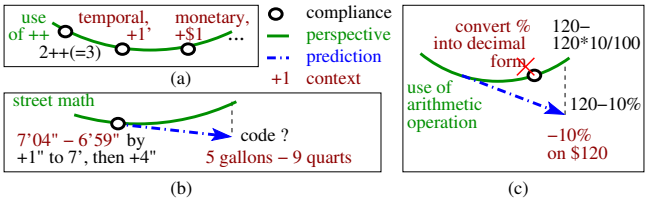

Figure 1: (a) Pragmatic view of "++" rule across varied context, (b) misconception (dash line) from incorrect understanding, and (c) misconception from incomplete knowledge.

We study students' learning through what we call *non-identical cues*, where each task is similar but not identical to previous ones. This is because programming inherently demands precision for the logical reasoning, structural comprehension, and expression within coding rules that differ from natural languages. These rules can broadly be applicable across similar tasks. For instance, as shown in Figure 1 (a), the "++" operation can increment values in a vast array of diverse real-world contexts, such as "next minute" or "earning another dollar." To master such rules, novices must apply their understanding, verify their usage in each problem-solving scenario, and adapt to new variations of the rule as needed [3], fostering deeper comprehension and greater flexibility in problem-solving.

We focus on beginners' first programming class with Python, emphasizing the initial project on arithmetic operations and the subsequent test to measure actual performance rather than cumulative performance reinforced through practice. This foundational phase is crucial, as arithmetic operations are essential in both declarative and imperative languages, enabling early skill development without the distraction of other beginner challenges (e.g., [4]). However, many students who easily memorize lecture materials often struggle to apply them to new tasks (e.g., [2]), particularly during paper-based tests where trial-and-error feedback, commonly available in homework, is absent. Our observations suggest that the primary cause is a *misconception* [1], where students develop intuitive yet incorrect understandings of the rules. Instructors expect students to easily retrieve operators and syntax practiced early, yet many novices, despite believing they are well-prepared by reviewing past programs and solutions, still underperform.

Figure 1 (b) shows a common novice misconception arising from incorrect use of arithmetic operations. Students are taught the modulus operator "%" and are expected to calculate the remaining water in a 5-gallon tank after removing 9 quarts, yielding 2 gallons and 3 quarts with the formulas "$(5*4-9)//4$" and "$(5*4-9)\%4$." Some students adopt the street math strategy, adding 3 quarts, rounding to the nearest gallon, and adding 2 gallons, but struggle to code this process without advanced concepts like loops. Many of

#	time	question	use & goal
1	before	Will a qn I don't know appear on test?	cueing
2	test	What score do I expect?	self-esteem
3	after	Has my preparation covered the test?	prep adequacy
4	test	Would I like to continue with the class?	self-efficacy

Table 1: Adopted materials – questions.

these students will resort to reverse-engineering, submitting flawed code such as "$g-3$" (= 2) and "$q/3$" (= 3) after initializing "$g, q = 5, 9$," which works for the sample values but fails with different inputs.

Figure 1 (c) shows another misconception, stemming from an incomplete understanding of core concepts. For example, calculating a 10% discount on \$120 should be written as "120 − (120∗10/100.0)" or "120∗(1 − 10/100.0)." However, students may mistakenly write "120 − 10%" if they forget to divide the percentage to convert it into a decimal. This error stems from the "%" symbol being familiar in everyday use and also valid in programming, leading students to incorrectly assume the direct translation will work.

Research problem. From a student's perspective, this type of misconception creates an *illusionary difficulty* [5], reflecting an educational paradox: students are unaware their understanding is flawed, yet feedback from computers and instructors suggests otherwise. It feels like searching in the dark for an exit – despite numerous attempts, they can't find the solution and don't understand why they can't reach it. This differs from real difficulty, where they know what needs to be done but are unable to accomplish it.

Novices without instructor supervision may turn to external aids, especially under the pressure of deadlines and academic performance. Such overreliance on aids – like AI – can hinder deep learning, as students outsource critical thinking instead of developing their own understanding, ultimately weakening personal skills and mastery [7]. As highlighted in [5], it is essential to help students recognize their struggles, *enhancing their metacognitive skills* and making learning challenges more manageable and actionable.

Our research seeks to address the following questions: (1) How can this student difficulty be identified (or missed) in traditional class assessments? (2) How can a student overcome this difficulty by enhancing cognitive skills while utilizing existing learning resources without requiring additional materials?

Methodology. Our engagement employs Veenman's metacognitive cueing approach [6], utilizing a single self-check question (see Q1 in Table 1) as scaffolding to guide students and shift their focus from past accomplishments to ongoing learning exploration aimed at reducing potential misconceptions. This method enhances their metacognitive skills in overcoming unrecognized illusionary difficulties, promoting spontaneous and continuous self-directed learning to improve test preparation and performance – all without relying on any external aid or additional learning material.

We use a traditional self-reported evaluation with survey question Q2 (Table 1) to assess if students can accurately reflect on their knowledge before the test. Failure to do so indicates unrecognized knowledge gaps, supporting evidence of misconceptions and illusionary difficulty in the metacognition. Survey question Q3 evaluates if students sufficiently explored the relevant materials for the test, helping determine their level of preparation. The effectiveness of our cueing intervention for addressing illusionary difficulty is measured by self-efficacy assessed through question Q4.

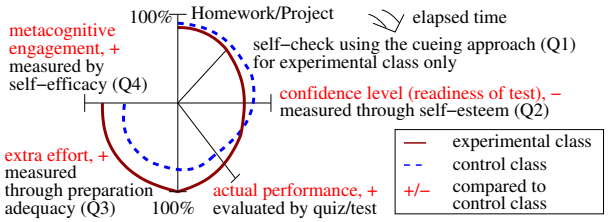

Figure 2: Intervention effects after adopting Table 1 in class.

Preliminary results and conclusion. We conducted a pilot experiment in CS students' first programming class, focusing on their first quiz and Python arithmetic operations. Two sections were randomly assigned: an experimental class (23 students) and a control class (16 students). Both followed the same procedure, except for the self-check question Q1. Homework scores were collected, and a T-test value of 0.37 confirmed fair comparison conditions.

Results (Figure 2) show that the control group's quiz scores dropped compared to student predictions based on their prior homework experience, revealing unrecognized knowledge gaps and misconceptions. To address this, the cueing approach with Q1 was used to shift the experimental group's focus during test preparation from reflecting on past achievements to ongoing exploration. While this initially reduced confidence in a self-evaluation (Q2), it improved test preparation (Q3) and ultimately led to better quiz performance. Our findings align with the principles of "delayed gratification," guiding beginners toward a better long-term self-efficacy (Q4).

The experiment answered the key research questions. First, it confirmed that students were unaware of their misconceptions, which appeared as illusionary difficulty. Novices often overestimated their abilities, reflecting the overconfidence phenomenon noted in prior literature. Traditional self-evaluations (Q2) encouraged overly positive emotions, reducing interest in further exploration and conceptual reconstruction (Q3), highlighting the need for more effective learning assessments. Second, the results presented here emphasized the importance of stepping outside comfort zones to identify knowledge gaps. Although this initially lowered confidence, the experimental class showed that the proposed scaffolding intervention effectively enhanced cognitive skills and performance, despite a temporary drop in self-esteem.

References

[1] Christophe Malaterre, Emmanuelle Javaux, and Purificación López-García. 2023. Misconceptions in Science. *Perspectives on Science* 31, 6 (2023), 717–743. https://doi.org/10.1162/posc_a_00590

[2] Anthony Robins, Janet Rountree, and Nathan Rountree. 2003. Learning and teaching programming: A review and discussion. *Computer Science Education* 13, 2 (2003), 137–172. https://doi.org/10.1076/csed.13.2.137.14200

[3] Ute Schmid, Rene Mercy, and Fritz Wysotzki. 1998. Programming by Analogy: Retrieval, Mapping, Adaptation and Generalization of Recursive Program Schemes. *Proc. of the Annual Meeting of the GI Machine Learning Group, FGML.*

[4] James Spohrer and Elliot Soloway. 1986. Novice Mistakes: Are the Folk Wisdoms Correct? *Commun. ACM* 29, 7 (1986), 624–632. https://doi.org/10.1145/6138.6145

[5] Yusuke Uegatani, Hiroki Otani, Shintaro Shirakawa, and Ryo Ito. 2023. Real and illusionary difficulties in conceptual learning in mathematics: comparison between constructivist and inferentialist perspectives. *Mathematics Education Research Journal* (2023). https://doi.org/10.1007/s13394-023-00478-6

[6] Marcel Veenman, Lieneke Kerseboom, and Cornelie Imthorn. 2000. Test anxiety and metacognitive skillfulness: availability versus production deficiencies. *Anxiety, Stress, and Coping* 13, 4 (2000), 391–412.

[7] Chunpeng Zhai, Santoso Wibowo, and Lily Li. 2024. The effect of over-reliance on AI dialogue systems on students' cognivie abilities: a systematic review. *Smart Leraning Environments* 11, 28 (2024). https://doi.org/10.1186/s40561-024-00316-7