T I E Journal
Volume 4 Issue 3, 2026
http://ijtie.com/v403/n40329
ISSN: 2820-7521

The International Journal of Technology, Innovation, & Education

Effects of a Multimodal Artificial Intelligence Tutor on Conditional Statement Achievement and Engagement

Dimitrios Kravvaris

Independent Scholar, Ioannina, Greece

jkravv@gmail.com

Received: June 18, 2026; **Accepted**: August 25, 2026; **Published**: October 31, 2026

Abstract

Conditional statements are a well-documented stumbling block for novice programmers. Beginners tend to misread the flow of control rather than trace which branch a given condition actually selects, and that habit, left alone, tends to follow them into harder material. This study reports a between-groups experiment. We examined whether a multimodal artificial intelligence (AI) tutor, one that generates visual representations of conditional execution and offers adaptive help when a student answers incorrectly, improves learning outcomes and engagement relative to traditional classroom instruction. Forty-four students aged 11 to 12 were assigned to two intact classes of 22. The first group received a standard, teacher-led lesson on if and if-else statements, built around textbook explanations and worked examples on the board. The second covered the same material in pairs, using a multimodal AI tutor that supplied step-by-step flowcharts and execution traces whenever a pair answered incorrectly, following the driver-navigator pair programming technique. Learning was measured with a pre-test and post-test on predicting the output of conditional code; engagement was gauged through a short questionnaire and classroom observation. The tutor group scored significantly higher on the post-test and reported substantially greater interest and enjoyment. These findings suggest that multimodal AI tools, used to visualize conditional execution and to scaffold corrections after errors, can lift both achievement and motivation in introductory programming.

Keywords: multimodal artificial intelligence, conditional statements, programming education, student engagement, intelligent tutoring systems

Kravvaris, D. (2026). Effects of a multimodal artificial intelligence tutor on conditional statement achievement and engagement. *The International Journal of Technology, Innovation, and Education, 4*(3), 110–129. https://ijtie.com/v403/n40329

CC BY

1. Introduction

Conditional statements decide whether a block of code runs, based on whether a condition is true. They are among the first control structures a student meets. The if statement runs a block of code only when its condition holds; the if-else statement runs one block when the condition is true and a second when it is false. The idea looks simple. Yet for beginners aged 10 to 12 it remains a stubborn source of error, because many keep reading code as a straight, line-by-line sequence instead of reasoning about which branch a condition selects (Lodge et al., 2018). The errors are familiar: assuming the body of an if always runs, that both branches of an if-else execute, or that an else fires regardless of the condition. This control-flow misconception has been tied to weaker performance in later topics, among them loops, nested conditionals, and boolean logic, which makes the shift from sequential to conditional reasoning a critical juncture in programming education (Hopcan et al., 2022).

Traditional instruction on conditionals usually rests on three things: teacher explanation, worked examples on the board, and practice exercises from a textbook or worksheet. For many students it works well enough. But it leaves little room for individualized feedback. A teacher overseeing more than twenty learners cannot trace the execution of a program for every student who slips, and the flow diagrams printed in a textbook are fixed, with no way to regenerate them for the particular snippet a given student finds hard (Lodge et al., 2018).

Multimodal AI tools offer a plausible complement (Lee et al., 2023; Qu et al., 2025). Such a tutor can present a code-reading task, generate an on-demand visual, a flowchart or a step-by-step execution trace for the snippet at hand, and, when the student answers incorrectly, produce a tailored visual and verbal explanation aimed at the specific error (Rau et al., 2013; Ng & Chan, 2022). Feedback of this kind is immediate, individualized, and visual. It is hard to deliver at scale in an ordinary classroom. Yet it sits naturally within current multimodal AI systems (Xu et al., 2025).

The purpose of this study was to compare a multimodal AI tutor against traditional, teacher-led instruction on a single, well-defined topic: conditional (if and if-else) statements. We used a between-groups design with two intact classes of students aged 11 to 12. One class received traditional instruction; the other worked with a multimodal AI tutor that generated visual representations of each conditional's execution and provided adaptive assistance after incorrect answers (Liu et al., 2025; Rau et al., 2013). This study, therefore, asks two research questions. First, does a multimodal AI tutor improve post-test performance on conditional statements relative to traditional instruction? Second, does it raise student engagement? From these

questions we derived two hypotheses. We hypothesized that (a) students in the multimodal AI tutor group would achieve significantly higher post-test scores on conditional statements than students in the traditionally taught group, and (b) students in the multimodal AI tutor group would report greater interest in and enjoyment of the lesson than their traditionally taught peers.

2. Literature Review

2.1. Difficulties in Learning Conditional Statements

Research on introductory programming has repeatedly singled out conditional statements as a useful diagnostic window into how students understand control flow (Lodge et al., 2018). A beginner working from a sequential, line-by-line mental model will often predict that both the if and the else block run, or that an if body executes even when its condition is false, overlooking that the condition is what selects the branch. Other errors recur. Some students ignore the condition altogether; others lose track of which statements the if governs; and once conditionals are nested, an else gets attached to the wrong if. Left unaddressed, these patterns carry into more advanced control structures, and they have been linked to broader trouble with logical and computational reasoning (Hopcan et al., 2022).

2.2. Visual Representations for Conditional Execution

Visual models have long been recommended for helping students' reason about control flow, since they make the path of execution perceptually available rather than purely textual (Robertson et al., 2020). Two are common: flowcharts, with a decision diamond and two outgoing paths, and step-by-step execution traces that highlight the line currently running. The flowchart is especially useful. It sets the true and false paths side by side, so that the branch actually taken, not the order of the lines on the page, becomes the basis for predicting what a program does (Ponggawa et al., 2026). But the value of any such representation hinges on one thing: whether it is generated in response to the specific code being read and the specific error a student has made, rather than offered as a generic, pre-made diagram unrelated to the task (Li et al., 2024).

2.3. Multimodal AI Tutors and Adaptive Feedback

Multimodal generative AI systems can produce text, images, and spoken explanations within a single exchange, adapting each output to a learner's input (Zhao et al., 2026). For conditional statements this matters. A tutor can build a fresh flowchart or execution trace for any snippet a student is asked to evaluate, and, when the answer is wrong, generate a further explanation, often paired with an annotated diagram, that targets the specific misconception the response

implies (Ng & Chan, 2022). Reviews of generative AI in education make a related point: adaptive, multimodal feedback may be especially valuable for topics like conditional execution, where visualizing the path through the code is central to understanding (Anderson & Boyle, 2023; Holmes et al., 2023).

Engagement and motivation count as worthwhile outcomes in their own right, partly because they accompany persistence and effort, and partly because they may mediate longer-term learning gains (Chen et al., 2022). Novel, interactive tools such as multimodal AI tutors can raise situational interest in a programming lesson, especially among students who usually find control flow frustrating (Urhahne & Wijnia, 2023). Here we examine both achievement and engagement for lessons on if and if-else statements, setting a multimodal AI tutor against traditional, teacher-led instruction.

3. Methodology

3.1. Research Design

This study used a between-groups experimental design with two intact classes (Blaney et al., 1977). Each class was assigned, as a whole, to one of two conditions: traditional instruction or multimodal AI tutor instruction. The AI tutor was developed in Google AI Studio and built on Google's Gemini 3.5 Flash multimodal model, with its design specifically aligned to the objectives of the study. Rather than supplying direct answers, the AI tutor was designed to scaffold student learning by providing supportive resources (Kravvaris & Fotou, 2025), including explanatory text and flowcharts. In order to ensure the accuracy and age-appropriateness of the generated content, the tutor was restricted to a fixed bank of ten conditional-statement tasks, and every flowchart and execution trace it could produce was reviewed by the research team before the lessons. Both groups covered the same topic, conditional (if and if-else) statements in Python programming language, in two school hours of equal duration (45 minutes each). We collected quantitative data through a pre-test and post-test on conditional statements, together with an engagement questionnaire administered after the lessons (Nia & Yaghoobzadeh, 2018). The research team also kept observation notes during both lessons.

3.2. Participants

Forty-four students aged 11 to 12 took part, organized into two intact classes of 22. Both classes were pre-existing timetabled sections at a single school, carried over from the previous school year rather than formed for this study. The two classes were assigned to conditions at random: Class 1 to the traditional instruction condition, and Class 2 to the multimodal AI tutor condition.

With respect to gender composition, Class 1 comprised 12 girls and 10 boys, and Class 2 comprised 11 girls and 11 boys. Both classes followed the same computing curriculum and had received equivalent prior instruction on variables and simple sequential statements. Neither had yet had a dedicated lesson on if and if-else statements. We obtained parental consent and student assent for all participants, in line with the ethical approval granted by the research team's institution.

3.3. Traditional Instruction Condition

Students in Class 1 received the lessons on if and if-else statements from their regular computing teacher, who followed the school's standard methodology. The lessons combined three elements: a teacher explanation of conditional logic (the structure of an if statement, the role of the condition, and the two-way branching of an if-else), worked examples on the board, and a set of practice exercises done individually on paper. The teacher drew flow diagrams on the board for selected examples. These were not, however, produced individually for each student, or for each error made.

3.4. Multimodal AI Tutor Condition

Students in Class 2 covered the same content, conditional (if and if-else) statements, but worked in pairs at shared desktop computers running the multimodal AI tutor. The exercises were provided in electronic format as a Microsoft Word document. We organized the 22 students into 11 pairs, one computer per pair. Each pair used the driver-navigator pair programming technique (Hopcan et al., 2022; Ngadengon et al., 2024): the driver held the keyboard and mouse and entered the pair's answer, while the navigator read the task aloud and checked the answer before submission. The two students traded roles after every task.

When a pair answered correctly, the AI tutor confirmed the answer briefly and moved on. When a pair answered incorrectly, it generated a step-by-step visual explanation tailored to that task. The code appeared beside a flowchart, the condition shown in a decision diamond and the true and false branches drawn as separate paths; alternatively, a highlighted execution trace showed which lines actually run, with annotations marking whether the condition evaluated to true or false and which block executed. The pair was then asked to reattempt the task using the visual before proceeding. Every visual the tutor produced had been reviewed by the research team beforehand, to confirm correctness and age-appropriateness, and the tutor was restricted to a fixed bank of 10 conditional-statement tasks spanning a range of difficulty comparable to the practice exercises used in Class 1.

3.5. Measures

A 10-item pre-test and a matched 10-item post-test on conditional statements were given to both classes, immediately before and immediately after the lessons (the full pre-test and post-test items appear in Appendix A and Appendix B, respectively). Each item presented a short code snippet using an if or if-else statement and asked students to predict the program's output, that is, which line or lines would run. Difficulty varied. Items ranged from a single if with a plainly true or false condition to if-else statements and short sequences of two separate conditions. To reduce practice effects, the pre-test and post-test used different items of matched difficulty.

After the post-test, every student completed a short engagement questionnaire (Appendix D): five items on a 4-point scale, asking how interesting, enjoyable, and clear they found the lessons, and how confident they felt about using if and if-else statements afterward. The research team also kept classroom observation notes during both lessons, attending to engagement indicators such as on-task behavior, questions asked, and visible enthusiasm or frustration.

3.6. Procedure

Both classes sat the pre-test at the start of the lessons. They then received roughly 20 minutes of instruction in their assigned condition, working through the shared bank of lesson exercises (Appendix C), and finally completed the post-test and engagement questionnaire at the end. The same regular computing teacher delivered both lessons on the same day, having received a short orientation on the multimodal AI tutor before the Class 2 lessons. We matched the content covered, the number of tasks attempted, and the total lessons duration as closely as we could across the two conditions.

4. Results

The two groups were closely matched at pre-test (Class 1: M = 4.7, SD = 1.1; Class 2: M = 4.7, SD = 1.1), and an independent-samples t-test revealed no significant difference between them at baseline. This indicates that the two groups began with comparable prior knowledge of conditional statements. Since the two classes were equivalent at pre-test, our primary between-groups comparison was carried out on post-test scores directly. Both groups improved from pre-test to post-test. The size of that improvement, however, diverged sharply.

4.1. Post-Test Achievement

Students in the multimodal AI tutor group (Class 2) achieved a significantly higher mean post-test score than students in the traditional instruction group (Class 1). The mean for Class 2 was 8.2 out of 10 (SD = 0.9), against 6.5 (SD = 1.2) for Class 1. This difference was statistically significant, $t(42) = 5.32$, $p < .001$, and corresponded to a large effect, $d = 1.60$, 95% CI [1.05, 2.35], which supports the first hypothesis. The mean gain from pre-test to post-test was likewise larger for Class 2 ($M = 3.5$) than for Class 1 ($M = 1.8$), consistent with the same pattern.

Item-level analysis showed that the largest between-group gaps surfaced on items asking students to predict the output of if-else statements and short sequences of two conditions, the item type most strongly tied to control-flow misconceptions. On these items, students in Class 2 were far more likely to answer correctly. Where they showed their work, they more often traced the condition and named the branch that would run; students in Class 1, by contrast, more often assumed that both branches execute or that an if body always runs, with the occasional slip on the condition itself. This is perhaps the most telling result of our study.

4.2. Engagement and Attitudes

On the post-lesson engagement questionnaire, students in Class 2 reported substantially higher interest and enjoyment than those in Class 1 (Class 2: $M = 3.5$, $SD = 0.4$; Class 1: $M = 2.6$, $SD = 0.6$). This difference in mean engagement rating was statistically significant, $t(42) = 5.85$, $p < .001$, $d = 1.77$, which supports the second hypothesis. A large majority of Class 2 students rated the lessons interesting and enjoyable, and felt more confident about using if and if-else statements afterward. Class 1 responses were more mixed. A sizable share called the lessons neutral, or somewhat boring.

The classroom observation notes squared with the questionnaire. Pairs in Class 2 were often seen discussing the generated flowchart before submitting an answer, swapping driver and navigator roles between tasks, and staying on task throughout the lessons, including during the practice tasks near the end of the session. In Class 1, on-task behavior held up well during the teacher-led explanation but tailed off during individual practice, where several students were observed waiting for help, or appearing to disengage after an error on the worksheet. Figure 1 plots the mean pre-test and post-test scores for the two groups, and Table 1 summarizes the achievement and engagement results.

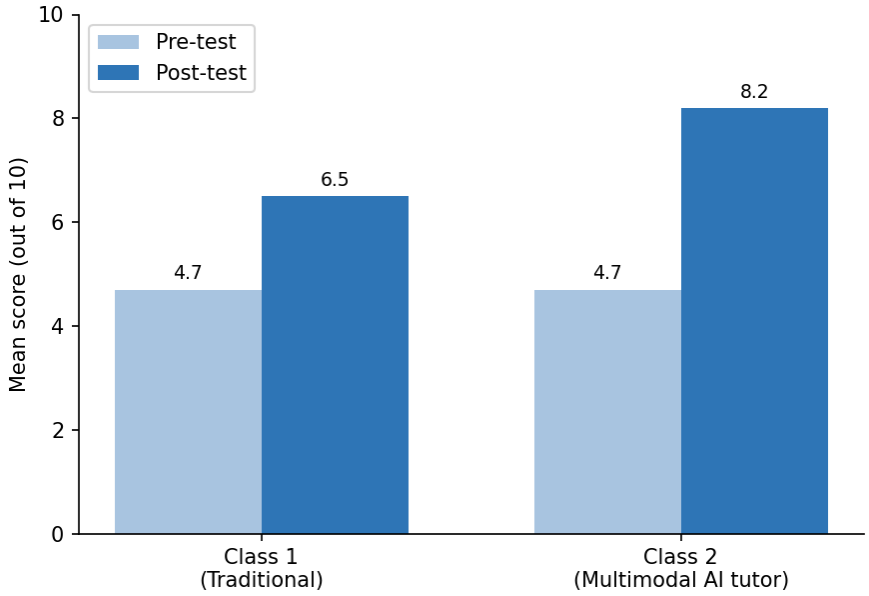

Figure 1. *Mean scores (out of 10) on the conditional statements test at pre-test and post-test for the traditional instruction group (Class 1) and the multimodal AI tutor group (Class 2)*

Table 1. *Pre-test and post-test means (and standard deviations) for the conditional statements test, and mean engagement ratings, by group*

Group	Pre-test M	Pre-test SD	Post-test M	Post-test SD	Engagement M	Engagement SD
Class 1 (Traditional)	4.7	1.1	6.5	1.2	2.6	0.6
Class 2 (Multimodal AI tutor)	4.7	1.1	**8.2**	0.9	**3.5**	0.4

Note. M = mean, SD = standard deviation, CI = confidence interval; n = 22 per group (Class 2 worked in 11 pairs using the driver-navigator technique). Engagement rating is the mean of five items rated on a 4-point scale (1 = strongly disagree to 4 = strongly agree). Between-groups comparisons: post-test $t(42) = 5.32$, $p < .001$, $d = 1.60$; engagement $t(42) = 5.85$, $p < .001$, $d = 1.77$; pre-test not significant.

5. Discussion

The results support both hypotheses. Students who used the multimodal AI tutor to read conditional code scored significantly higher on the post-test than those taught in the traditional, teacher-led way on the same content, and they reported substantially more interest and enjoyment during the lessons. The biggest gains landed on items involving if-else statements and short sequences of conditions, the type most associated with control-flow misconceptions. That pattern suggests the tutor's visual representations and adaptive feedback hit this common source of error directly.

Three features of the Class 2 condition seem especially relevant here. First, generating flowcharts and execution traces on demand for each specific snippet gave every pair access to a visual built for the task at hand, rather than relying on a handful of diagrams drawn on the board for the whole class. Second, the adaptive response to wrong answers, where the tutor produced a step-by-step visual explanation and asked the pair to try again, delivered a kind of

immediate correction that one teacher can scarcely reproduce across twenty-two students. Third, the driver-navigator structure forced students to put their reasoning into words for a partner before submitting, which may have prompted more explicit talk about conditions, branches, and the order of execution than either solo work or whole-class instruction would. Taken together, these features may have nudged Class 2 students away from a flat, line-by-line reading of code and toward tracing which branch a condition selects.

The engagement results suggest the benefits were not confined to achievement. Class 2 students stayed on task through the practice portion of the lessons, unlike some of their Class 1 peers, who seemed to disengage after an error on the worksheet. This fits a simple idea. Immediate, visual feedback on a wrong answer may ease the frustration of not knowing how to proceed, letting students keep attempting tasks instead of waiting for the teacher or giving up.

Several limitations should temper these results. The study covered two lessons with two intact classes of 22, assigned to conditions at the class level rather than the individual level. In Class 2, students worked in pairs rather than alone, so the post-test and engagement results for that group reflect performance and attitudes within pairs at a shared computer, not fully independent responses; it is possible that the pair-based, driver-navigator structure itself contributed to the gains, alongside the multimodal AI tutor. As with any between-groups design built on intact classes, differences between the two classes, prior teacher, classroom dynamics, composition, may have driven some of the observed differences independently of the instructional condition. And the sample, though large enough to detect a sizable effect on two lessons, is modest. We looked at two lessons on a single topic; the study says nothing about whether similar effects would appear for other programming topics, such as loops or functions, or whether the achievement gains would hold over time without continued use of the tutor.

As with any classroom use of generative AI, an extended deployment would call for ongoing monitoring of the content the tutor generates, even though we reviewed every visual in advance here and confined the tutor to a fixed bank of tasks. Larger studies, with multiple lessons and individual-level assignment, would help establish whether the achievement and engagement benefits seen here carry over across topics, classes, and time.

6. Conclusion

This study set traditional, teacher-led instruction against a multimodal AI tutor for two lessons on conditional (if and if-else) statements, with 44 students aged 11 to 12 and the tutor group working in pairs at shared computers under a driver-navigator approach. Students who used the

tutor, which generated visual representations of each conditional's execution and offered adaptive, visual help after wrong answers, scored significantly higher on the post-test and reported substantially more interest and enjoyment than students taught traditionally on the same content. The implication is straightforward. Multimodal AI tools can meaningfully support both achievement and engagement on a topic, conditional statements, that is widely recognized as a difficult transition point in introductory programming.

From a practical standpoint, these results suggest that teachers introducing or reinforcing if and if-else statements may gain from multimodal AI tools that generate individualized visual representations and respond directly to student errors, especially during independent practice, when one-to-one teacher support is hardest to provide. Such tools need not replace teacher-led explanation. In this study the tutor occupied the same lesson content and duration as the traditional approach, which suggests it can slot into an existing lesson structure rather than demanding substantial extra time.

Future research should build on this work with larger samples, individual-level assignment where feasible, and a wider span of programming topics and lesson durations, to learn whether the achievement and engagement gains observed in this lesson study persist over time and generalize across contexts. Attention should also fall on the accuracy and appropriateness of AI-generated visual content in any extended deployment, so that the benefits seen here are realized consistently across different classrooms and student populations.

Disclosure Statement

The authors declare that there is no conflict of interest regarding the publication of this article. No financial, personal, or professional relationships have influenced the research, analysis, or conclusions presented in this work.

Notes on Contributors

Dimitrios Kravvaris holds a Ph.D. in Informatics from the Ionian University, Greece, where his research focused on the analysis of educational video lectures using social and linguistic metadata. He currently serves as Deputy Headmaster and Computer Science teacher at the 3rd Primary School of Ioannina. His research interests include educational data mining, social network analysis, STEM education, educational robotics, and AI in teaching. He serves as a reviewer for the Journal of the Knowledge Economy (Springer) and Engineering Applications of Artificial Intelligence (Elsevier).

jkravv@gmail.com

ORCID

Dimitrios Kravvaris https://orcid.org/0000-0003-3097-938X

References

Anderson, R. K., & Boyle, M. (2023). Generative AI tutors for primary mathematics: A systematic review. *Computers and Education: Artificial Intelligence*, *5*, 100142–100142. https://doi.org/10.1016/j.caeai.2023.100142

Blaney, N. T., Stephan, C. W., Rosenfield, D., Aronson, E., & Sikes, J. (1977). Interdependence in the classroom: A field study. *Journal of Educational Psychology*, *69*(2), 121–128. https://doi.org/10.1037/0022-0663.69.2.121

Chen, L., Wu, Y., & Park, S. (2022). Multimodal learning analytics in K-12 mathematics: Opportunities and risks. *British Journal of Educational Technology*, *53*(6), 1601–1620. https://doi.org/10.1111/bjet.13210

Holmes, W., Bialik, M., & Fadel, C. (2023). *Artificial intelligence in education* (pp. 621–653). https://doi.org/10.58863/20.500.12424/4276068

Hopcan, S., Polat, E., & Albayrak, E. (2022). Collaborative Behavior Patterns of Students in Programming Instruction. *Journal of Educational Computing Research*, *60*(4), 1035–1062. https://doi.org/10.1177/07356331211062260

Kravvaris, D., & Fotou, A. (2025). Responsible AI in the classroom: A pedagogical and ethical framework for developing educational chatbots with Gemini API. In *Artificial Intelligence and Innovation in Education: Ethical and Technological Dimensions – AI Education 2025 Official Conference Proceedings* (pp. 303–310). AI Education 2025. ISBN 978-618-5630-30-0.

Lee, G.-G., Shi, L., Latif, E., Gao, Y., Bewersdorff, A., Nyaaba, M., Guo, S., Wu, Z., Liu, Z., Wang, H., Mai, G., Liu, T., & Zhaı, X. (2023). Multimodality of AI for Education: Towards Artificial General Intelligence. In *ArXiv.org*. https://doi.org/10.48550/arxiv.2312.06037

Li, Y., Yang, R., Gui, S., Shi, P., Huang, X., Yang, D., Zhang, X. J., & Gai, Y. (2024). Visualizing program behavior: A study of enhanced program diagrams using LLM. In *2024 IEEE Frontiers in Education Conference (FIE)* (pp. 1–5). IEEE. https://doi.org/10.1109/fie61694.2024.10893082

Liu, J., Sun, D., Sun, J., Wang, J., & Yu, P. L. H. (2025). Designing a generative AI enabled learning environment for mathematics word problem solving in primary schools: Learning performance, attitudes and interaction. *Computers and Education Artificial Intelligence*, *9*, 100438–100438. https://doi.org/10.1016/j.caeai.2025.100438

Lodge, J. M., Kennedy, G., Lockyer, L., Arguel, A., & Pachman, M. (2018). Understanding Difficulties and Resulting Confusion in Learning: An Integrative Review. *Frontiers in Education*, *3*. https://doi.org/10.3389/feduc.2018.00049

Ng, O.-L., & Chan, T. (2022). Multimodal mathematics learning with AI-generated representations: A design-based study. *ZDM – Mathematics Education*, *54*(7), 1471–1487. https://doi.org/10.1007/s11858-022-01376-2

Ngadengon, Z., Subramaniam, T. S., Yasak, Z., Syukri, M., & Hazim, M. N. (2024). COMPUTATIONAL THINKING AND PAIR PROGRAMMING: A SYSTEMATIC REVIEW. *International Journal of Modern Education*, *6*(20), 103–119. https://doi.org/10.35631/ijmoe.620008

Nia, H. S., & Yaghoobzadeh, A. (2018). Analysis of pretest-posttest studies with control group: Comparison of two types of effect sizes. *DOAJ (DOAJ: Directory of Open Access Journals)*. https://doaj.org/article/598eda4b5e8a45ba9b3f3855b8a9af78

Ponggawa, V., Annas, M. N., Boediman, T. P. P., Abdullah, E. R., & Ramopolii, V. J. F. (2026). The Application of Flowcharts in Understanding Programming Logic for New Students. *Cerdika Jurnal Ilmiah Indonesia*, *6*(3), 1053–1061. https://doi.org/10.59141/cerdika.v6i3.3395

Qu, A., Wen, Y., Zhang, J., Yunge, W., Zhao, Y., Alok, P., Salazar-Gómez, A. F., Liang, P. P., & Zhao, J. (2025). ClassMind: Scaling Classroom Observation and Instructional Feedback with Multimodal AI. In *ArXiv.org*. https://doi.org/10.48550/arxiv.2509.18020

Rau, M. A., Aleven, V., Rummel, N., & Pardos, Z. A. (2013). How Should Intelligent Tutoring Systems Sequence Multiple Graphical Representations of Fractions? A Multi-Methods Study. *International Journal of Artificial Intelligence in Education*, *24*(2), 125–161. https://doi.org/10.1007/s40593-013-0011-7

Robertson, J., Gray, S., Toye, M., & Booth, J. N. (2020). The relationship between Executive Functions and Computational Thinking. *International Journal of Computer Science Education in Schools*, *3*(4), 35–49. https://doi.org/10.21585/ijcses.v3i4.76

Urhahne, D., & Wijnia, L. (2023). Theories of Motivation in Education: an Integrative Framework. *Educational Psychology Review*, *35*(2). https://doi.org/10.1007/s10648-023-09767-9

Xu, T., Zhang, Y., Chu, Z., & Wen, Q. (2025). Multimodal AI Teacher: Integrating Edge Computing and Reasoning Models for Enhanced Student Error Analysis. *AI Magazine*, *46*(3). https://doi.org/10.1002/aaai.70030

Zhao, C. Q., Cao, J., & Koedinger, K. R. (2026). LLM-based Multimodal Feedback Produces Equivalent Learning and Better Student Perceptions than Educator Feedback. In *arXiv (Cornell University)*. Cornell University. https://doi.org/10.48550/arxiv.2601.15280

Appendix A

PRE-TEST: CONDITIONAL STATEMENTS (10 ITEMS)

Instructions: For each code snippet, write what the program prints (its output). If nothing is printed, write "nothing". Show your reasoning in the space provided where requested.

1. What is printed?

```
score = 8
if score > 5:
    print("pass")
```

2. What is printed?

```
score = 3
if score > 5:
    print("pass")
```

3. What is printed?

```
temp = 30
if temp > 25:
    print("hot")
else:
    print("cool")
```

4. What is printed?

```
n = 4
if n > 10:
    print("big")
else:
    print("small")
```

5. What is printed? (show your reasoning)

```
x = 6
if x % 2 == 0:
    print("even")
else:
    print("odd")
```

6. What is printed?

```
y = 7
if y < 0:
    print("negative")
print("done")
```

7. What is printed?

```
a = 10
b = 20
if a > b:
    print("a wins")
else:
    print("b wins")
```

8. What is printed? (show your reasoning)

```
mark = 50
if mark >= 50:
    print("passed")
else:
    print("failed")
```

9. What is printed? (show your reasoning)

```
n = 5
if n > 3:
    print("A")
if n > 8:
    print("B")
```

10. What is printed?

```
rain = True
if rain:
    print("take umbrella")
else:
    print("no umbrella")
```

Answer key: 1. pass; 2. nothing (the condition is false, so the if body is skipped); 3. hot; 4. small; 5. even (6 is divisible by 2); 6. done (the if body is skipped because 7 is not < 0, but print("done") runs regardless); 7. b wins (a is not greater than b); 8. passed (50 >= 50 is true at the boundary); 9. A (only the first condition is true; the two ifs are independent); 10. take umbrella (rain is True).

Appendix B

POST-TEST: CONDITIONAL STATEMENTS (10 ITEMS)

Instructions: For each code snippet, write what the program prints (its output). If nothing is printed, write "nothing". Show your reasoning in the space provided where requested.

1. What is printed?

```
points = 9
if points > 5:
    print("win")
```

2. What is printed?

```
points = 2
if points > 5:
    print("win")
```

3. What is printed?

```
speed = 40
if speed > 30:
    print("fast")
else:
    print("slow")
```

4. What is printed?

```
n = 3
if n > 10:
    print("high")
else:
    print("low")
```

5. What is printed? (show your reasoning)

```
x = 9
if x % 2 == 0:
    print("even")
else:
    print("odd")
```

6. What is printed?

```
y = 4
if y < 0:
    print("negative")
print("end")
```

7. What is printed?

```
a = 30
b = 15
if a > b:
    print("a wins")
else:
    print("b wins")
```

8. What is printed? (show your reasoning)

```
mark = 49
if mark >= 50:
    print("passed")
else:
    print("failed")
```

9. What is printed? (show your reasoning)

```
n = 10
if n > 3:
    print("X")
if n > 8:
    print("Y")
```

10. What is printed?

```
sunny = False
if sunny:
    print("sunglasses")
else:
    print("jacket")
```

Answer key: 1. win; 2. nothing (the condition is false); 3. fast; 4. low; 5. odd (9 is not divisible by 2); 6. end (the if body is skipped, but print("end") runs regardless); 7. a wins (30 > 15 is true); 8. failed (49 >= 50 is false); 9. X then Y (both conditions are true; the two ifs are independent); 10. jacket (sunny is False, so the else block runs).

Appendix C

LESSON EXERCISES: CONDITIONAL STATEMENTS (10 ITEMS, USED BY BOTH GROUPS)

These exercises were used during the two lessons by both groups. Students in Class 1 completed them individually on paper, with the teacher demonstrating selected examples on the board. Students in Class 2 completed them in pairs at a shared computer running the multimodal AI tutor, which provided a step-by-step visual explanation (a flowchart or execution trace) if the pair's first answer was incorrect.

Lesson 1

1. If only: what is printed? Draw a flowchart with one decision.

```
x = 7
if x > 5:
    print("yes")
```

2. If only (false condition): what is printed? Trace which lines run.

```
x = 2
if x > 5:
    print("yes")
print("end")
```

3. If-else (true branch): which branch runs? Use a flowchart.

```
n = 8
if n > 4:
    print("big")
else:
    print("small")
```

4. If-else (false branch): which branch runs?

```
n = 1
if n > 4:
    print("big")
else:
    print("small")
```

Lesson 2

5. Even or odd: trace the condition and predict the output.

```
v = 10
if v % 2 == 0:
    print("even")
else:
    print("odd")
```

6. Comparing two variables: which is printed?

```
a = 5
b = 9
if a > b:
    print("a")
else:
    print("b")
```

7. A skipped if followed by a statement: what is printed? (Does print("hello") always run?)

```
age = 12
if age >= 18:
    print("adult")
print("hello")
```

8. Two separate ifs: what is printed?

```
n = 6
if n > 2:
    print("P")
if n > 9:
    print("Q")
```

9. Boolean variable as the condition: which branch runs?

```
door_open = True
if door_open:
    print("come in")
else:
    print("closed")
```

10. Boundary value: does it print pass or fail?

```
mark = 50
if mark >= 50:
    print("pass")
else:
    print("fail")
```

Appendix D

POST-LESSON ENGAGEMENT AND ATTITUDE QUESTIONNAIRE

Instructions: Circle the number that best shows how much you agree with each sentence about the lessons. 1 = Strongly disagree, 2 = Disagree, 3 = Agree, 4 = Strongly agree.

1. I found the lessons on if and if-else statements interesting. 1 2 3 4

2. I enjoyed the lessons. 1 2 3 4

3. The explanations in both lessons were clear and easy to follow. 1 2 3 4

4. I feel confident about using if and if-else statements after those lessons. 1 2 3 4

5. I would like to learn other programming topics this way. 1 2 3 4

Scoring note: the overall engagement rating reported in Table 1 is the mean of the five item scores (range 1 to 4) for each student.