IMPROVING NOVICE PROGRAMMING OUTCOMES IN UZBEKISTAN: A QUASI-EXPERIMENTAL STUDY OF THE IPRV INSTRUCTIONAL PROTOCOL

Gurdeep Singh Virdee
MSc Data Science (Dual Masters), Assistant Professor, Department of Computer Science,
Acharya University, Karakul, Uzbekistan
https://doi.org/10.5281/zenodo.18400095

Abstract. *Students entering higher education in Uzbekistan frequently begin programming courses with minimal prior exposure to formal computing concepts. In such contexts, conventional instruction—typically consisting of theoretical explanation followed by unstructured laboratory execution—often fails to support conceptual understanding among novice learners. This paper reports the classroom implementation and evaluation of the IPRV (Instruct–Predict–Run–Verify) pedagogical protocol, a structured instructional cycle designed to promote reasoning and reflection prior to code execution. The IPRV protocol was implemented over a four-month instructional period in undergraduate programming courses and evaluated using a quasi-experimental, non-equivalent groups design. Student performance in an intervention group taught using IPRV (n = 25) was compared with a control group receiving traditional instruction followed by laboratory execution (n = 28). Results indicate that student performance in the IPRV group was strongly associated with higher learning outcomes (Mann–Whitney U = 581.5, p < .001), with a large practical effect size (r = 0.57). Distributional and quartile analyses further suggest reduced performance disparity and improved outcomes among lower-performing students. While the instructional components of IPRV are individually well established, the contribution of this study lies in their mandatory sequencing, enforced prediction commitment, and empirical validation as a unified pedagogical protocol for novice programmers. The findings suggest that carefully constrained instructional design can meaningfully support introductory programming learning in contexts where students enter higher education with limited prior IT exposure. These results have immediate practical implications for curriculum developers and university instructors in Uzbekistan, offering a scalable, evidence-based strategy to enhance digital literacy and instructional quality in alignment with national education reforms.*

Keywords: *IT education reform, Novice programming pedagogy, Instructional scaffolding, Cognitive load theory.*

1. Introduction

Rapid advances in digital technologies have positioned programming literacy as a core competency within contemporary higher education systems. In Uzbekistan, national initiatives such as IT Park Uzbekistan and One Million Coders reflect a sustained policy effort to strengthen computational skills and align education with emerging labor market demands. International organizations, including UNESCO and UNICEF, have similarly emphasized the role of structured, accessible instruction in supporting digital education reform within the country (UNESCO, 2023; UNICEF, 2025).

Despite these initiatives, a persistent challenge remains at the university level: A substantial proportion of students begin university studies without prior hands-on experience in computer

programming. Although informatics is included in the secondary school curriculum, instructional depth and emphasis on conceptual reasoning vary considerably across institutions and regions. As a result, introductory programming courses frequently enroll students who lack foundational mental models of program execution and struggle to engage meaningfully with code.

In many university settings, introductory programming instruction follows a familiar pattern: theoretical explanation of syntax and constructs, followed by immediate execution of example programs in laboratory sessions. While this approach may suffice for students with prior exposure, it often proves insufficient for true beginners. Novice learners frequently rely on trial-and-error execution, using program output as a substitute for understanding rather than as evidence to evaluate prior reasoning. This tendency can lead to superficial learning, persistent misconceptions, and declining engagement.

This study examines an alternative instructional approach: the IPRV (Instruct–Predict–Run–Verify) pedagogical protocol. IPRV structures introductory programming instruction as a mandatory learning cycle in which students must first receive explicit instruction, then commit to predictions about program behavior before execution, and finally verify outcomes through guided reflection. Unlike instructional approaches in which prediction or reflection are optional, IPRV enforces these stages as non-negotiable components of learning.

The protocol was implemented continuously over a four-month period in undergraduate programming courses and evaluated under authentic classroom conditions. Rather than proposing a new learning theory, this study investigates whether a constrained and systematically sequenced instructional protocol is associated with improved learning outcomes for novice programmers. The findings contribute empirical evidence relevant to programming education in Uzbekistan and comparable contexts where students enter higher education with heterogeneous educational backgrounds and limited prior exposure to computing.

2. Computer Science Education in Uzbekistan

Computer science education in Uzbekistan has expanded considerably since independence, yet its development has been uneven across educational levels and regions. During the Soviet period, computing education was largely theoretical and restricted to specialized institutions. Following independence, informatics was introduced as a distinct subject within the national curriculum; however, the depth of programming instruction and emphasis on conceptual reasoning continue to vary widely (ResearchGate, 2024).

In recent years, the Uzbek government has placed renewed emphasis on digital education. National initiatives such as One Million Coders and the establishment of IT Park Uzbekistan reflect a policy commitment to developing practical IT skills and fostering industry–academia collaboration. Parallel efforts by international organizations have supported instructional innovation and teacher development. For example, UNICEF-supported technology-enhanced learning pilots and UNESCO-backed distance education initiatives during the COVID-19 pandemic demonstrated that structured, well-supported instruction can lead to measurable learning gains even in resource-constrained settings (UNESCO, 2024; UNICEF, 2025).

These initiatives highlight two themes relevant to the present study. First, effective learning outcomes in Uzbekistan have been associated more strongly with instructional structure and teacher guidance than with the introduction of advanced technologies alone. Second, pedagogical approaches must remain adaptable to institutional contexts characterized by heterogeneous student preparation and variable access to computing infrastructure.

The IPRV pedagogical protocol was developed within this educational landscape. By emphasizing explicit instruction, structured reasoning, and guided reflection, the protocol aligns with national priorities while addressing persistent challenges in introductory programming education at the university level.

3. Challenges for Novice Programming Learners

Learning to program presents well-documented challenges for novice learners, particularly those encountering formal programming concepts for the first time. Programming requires learners to coordinate abstract syntax, semantics, and execution flow—demands that impose a high intrinsic cognitive load on working memory (Sweller, Ayres, & Kalyuga, 2011). Without carefully designed instructional support, novice learners are prone to overload, resulting in fragmented understanding and persistent misconceptions.

Research in computer science education has shown that many beginners struggle to form accurate mental models of how programs execute, even after completing introductory courses (Robins, Rountree, & Rountree, 2003). In traditional instructional settings, students often move directly from explanation to execution, relying on trial-and-error approaches that prioritize producing output over understanding underlying logic. Such practices can mask misconceptions rather than resolve them.

Motivational factors further complicate novice learning. Early difficulties, repeated errors, and lack of formative feedback can reduce learner confidence and engagement. This pattern is particularly evident among students entering higher education without prior exposure to computing, a common situation in the Uzbek context. When execution becomes the primary learning mechanism, students may disengage cognitively, using program output as a proxy for understanding.

The IPRV pedagogical protocol was designed to address these challenges by restructuring how novices engage with code. By enforcing prediction commitment prior to execution and systematic verification afterward, IPRV seeks to promote deliberate reasoning, support mental model development, and reduce unproductive trial-and-error behavior. The protocol does not eliminate execution but situates it within a structured reasoning cycle that encourages deeper engagement with program behavior.

4. The IPRV Pedagogical Protocol

The IPRV (Instruct–Predict–Run–Verify) pedagogical protocol is a structured instructional cycle designed to support novice programming learners through mandatory sequencing and constrained engagement. Unlike loosely structured active-learning approaches, IPRV functions as an instructional protocol: each stage must be completed before progression to the next is permitted.

While instruction, prediction, execution, and reflection are individually familiar elements in programming pedagogy, the contribution of IPRV lies in their enforced integration and empirical validation as a single, repeatable learning cycle. The protocol was implemented consistently across all instructional units during the four-month study period.

4.1 Instruct

Each instructional cycle begins with explicit instructor-led instruction. During this stage, the instructor introduces programming concepts, syntax, and execution logic using short explanations and worked examples. The focus is on establishing a clear conceptual baseline rather than encouraging exploratory discovery.

This emphasis aligns with research demonstrating that novice learners benefit from guided instruction and modeling before engaging in independent problem-solving (Sweller et al., 2011).

In the IPRV classrooms, instruction was deliberately concise and concept-focused, avoiding unnecessary syntactic complexity that could increase extraneous cognitive load.

Importantly, instruction in IPRV is not treated as a sufficient learning activity on its own. Students are explicitly informed that understanding will be tested through reasoning and prediction rather than immediate execution.

4.2 Predict (Prediction Commitment)

Following instruction, students enter the Predict stage, where they are required to commit to a predicted outcome for a given code segment or problem scenario. Predictions are recorded explicitly before any execution is allowed.

This requirement distinguishes IPRV from instructional approaches in which prediction is optional, informal, or purely mental. By enforcing prediction commitment, the protocol ensures that all students externalize their reasoning and articulate expectations about program behavior.

Prediction-based activities have been shown to activate prior knowledge and expose misconceptions (Guzdial, 2010). In IPRV, the pedagogical value of prediction is strengthened by its mandatory nature: students cannot proceed without first engaging cognitively with the problem. This design increases learner accountability and reduces passive engagement.

4.3 Run

Only after predictions are recorded do students proceed to the Run stage, where they execute the code using a programming environment. Execution is deliberately delayed to prevent students from using output as a substitute for reasoning.

During execution, students are encouraged to observe program behavior carefully, noting intermediate values and control flow. Execution is framed as a means of testing prior reasoning rather than discovering answers through trial and error.

This separation of reasoning and execution is particularly important for novice learners, who are more susceptible to cognitive overload when multiple demands compete simultaneously for working memory resources (Sweller et al., 2011).

4.4 Verify

The Verify stage requires students to compare predicted outcomes with actual execution results. This comparison is conducted through guided discussion, instructor questioning, or structured reflection activities.

Verification serves as the reflective core of the protocol. Discrepancies between prediction and outcome prompt learners to identify faulty assumptions and revise their mental models. Errors are treated as diagnostic information rather than failures, reinforcing a learning-oriented classroom culture.

Research on novice programming consistently highlights the importance of immediate, structured reflection in supporting conceptual change (Robins, Rountree, & Rountree, 2003). By embedding verification as a mandatory stage, IPRV ensures that reflection is systematic rather than incidental.

5. Theoretical and Pedagogical Foundations of IPRV

The IPRV pedagogical protocol is grounded in established principles from cognitive psychology and computer science education. Rather than proposing a new learning theory, the protocol operationalizes existing research into a constrained instructional design suited to novice learners.

5.1 Cognitive Load Regulation

The abstract and symbolic nature of programming concepts places substantial intrinsic demands on novice learners' working memory. Cognitive Load Theory suggests that instructional design should minimize extraneous load to prevent working memory overload, particularly for novices (Sweller et al., 2011).

IPRV addresses this by sequencing tasks so that learners engage in one cognitive demand at a time: instruction precedes reasoning, reasoning precedes execution, and execution precedes reflection. This separation reduces cognitive interference and supports schema construction over repeated cycles.

5.2 Prediction, Commitment, and Conceptual Change

Research on prediction-based learning indicates that requiring learners to anticipate outcomes can activate prior knowledge structures and reveal underlying misconceptions (Guzdial, 2010). IPRV extends this principle by requiring prediction commitment, ensuring that learners actively engage with program logic rather than passively observing execution.

The commitment aspect is pedagogically significant. When learners compare predicted and actual outcomes, cognitive conflict becomes salient, increasing the likelihood of conceptual change. In contrast, unstructured execution—common in traditional laboratory settings—may allow misconceptions to persist unnoticed.

5.3 Constraint-Based Pedagogy for Novices

A defining characteristic of IPRV is its use of instructional constraints. Rather than maximizing learner freedom, the protocol deliberately restricts when execution can occur. Research suggests that such constraints can be beneficial for novices by discouraging unproductive strategies and focusing attention on core concepts (Sweller et al., 2011).

Within this framework, IPRV can be understood as a pedagogical protocol rather than a collection of activities. Its novelty lies in demonstrating that carefully designed instructional constraints—when applied consistently—are associated with improved learning outcomes in introductory programming contexts.

6. Methodology (Actual Implemented Study)

6.1 Research Design

This study employed a quasi-experimental, non-equivalent groups design to examine the association between the IPRV pedagogical protocol and student learning outcomes in introductory programming courses. Random assignment was not feasible due to institutional constraints; therefore, intact class sections were observed over a continuous four-month instructional period.

The study evaluates classroom practice under authentic conditions rather than testing a hypothetical intervention. The IPRV protocol was implemented from the first week of instruction in the intervention group, while the control group followed conventional instructional practices.

6.2 Participants

Participants were undergraduate students enrolled in introductory programming or computer science courses at a higher education institution in Uzbekistan. All participants were novice learners with limited prior exposure to formal programming.

- IPRV group (intervention): 25 students
- Control group: 28 students

Both groups followed the same syllabus, covered identical topics, and completed the same assessment.

6.3 Baseline Equivalence

To ensure comparability between groups, all students completed an entrance diagnostic test prior to the start of instruction. Analysis of diagnostic scores indicated no meaningful difference in baseline performance between the IPRV and control groups. This finding supports the assumption that observed differences in post-intervention performance are associated with instructional conditions rather than pre-existing disparities.

6.4 Instructional Conditions

IPRV Group

Students in the intervention group were taught using the IPRV (Instruct–Predict–Run–Verify) pedagogical protocol throughout the four-month period. Each instructional unit adhered to the same mandatory sequence:

1. Instructor-led explanation of concepts and examples
2. Student prediction with explicit commitment
3. Code execution
4. Guided verification and reflection

Progression between stages was enforced consistently across lessons.

Control Group

Students in the control group received conventional instruction, consisting of theoretical explanation followed by laboratory-based code execution. Prediction and structured verification were not systematically incorporated. Students typically executed code immediately after instruction.

6.5 Standardization of Instructional Conditions

To minimize instructional variability, both groups used:

- The same syllabus and learning objectives
- The same instructional materials and examples
- The same assessment instrument and grading rubric
- The same instructional duration and schedule

This standardization reduces the likelihood that differences in outcomes were attributable to curricular or assessment-related factors.

6.6 Assessment and Data Analysis

Learning outcomes were measured using a post-instruction assessment aligned with course objectives, with a maximum score of 30 marks. Score distributions were assessed for normality using the Shapiro–Wilk test; results indicated non-normal distributions. Consequently, the Mann–Whitney U test was used for group comparison, and rank-biserial correlation (r) was calculated as an effect size measure. Descriptive statistics and distributional analyses were also conducted.

7. Results

7.1 Descriptive Statistics

Descriptive statistics are summarized in **Table 1**. Students in the IPRV group achieved higher mean and median scores than those in the control group, with a mean difference of 9.27 marks—approximately 31% of the total assessment score.

Group	N	Mean	Median	SD	Min	Max
Control	28	14.57	16.00	5.83	2	25
IPRV	25	23.84	26.00	6.25	13	30

Table 1. Descriptive Statistics of Student Test Scores

Notably, the minimum score in the IPRV group exceeded the lower quartile of the control group, suggesting reduced extreme underperformance among students exposed to the IPRV protocol.

7.2 Inferential Analysis

Given the non-normal distribution of scores, a Mann–Whitney U test was used to compare performance between groups. The analysis revealed a statistically significant difference:

- $U = 581.5$
- $p < .001$

The corresponding effect size, calculated using rank-biserial correlation, was $r = 0.57$, indicating a large practical effect. This result suggests that the performance gains associated with the IPRV protocol are not only statistically reliable but educationally meaningful.

7.3 Distributional Patterns (Figure 1 and Figure 3)

Figure 1 (boxplot comparison) illustrates a clear upward shift in the score distribution for the IPRV group relative to the control group. The control group exhibits greater dispersion and lower outliers, whereas the IPRV group demonstrates a more compact distribution centered at higher scores.

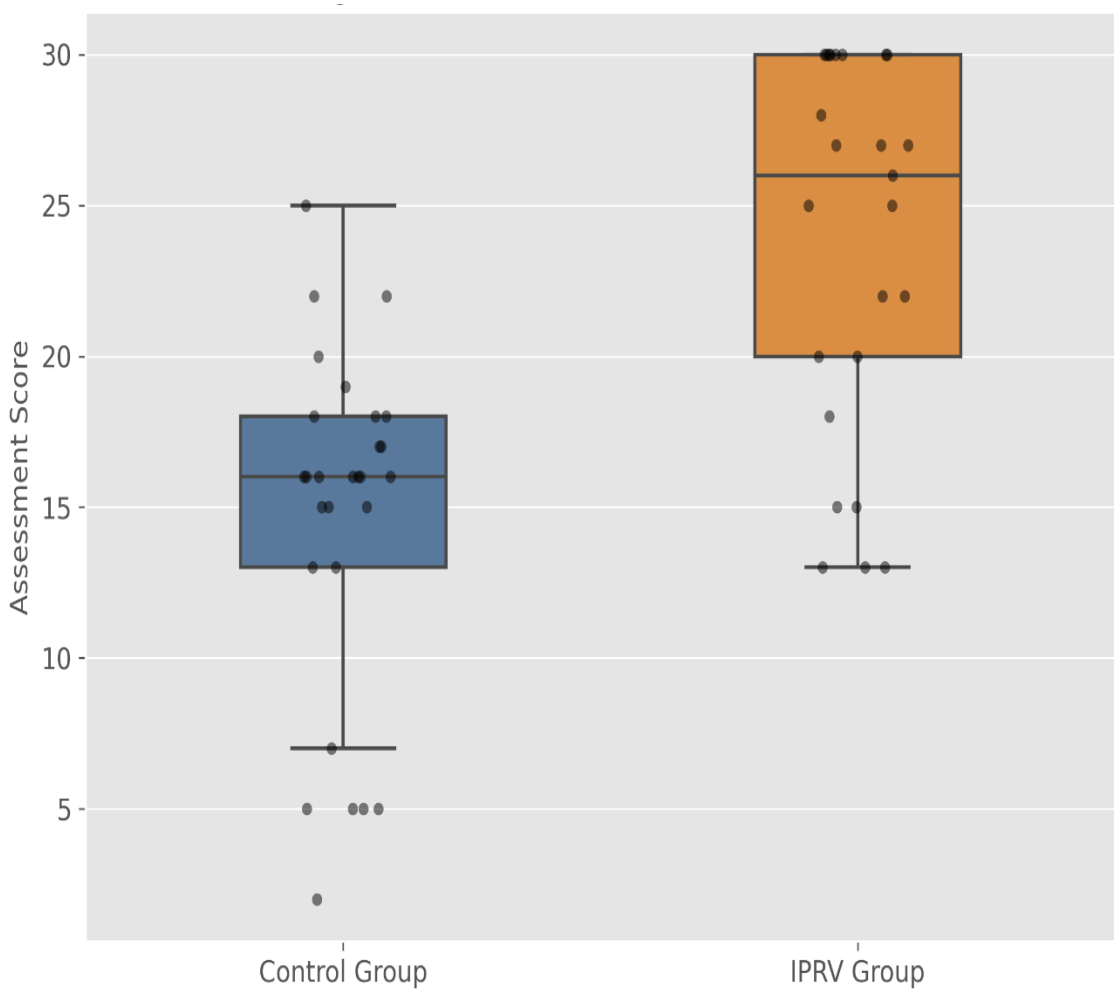


Figure 1. Boxplot Comparison of Post-test Scores

Figure 2 (histogram comparison) further highlights this pattern, showing that IPRV students are concentrated in higher score ranges, while control group scores are more widely spread across lower and mid-range values. These distributional differences suggest that the observed performance gains are not driven by a small subset of high achievers.

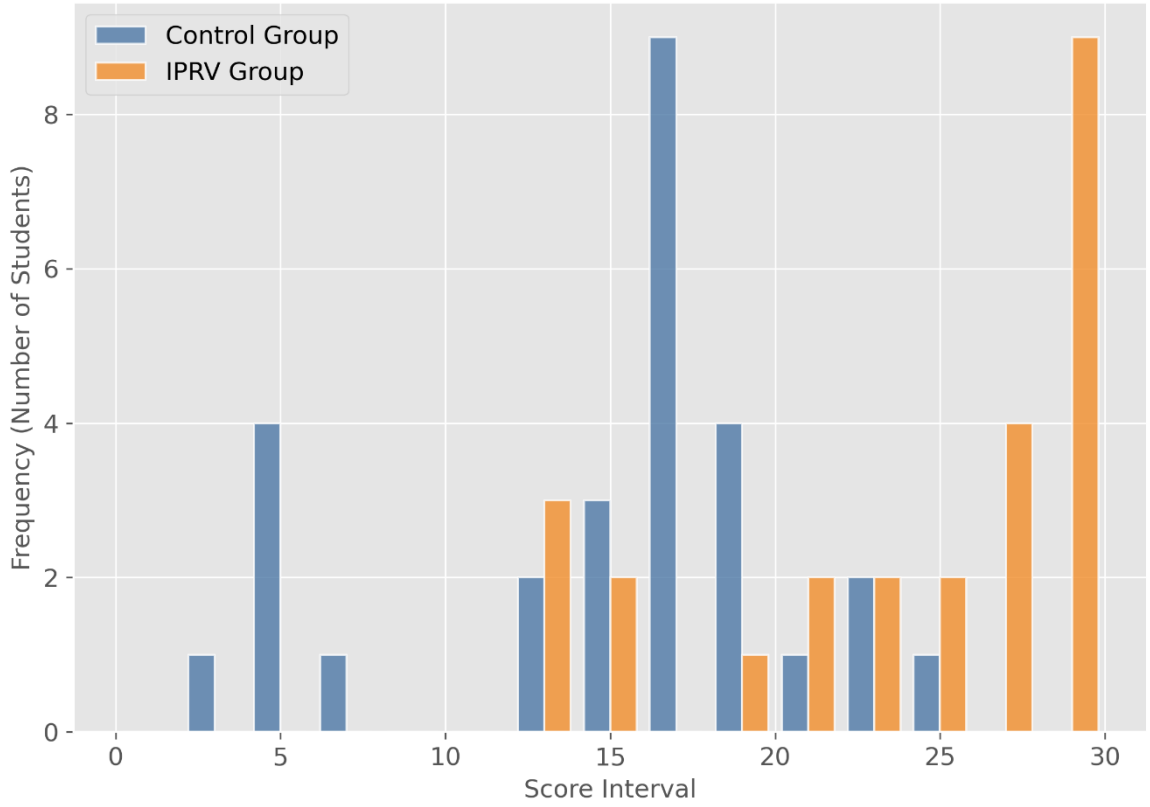


Figure 2. Frequency Distribution of Scores

7.4 Mean Scores and Confidence Intervals (Figure 3)

Figure 3 presents mean scores with 95% confidence intervals for both groups. The limited overlap between confidence intervals reinforces the robustness of the observed difference. The IPRV group's confidence interval lies entirely above that of the control group, indicating consistent performance advantages across the group.

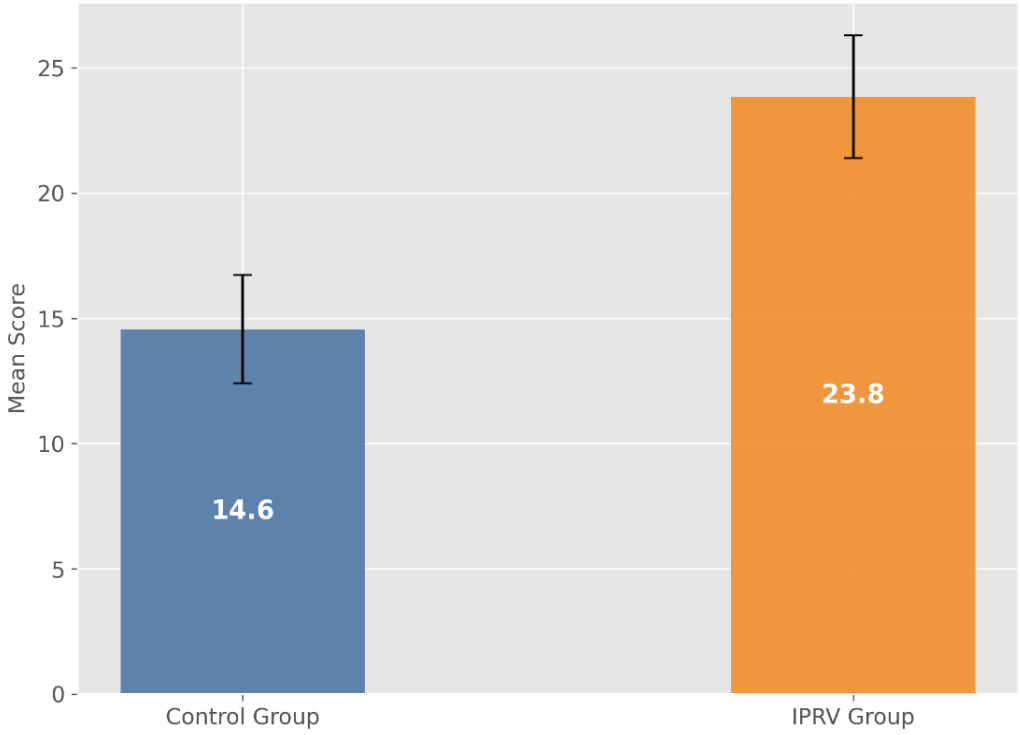


Figure 3. Mean Scores with Standard Error

7.5 Quartile and Equity Analysis (Figure 4)

Figure 4 compares quartile values for both groups and reveals a notable equity-related pattern: the first quartile of the IPRV group exceeds the median of the control group. This suggests that lower-performing students in the IPRV group achieved outcomes comparable to or better than average-performing students in the control condition.

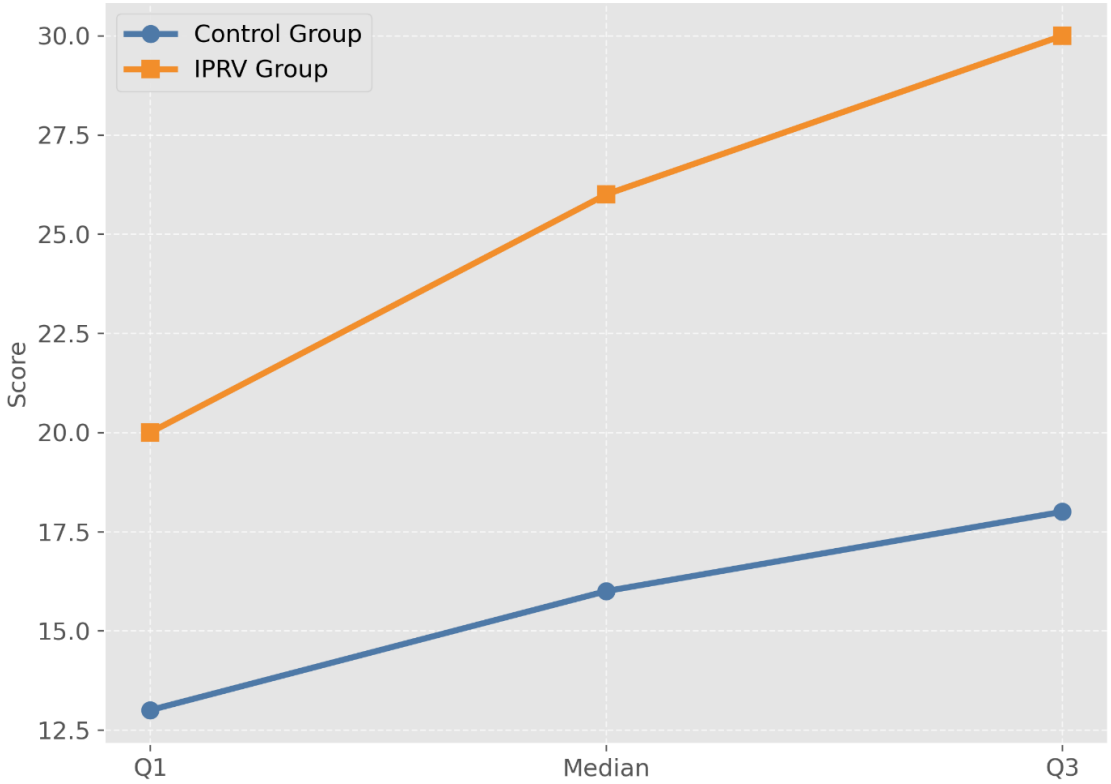


Figure 4. Quartile Comparison of Score Distributions

This pattern indicates that the IPRV protocol is associated not only with higher overall performance but also with reduced performance disparity among novice learners.

7.6 Summary of Findings

Across descriptive, inferential, and distributional analyses, the results consistently show that implementation of the IPRV protocol was strongly associated with improved learning outcomes over the four-month instructional period. The convergence of multiple analytic perspectives strengthens confidence in the robustness of these findings.

8. Discussion and Implications

The results of this study indicate that implementation of the IPRV (Instruct–Predict–Run–Verify) pedagogical protocol was strongly associated with improved learning outcomes for novice programming students over a four-month instructional period. The observed differences between the IPRV and control groups were statistically significant and accompanied by a large practical effect size, suggesting that the protocol has meaningful educational relevance.

A central implication of these findings is the importance of instructional sequencing and constraint in novice programming education. Unlike the control condition—where students typically progressed from instruction directly to execution—the IPRV protocol required learners to commit to predictions before running code and to explicitly verify outcomes afterward. This sequencing appears to shift execution from a trial-and-error activity to a mechanism for testing prior reasoning.

Distributional analyses further suggest that the benefits of IPRV extend beyond high-performing students. The elevation of lower quartile scores and the reduced spread of outcomes

indicate that the protocol may help mitigate performance disparities commonly observed in introductory programming courses. This equity-oriented pattern is particularly relevant in the Uzbek higher education context, where students enter university with diverse educational backgrounds and varying levels of preparation.

From a pedagogical perspective, the findings support the view that constraint-based instructional design can be beneficial for novice learners. While flexibility and open-ended exploration are often emphasized in advanced learning contexts, this study suggests that beginners may benefit from more structured engagement that prioritizes reasoning and reflection before execution.

At an institutional level, the IPRV protocol offers a practical and scalable instructional approach. Its implementation does not require specialized software or additional technological resources, making it suitable for a wide range of higher education settings. For universities in Uzbekistan and similar contexts, IPRV provides an evidence-informed method for strengthening introductory programming instruction without substantial infrastructural investment.

9. Conclusion and Future Work

This study examined the classroom implementation of the IPRV (Instruct–Predict–Run–Verify) pedagogical protocol in introductory programming education for novice learners in Uzbekistan. The findings indicate that the protocol was associated with significantly higher learning outcomes compared to conventional instruction followed by laboratory execution.

Rather than advancing a new learning theory, this study contributes by formalizing and empirically examining a constrained instructional protocol implemented in authentic classroom settings, that reshapes how novice learners engage with programming tasks. By enforcing prediction commitment prior to execution and systematic verification afterward, IPRV promotes deliberate reasoning and supports the development of accurate mental models.

Importantly, this paper focuses on the pre-productive phases of novice learning, emphasizing conceptual understanding before independent code creation. Future research will extend this work by examining how learners transition from verification to guided and independent program construction, including the design and evaluation of additional scaffolding phases that support productive coding without overwhelming beginners.

Within the context of Uzbekistan’s ongoing educational reforms, the IPRV protocol offers a practical, evidence-informed approach for improving introductory computer science education. More broadly, the study suggests that carefully designed instructional constraints can play a valuable role in supporting novice learners in cognitively demanding domains.

REFERENCES

1. Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences*, 111(23), 8410–8415. https://doi.org/10.1073/pnas.1319030111
2. Guzdial, M. (2010). Creating (and improving) options for CS practice: Practice-It! and beyond. *Computing Education Research Blog*. https://computinged.wordpress.com/2010/06/25/creating-and-improving-options-for-cs-practice-practice-it-and-beyond/

3. Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2), 137–172. https://doi.org/10.1076/csed.13.2.137.14200
4. Sbaraglia, S. (2023). *Teaching informatics to novices: Big ideas and the necessity of optimal guidance*. University of Bologna. http://amsdottorato.unibo.it/id/eprint/10914/
5. Sweller, J., Ayres, P., & Kalyuga, S. (2011). *Cognitive Load Theory*. Springer. https://doi.org/10.1007/978-1-4419-8126-4
6. UNESCO. (2023). Advancing digital education in Uzbekistan. https://www.unesco.org/en/articles/advancing-digital-education-uzbekistan
7. UNESCO. (2024). Uzbekistan shows the world how distance education can work in a crisis. *World Education Blog*. https://world-education-blog.org/2025/07/07/uzbekistan-shows-the-world-how-distance-education-can-work-in-a-crisis/
8. UNICEF. (2025). *Eduten piloting implementation report: Uzbekistan*. https://www.unicef.org/innovation/media/24166/file/Uzbekistan%20Eduten%20Piloting%20Implementation%20Report.pdf
9. ResearchGate. (2024). *The Development of IT Education in Uzbekistan*. https://www.researchgate.net/publication/396950637_The_Development_of_IT_Education_in_Uzbekistan