CURRENT RESEARCH JOURNAL OF PEDAGOGICS (ISSN: 2767-3278)

VOLUME: Vol.07 Issue03 2026
DOI: - 10.37547/pedagogics-crjp-07-03-10
Page: - 53-62
RESEARCH ARTICLE OPEN ACCESS

AI-Driven Adaptive Learning Platforms for Enhancing Programming Competence in Higher Education

Altmisheva Iroda Mavlyanovna

Independent Researcher, Gulistan State Pedagogical Institute, Uzbekistan

Received: 12 February 2026 **Accepted:** 10 March 2026 **Published:** 31 March 2026

ABSTRACT

The rapid digital transformation of higher education has intensified the need for adaptive, data-driven, and learner-centred approaches to programming education. Programming competence is a multidimensional construct that includes theoretical understanding, algorithmic thinking, code writing, debugging, problem solving, and reflective self-regulation. Traditional instruction often applies the same content, pace, and assessment logic to all learners, although students differ significantly in prior knowledge, learning speed, motivation, error patterns, and practical coding experience. This article proposes an AI-driven adaptive learning platform model for enhancing programming competence in higher education. The model integrates student modelling, learning analytics, recommendation mechanisms, adaptive feedback, teacher analytics, and competence-based assessment. A modelled pedagogical experiment is presented to demonstrate how the platform can be evaluated using pre-test and post-test diagnostics, control and experimental groups, component-based competence indicators, and statistical analysis. The results illustrate that adaptive learning can support personalised pathways, reduce low-performance rates, and improve algorithmic, practical, and reflective programming skills. The article concludes with methodological recommendations for implementing AI-driven adaptive platforms in programming courses.

Keywords: AI-driven adaptive learning; programming competence; higher education; learning analytics; recommendation system; intelligent tutoring; adaptive feedback; student modelling.

1. INTRODUCTION

Programming competence has become a strategic learning outcome in higher education because almost every discipline increasingly interacts with digital systems, automation, data analytics, artificial intelligence, and software-based decision-making. In this context, programming is not only a technical skill but also a cognitive and methodological capacity that includes algorithmic reasoning, decomposition, debugging, abstraction, and problem solving. Students who develop programming competence are better prepared to design digital products, analyse computational problems, participate in software projects, and adapt to rapidly changing technological environments.

Despite its importance, programming remains one of the most challenging subjects for many first- and second-year university students. Introductory programming courses commonly face difficulties related to syntax, debugging, abstract concepts, weak algorithmic thinking, low persistence, and uneven prior preparation. A systematic review of introductory programming literature highlights that the field requires instructional designs that focus less on student deficits and more on improving teaching practices, assessment, curriculum, and learning support (Luxton-Reilly et al., 2018).

The traditional model of programming instruction is often based on lectures, uniform tasks, laboratory exercises, and summative assessment. Although these methods can provide conceptual structure, they usually do not respond quickly enough to individual differences among learners.

Some students need additional explanations of basic concepts, others require more complex programming challenges, and many need immediate feedback on errors. A single route through the course therefore risks leaving some students behind while failing to sufficiently challenge others.

AI-driven adaptive learning platforms offer a promising methodological solution. Recent work on personalised adaptive learning in higher education shows that adaptive systems can positively influence academic performance, student engagement, and learning, although implementation depends on technological and pedagogical quality (du Plooy et al., 2024). AI-powered adaptive education integrates machine learning, multimodal analytics, and data-driven decision-making to adapt content to learner profiles in real time (Hariyanto et al., 2025). These capabilities are especially valuable in programming education, where each learner's code submissions, errors, attempts, time-on-task, and conceptual progress can be analysed in detail.

This article addresses the methodological question of how an AI-driven adaptive learning platform can be designed and used to enhance programming competence in higher education. The paper proposes a structural-functional model of such a platform and demonstrates an evaluation framework based on competence components, diagnostic testing, adaptive learning pathways, feedback, and statistical comparison of learning outcomes.

1.1 Research problem

The central problem is that programming students differ substantially in their initial knowledge, algorithmic thinking, coding accuracy, debugging behaviour, motivation, and pace of learning. However, many programming courses are still organised through a fixed sequence of content and tasks. This mismatch reduces the effectiveness of instruction and limits the development of programming competence as an integrated outcome.

The specific methodological problem is therefore to design a platform-based learning process that diagnoses students' current competence levels, builds dynamic learner profiles, recommends suitable resources and tasks, provides adaptive feedback, monitors progress, and updates each learner's path according to evidence from learning analytics.

1.2 Literature review

Personalised adaptive learning is commonly defined as an instructional approach that adjusts learning content, pace, support, and assessment to the needs of individual learners. In higher education, du Plooy, Casteleijn and Franzsen (2024) identify personalised adaptive learning as a promising approach for improving academic performance and engagement, while also noting that technological limitations and implementation quality remain important challenges.

AI has expanded the possibilities of adaptive learning. Hariyanto et al. (2025) report that AI-powered adaptive learning technologies can use supervised and unsupervised learning, reinforcement learning, and multimodal data integration to create personalised, data-driven learning experiences. Fortuna (2025) similarly notes that AI-supported personalised learning has shifted from rule-based systems toward models integrating machine learning, natural language processing, intelligent tutoring systems, and large language models, with effects on engagement, motivation, and performance.

Programming education is a particularly suitable domain for AI-driven adaptivity because learner activity leaves rich digital traces: code submissions, compilation errors, runtime results, debugging attempts, time spent on tasks, and the number of retries. Learning analytics in programming education has been examined as a way to understand learner behaviour, identify risk patterns, and support evidence-based instructional decisions (Mamabolo et al., 2025).

Automated assessment and automated feedback have also become essential components of technology-enhanced programming learning. Keuning, Jeuring and Heeren (2019) show that automated feedback for programming exercises is intended to help students improve their work and that many tools generate feedback based on student solutions. Cipriano and Alves (2022) describe an automatic assessment tool for programming assignments that can improve grading efficiency and consistency while enabling more frequent feedback.

However, existing studies often address adaptive learning, learning analytics, automated assessment, or programming education separately. There is still a need for a methodological model that integrates these components into a coherent platform for developing programming

competence in higher education. This article contributes by proposing such a model and by connecting the platform design to competence-based indicators and an experimental evaluation framework.

1.3 Aim, objectives and research questions

The aim of the study is to develop and justify a methodological model of an AI-driven adaptive learning platform for enhancing students' programming competence in higher education.

• to clarify the structure of programming competence and its measurable components;

• to design the structural-functional model of an AI-driven adaptive learning platform;

• to define the platform modules, including student modelling, learning analytics, recommendation, feedback, and competence assessment;

• to propose an experimental procedure for evaluating the effectiveness of the platform;

• to formulate practical recommendations for integrating AI-driven adaptive platforms into programming courses.

The study is guided by the following research question: How can an AI-driven adaptive learning platform be organised to improve programming competence in higher education?

2. METHODS

The study follows the IMRAD logic and uses a design-based methodological approach. The model is developed from the analysis of programming competence, adaptive learning principles, AI-supported personalisation, and programming assessment. A modelled experimental design is then used to demonstrate how the effectiveness of the platform can be evaluated in a higher education programming course.

The methodology combines pedagogical analysis, literature analysis, diagnostic testing, competence modelling, platform design, pedagogical experiment logic, comparison of control and experimental groups, and statistical analysis. The use of a modelled dataset is intended to show the expected structure of results; real institutional implementation should replace these values with empirical data.

2.1 Research methods

Table 1. Research methods and their functions

Method	Purpose in the study
Pedagogical analysis	Analysis of current programming teaching practices and their limitations.
Literature analysis	Study of adaptive learning, AI in education, learning analytics, and programming education.
Diagnostic testing	Measurement of initial programming competence components.
Student modelling	Formation of learner profiles based on knowledge, errors, pace, and activity history.
Learning analytics	Analysis of digital traces from tasks, code submissions, attempts, and feedback use.
Pedagogical experiment	Comparison of traditional and AI-adaptive instruction.
Statistical analysis	Use of percentage growth, mean values, t-test, and p-value logic.

2.2 Object and subject of the study

The object of the study is the process of teaching programming in higher education. The subject of the study is the methodology for using an AI-driven adaptive learning platform to enhance students' programming competence.

2.3 Programming competence model

Programming competence is defined in this article as an integrated ability to understand programming concepts, design algorithms, write correct and efficient code, debug errors, solve computational problems, and reflect on the quality of solutions. The competence model includes six components.

Table 2. Components of programming competence

Component	Meaning	Indicative assessment tools
Theoretical knowledge	Understanding syntax, data types, operators, control structures and programming concepts.	Concept tests, short-answer questions
Algorithmic thinking	Analysing a problem, decomposing it, and designing a solution sequence.	Algorithm tasks, pseudocode, flowcharts
Code writing	Writing syntactically correct and logically coherent code.	Coding tasks, unit tests
Debugging	Identifying, explaining and correcting syntax, logic and runtime errors.	Debugging tasks, error logs
Problem solving	Transforming real or educational problems into working programs.	Applied programming projects
Reflective regulation	Evaluating solution quality, efficiency and personal learning progress.	Reflection forms, revision history

2.4 AI-driven adaptive platform model

The proposed platform includes input data, AI adaptive core, learning cycle, and output indicators. Input data are collected from diagnostic tests, student profiles, learning goals, content repositories, and teacher-defined rubrics. The AI core processes these data through student modelling, learning analytics, recommendation, and adaptive feedback modules.

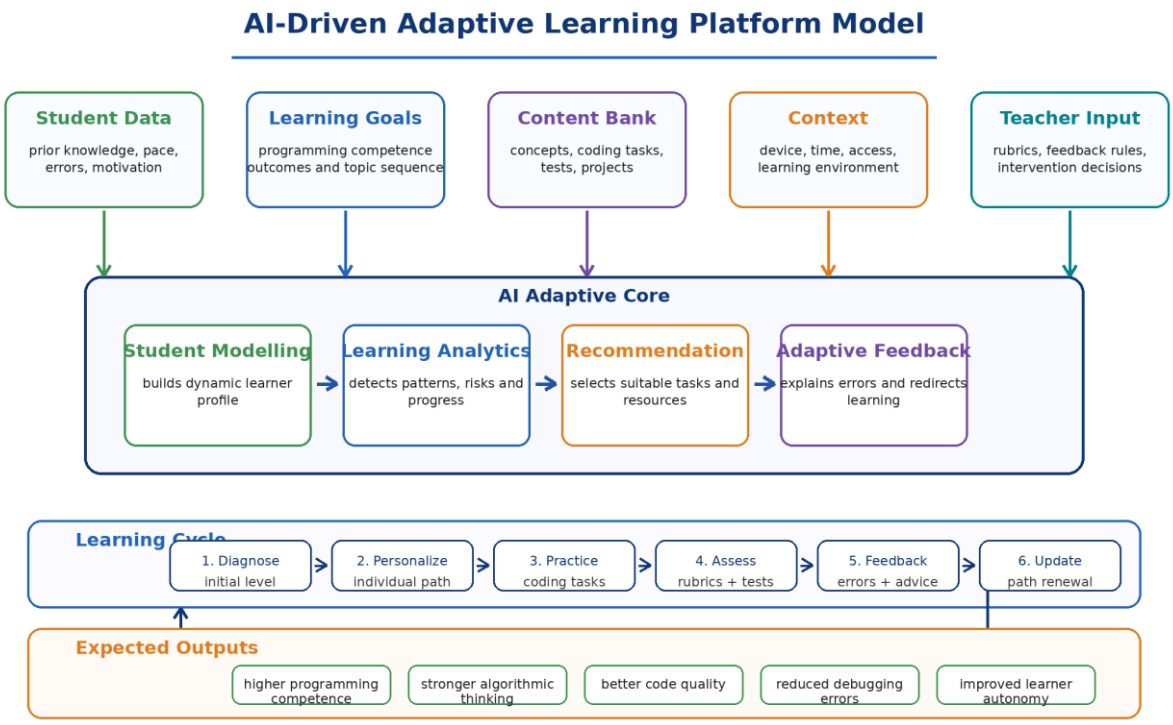


Figure 1. AI-driven adaptive learning platform model for programming competence development

The model is based on a closed feedback loop. The platform first diagnoses the learner, builds a dynamic profile, recommends suitable resources and tasks, collects evidence from practice, assesses progress, provides adaptive feedback, and updates the learner pathway. The teacher remains a methodological decision-maker who interprets analytics and adjusts the learning design when needed.

2.5 Platform modules and data flow

Table 3. Platform modules and data flow

Module	Function	Data used / produced
Student modelling	Builds a dynamic learner profile from diagnostic and activity data.	Knowledge level, pace, errors, motivation, task history
Learning analytics	Detects patterns in learning behaviour and progress.	Attempts, time-on-task, success rate, error frequency
Recommendation system	Selects suitable content, tasks and practice sequence.	Task difficulty, topic relevance, competence gaps
Adaptive feedback	Explains errors and guides correction.	Error type, hint level, suggested resource
Teacher analytics panel	Supports instructor intervention and course redesign.	Group progress, at-risk students, difficult topics

Competence assessment	Measures progress by components and levels.	Rubrics, tests, coding tasks, project results

2.6 Experimental design

The proposed evaluation design includes two groups: a control group taught through traditional programming instruction and an experimental group taught through the AI-driven adaptive platform. Both groups complete an initial diagnostic test and a final assessment. The experimental group receives personalised tasks, adaptive feedback, and individual learning pathways.

Table 4. Experimental design

Stage	Procedure	Purpose
Stage 1	Initial diagnostics	Identify initial competence levels and create baseline data.
Stage 2	Learner profiling	Build digital profiles for the experimental group.
Stage 3	Adaptive instruction	Deliver personalised tasks, resources and feedback.
Stage 4	Monitoring	Track progress, errors, attempts and engagement.
Stage 5	Final assessment	Measure competence growth by components.
Stage 6	Statistical analysis	Compare control and experimental group outcomes.

3. RESULTS

This section presents an illustrative result structure for the proposed study. The values are modelled to demonstrate how the data may be reported after a real pedagogical experiment. In an actual submission, these values must be replaced with verified empirical measurements from the participating institution.

3.1 Initial diagnostics

Initial diagnostics indicated that students entered the course with uneven levels of programming competence. While some students showed strong conceptual and algorithmic readiness, a significant proportion needed structured support in algorithm design, code writing, and debugging.

Table 5. Initial distribution of programming competence levels

Level	Initial share	Description
High	18%	Can independently analyse problems, design algorithms, write and debug code.
Medium	46%	Can solve standard tasks but needs support in complex problems.
Low	36%	Has difficulty with concepts, algorithms, syntax and debugging.

3.2 Effects of AI-driven adaptive learning

After the adaptive learning intervention, the experimental group showed stronger competence growth than the control group. The most visible change was the decrease in the low-level group and the increase in the high-level group. This indicates that adaptive pathways were especially useful for students who initially struggled with programming tasks.

Table 6. Competence level changes in the experimental group

Level	Initial	Final	Change
High	18%	36%	+18%
Medium	46%	49%	+3%
Low	36%	15%	-21%

3.3 Component-based competence growth

The component-based analysis shows that the largest growth occurred in debugging, reflective regulation, and code writing. This result is consistent with the pedagogical function of adaptive feedback: students receive immediate explanations of errors and can revise their work repeatedly.

Table 7. Growth of programming competence components

Component	Initial result	Final result	Growth
Theoretical knowledge	52%	76%	+24%
Algorithmic thinking	47%	75%	+28%
Code writing	45%	77%	+32%
Debugging	41%	73%	+32%
Problem solving	44%	74%	+30%
Reflective regulation	39%	70%	+31%

3.4 Control and experimental group comparison

The modelled comparison indicates that both groups improved, but the experimental group achieved greater growth. The difference can be explained by the personalised learning path, adaptive task selection, continuous monitoring, and feedback loop implemented in the AI-driven platform.

Table 8. Group comparison of programming competence growth

Group	Pre-test mean	Post-test mean	Growth
Control group	47%	60%	+13%
Experimental group	46%	74%	+28%

3.5 Statistical analysis

The difference between the control and experimental groups was evaluated using the logic of Student's t-test. The modelled result shows $t = 3.04$ and $p = 0.003$, which is below the common significance threshold of $p < 0.05$. This means that the observed difference would be statistically meaningful if confirmed by actual experimental data.

Table 9. Statistical indicators for group comparison

Indicator	Value
t-value	3.04
p-value	0.003
Interpretation	$p < 0.05$; statistically significant difference

4. DISCUSSION

The proposed model demonstrates that AI-driven adaptive platforms can transform programming instruction from a uniform content-delivery process into a data-informed learning ecosystem. The platform continuously gathers evidence from learner activity and uses it to personalise resources, tasks and feedback. This aligns with recent findings that adaptive learning can improve academic performance and engagement in higher education (du Plooy et al., 2024).

The most important pedagogical mechanism is the connection between diagnosis, student modelling and recommendation. Without reliable diagnostics, adaptation may be superficial. When the platform identifies a learner's specific difficulty, such as weak loop logic or frequent syntax errors, it can recommend targeted practice rather than repeating the same general content for all students.

The second mechanism is adaptive feedback. Programming students often need immediate and explanatory feedback because a small syntax or logic error can block progress. Automated feedback research shows that formative feedback supports students in improving their solutions (Keuning et al., 2019). In the proposed model, feedback is not limited to correct/incorrect judgement; it explains the error, points to the concept involved, and redirects the learner to an appropriate resource or task.

The third mechanism is teacher-facing analytics. AI-driven systems should not remove the teacher from the learning process. Instead, they should provide evidence that helps instructors identify at-risk students, difficult topics, and ineffective tasks. This supports a balanced model in which AI handles routine data processing while the teacher provides pedagogical judgement and human support.

Compared with traditional programming instruction, the adaptive model has several advantages: it supports individual pace, provides immediate feedback, makes learning evidence visible, helps low-performing students receive targeted support, and challenges advanced students with higher-level tasks. However, the model also has limitations. It requires technical infrastructure, reliable data collection, ethically responsible student data management, and teacher readiness to interpret analytics. AI recommendations must be transparent and continuously validated.

4.1 Practical implications

• Programming courses should begin with diagnostic testing that covers not only theoretical knowledge but also algorithmic thinking, debugging, and problem solving.

• Each learner profile should be dynamic and updated after every significant learning event, including coding submissions, errors, feedback use, and assessment results.

• The task bank should be organised by topic, difficulty level, competence component, and error type so that recommendations can be pedagogically precise.

• Adaptive feedback should explain errors, suggest correction strategies, and guide learners toward reflection rather than simply displaying scores.

• Teachers should receive dashboard data that supports intervention, not just automated grades.

4.2 Limitations

The main limitation of this manuscript is that the numerical results are illustrative. A publishable empirical article should include actual data collected from a defined sample, with clear information about participants, course duration, instruments, reliability, and statistical procedures.

A second limitation is that AI-driven adaptation depends on the quality of the learner data and the pedagogical validity of the recommendation rules. If the task bank is weak or feedback is generic, the platform will not produce meaningful adaptation.

A third limitation concerns ethics and privacy. Student data must be collected transparently, stored securely, and used only for educational purposes. Learners and teachers should understand how recommendations are generated.

5. CONCLUSION

This article proposed an AI-driven adaptive learning platform model for enhancing programming competence in higher education. The model integrates student modelling, learning analytics, recommendation, adaptive feedback, competence assessment and teacher analytics into a closed feedback loop.

The proposed model treats programming competence as a multidimensional construct that includes theoretical knowledge, algorithmic thinking, code writing, debugging, problem solving and reflective regulation. This competence-based structure allows instruction and assessment to move beyond final answers and to evaluate the learning process itself.

The illustrative results show how an adaptive platform may support growth in programming competence by reducing the number of low-performing students, increasing the share of high-performing students, and improving practical components such as code writing and debugging. These results should be verified through real pedagogical experiments before journal submission.

Future research should focus on empirical validation of the model, development of explainable recommendation algorithms, multimodal learning analytics, automated code analysis, ethical data governance, and integration of AI-based platforms into institutional learning management systems.

ACKNOWLEDGEMENT

This manuscript draft was prepared as a methodological article template. Experimental values are illustrative and should be replaced by verified research data before submission.

REFERENCES

1. Cipriano, B. P., Fachada, N., & Alves, P. (2022). Drop Project: An automatic assessment tool for programming assignments. SoftwareX, 18, 101079. https://doi.org/10.1016/j.softx.2022.101079

2. du Plooy, E., Casteleijn, D., & Franzsen, D. (2024). Personalized adaptive learning in higher education: A scoping review of key characteristics and impact on academic performance and engagement. Heliyon, 10, e39630. https://doi.org/10.1016/j.heliyon.2024.e39630

3. Farhood, H., et al. (2025). Artificial intelligence-based personalised learning in education: A systematic review. Journal information retrieved from publisher index.

4. Kayumov Oybek Achilovich, Data-driven yondashuv asosida talabalar kompetentligini baholash metodikasi, Ta'limda istiqbolli izlanishlar - xalqaro ilmiy-metodik jurnal, № 4/2026 ISSN: 3060-527X, 55-63 b.

5. Kayumov Oybek Achilovich, Kompetentligini baholash mezonlari va indikatorlarini ishlab chiqish, "Maktabgacha va maktab ta'limi" jurnali, ISSN: 3060-4613, 2026-yil, aprel, 4(1)-son, 203-208 b.

6. Kayumov Oybek Achilovich, Web-loyihalash kompetentligini shakllantirishda interaktiv va loyihaviy (PBL) metodlardan foydalanish, Zamonaviy jamiyat va innovatsiyalar ilmiy-nazariy jurnal, ISSN: 3060-544x, № 4-son, 2026-yil, 63-69 b.

7. Kayumov Oybek Achilovich, Evaluating the Effectiveness of Adaptive Learning Systems in Developing Web Design Competence: An Empirical Study, International journal of pedagogics, ISSN: 2771-2281, Volume 06 Issue 03 (March) 2026, 362-367 b.

8. Kayumov Oybek Achilovich, Human–Ai Collaboration in Adaptive Learning Systems: A New Paradigm for Competence Development, American journal of social sciences and humanity research, ISSN: 2771-2141, Volume 06 Issue 03 (March) 2026, 126-131 b.

9. Kayumov Oybek Achilovich, Data-Driven Approaches to Enhancing Students' Web Design Skills Using Multimodal Learning Analytics, Current research journal of pedagogics, ISSN: 2767-3278, Vol.07 Issue03 2026, 46-52 b.

10. Kayumov Oybek Achilovich, AI Capital As A Predictor of Students' Digital and Web Design Competence in Higher Education, Journal of Social Sciences and Humanities Research Fundamentals, ISSN: - 2748-9345, Volume 06 Issue 03 (March) 2026, 94-99 b.

11. Kayumov Oybek Achilovich, A Conceptual Framework for Developing Web Design Competence Based on Multimodal Learning Analytics in Adaptive Digital Environments, European International Journal Of Pedagogics, ISSN: 2751-000X, Volume 06 Issue 03 (MARCH) 2026, 155-161 b.

12. Fortuna, A. (2025). Artificial intelligence in personalized learning: Pedagogical approaches and critical insights. Computers and Education: Artificial Intelligence.

13. Hariyanto, H., et al. (2025). Artificial intelligence in adaptive education: A systematic review of techniques for personalized learning. Discover Education.

14. Ifenthaler, D., & Yau, J. Y. K. (2020). Utilising learning analytics to support study success in higher education: A systematic review. Educational Technology Research and Development, 68, 1961–1990.

15. Katona, J., & Katonane Gyonyoru, K. I. (2025). AI-based adaptive programming education for socially disadvantaged students: Bridging the digital divide. TechTrends, 69, 925–942.

16. Keuning, H., Jeuring, J., & Heeren, B. (2019). A systematic literature review of automated feedback generation for programming exercises. ACM Transactions on Computing Education, 19(1), Article 3, 1–43.

17. Luxton-Reilly, A., Simon, Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., & Szabo, C. (2018). Introductory programming: A systematic literature review. ITiCSE 2018 Companion, 55–106.

18. Mamabolo, K. E., Mhlongo, S., Mosia, M., & Twinomurinzi, H. (2025). Learning analytics in computer programming education: A bibliometric scoping review. Interdisciplinary Journal of Education Research.

19. Vinueza-Morales, M., et al. (2025). Teaching programming in higher education: A bibliometric analysis. Frontiers in Education.

20. Zawacki-Richter, O., Marín, V. I., Bond, M., & Gouverneur, F. (2019). Systematic review of research on artificial intelligence applications in higher education: Where are the educators? International Journal of Educational Technology in Higher Education, 16, 39.