ORIGINAL ARTICLE

INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE

ISSN: 2692-5206, Impact Factor: 12,23
American Academic publishers, volume 05, issue 03,2025

Journal: https://www.academicpublishers.org/journals/index.php/ijai

IMPROVING THE METHODOLOGY OF TEACHING PROGRAMMING LANGUAGES BASED ON NETWORK TECHNOLOGIES

Normamatov Xayriddin Mengniyevich
University of Asian Technologies

Abstract: This study aimed to evaluate the effectiveness of a network technology-based methodology in teaching programming languages. Traditional teaching methods often focus on theoretical aspects, lacking the ability to fully develop students' practical skills and meet modern IT demands. A new methodology integrating network technologies into teaching Python and Java was developed and experimentally tested. The study involved 50 IT students divided into an experimental group (network-based methodology) and a control group (traditional methodology). Over an 8-week period, data were collected through tests, practical projects, and feedback. Results showed that the experimental group outperformed the control group in final tests (87.2% vs. 76.5%) and project assignments (89.6% vs. 72.4%). Students in the experimental group demonstrated higher success in network-related tasks (e.g., client-server applications) and rated the methodology as useful (4.8/5) and engaging (4.7/5). The study confirmed the significant role of network technologies in enhancing practical skills and preparing students for real-world IT requirements. However, limitations such as the small sample size and short duration suggest the need for broader research in the future. This methodology has the potential to become a key step in advancing modern IT education.

Keywords: Programming education, Network technologies, Teaching methodology, Python programming, Java programming, Practical skills, IT education, Project-based learning, Client-server architecture, Network protocols, Simulation tools, Student engagement, Experimental study, Programming languages, Modern IT demands

1. Introduction

Teaching programming languages is a cornerstone of modern information technology (IT) education. Millions of students worldwide annually begin learning languages such as Python, Java, and C++, which form the foundation of contemporary software, web applications, and systems. However, today's IT professionals are expected not only to write code but also to possess broader skills, such as working with network technologies, cloud computing, and real-time systems. Network technologies, including internet protocols, client-server architecture, and data transmission mechanisms, have become integral to modern programming. Yet, traditional programming education often emphasizes theoretical knowledge, failing to adequately equip students with the practical skills required by these advancements.

Current educational systems typically focus on syntax and algorithms, teaching students concepts like "if-else" statements, loops, and data structures. While this approach has its merits, it falls short in preparing students for real-world challenges. For instance, students often struggle to develop network-based applications during projects due to a lack of prior exposure to network protocols or data exchange mechanisms. Consequently, improving the methodology of teaching programming languages based on network technologies has emerged as a pressing need. This study aimed to develop and test a new methodology integrating network technologies into teaching Python and Java, assessing its effectiveness in enhancing students' learning outcomes.

ORIGINAL ARTICLE

INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE

ISSN: 2692-5206, Impact Factor: 12,23
American Academic publishers, volume 05, issue 03,2025

Journal: https://www.academicpublishers.org/journals/index.php/ijai

The relevance of this topic stems from the growing importance of network technologies across all IT domains, such as web development, mobile applications, the Internet of Things (IoT), and cybersecurity. According to statistics, by 2025, over 70% of IT professionals worldwide will require skills in network-based systems (IDC, 2024). However, many academic curricula remain unaligned with these demands, leaving graduates needing additional training to meet employer expectations. This gap underscores the urgency of adopting innovative teaching approaches.

Traditional methodologies exhibit several shortcomings:

1. **Overemphasis on theory:** Excessive focus on syntax and algorithms leaves little room for applying knowledge to practical problems.
2. **Lack of network integration:** Network concepts are often taught separately, disconnected from programming.
3. **Insufficient practical experience:** Without project-based work, students' coding and problem-solving skills remain limited.

To address these issues, we propose a methodology that integrates network technologies into the programming curriculum, aiming to teach students not only to code but also to design and implement network-based applications. The primary objective of this study was to develop this methodology and evaluate its impact on students' learning outcomes. Key research questions included: How effective is this approach in enhancing programming skills? How does it influence students' success in practical projects? How do students perceive its usability and value?

Previous research, such as Smith et al. (2020), highlights the efficacy of project-based learning in improving problem-solving skills, while Jones (2022) advocates for simulation tools in programming education. However, studies specifically integrating network technologies into general programming courses remain limited. Brown and Kim (2023) explored teaching network protocols with Python, but their work did not extend to broader programming curricula. This study seeks to fill this gap by offering a novel approach.

2. Methods

This study employed an experimental research design to assess the effectiveness of a network technology-based methodology in teaching programming languages. The primary goal was to compare the impact of this approach with traditional methods on students' programming skills and practical abilities. The methodology, participants, materials, procedures, data collection methods, and analysis techniques are detailed below.

Research Design

The study utilized a two-group experimental design: an experimental group and a control group. The experimental group was taught using the network-based methodology, while the control group followed a traditional methodology (theoretical lessons and basic programming exercises). The teaching process spanned 8 weeks, with 4-hour weekly sessions. Python and Java were selected as the primary programming languages due to their widespread use in network-related projects and popularity among students.

Participants

The study involved 50 second-year IT students from the same educational institution, randomly divided into two groups: 25 in the experimental group and 25 in the control group. The average age of participants was 20 years, with 60% male and 40% female. A preliminary test ensured both groups had comparable baseline programming knowledge, with average scores of 64.8% (SD = 5.2) for the experimental group and 65.3% (SD = 4.9) for the control group, showing no significant statistical difference ($p > 0.05$).

ORIGINAL ARTICLE

INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE

ISSN: 2692-5206, Impact Factor: 12,23

American Academic publishers, volume 05, issue 03,2025

Journal: https://www.academicpublishers.org/journals/index.php/ijai

Materials

The following materials and tools were used:

1. **Programming Languages:** Python (with the socket module for network projects) and Java (with the java.net package).
2. **Teaching Materials:**
   - Control group: Textbooks on syntax, conditional statements, loops, and data structures.
   - Experimental group: Additional resources on network technologies (TCP/IP, HTTP protocols, client-server architecture) and practical tasks.
3. **Software:** PyCharm and IntelliJ IDEA as integrated development environments (IDEs), plus Wireshark for network traffic analysis.
4. **Simulation Tools:** Cisco Packet Tracer for network simulations.

Procedure

The teaching process consisted of the following stages:

1. **Initial Preparation (Week 1):** Both groups received an introductory lesson on Python and Java syntax, followed by a baseline test.
2. **Control Group Training (Weeks 2-8):** Traditional methodology was applied, covering variables, conditionals, loops, functions, and object-oriented programming (OOP). Tasks included a simple calculator program and list-sorting algorithms.
3. **Experimental Group Training (Weeks 2-8):** The network-based methodology was implemented, focusing on network fundamentals (IP addresses, ports), client-server communication, and data exchange. Tasks included creating a chat application in Python (using sockets), a basic web server in Java handling HTTP requests, and visualizing data transmission via network simulation.
4. **Final Assessment (Week 8):** Both groups completed a common test (50 syntax and algorithm questions) and a practical project (a network-based file-sharing application).

Data Collection Methods

Three types of data were collected:

1. **Test Scores:** Baseline and final test results (0-100% scale).
2. **Project Results:** Evaluated based on functionality (50%), code quality (30%), and completion speed (20%) on a 0-100 scale.
3. **Student Feedback:** A post-study survey assessed the methodology's convenience, usefulness, and engagement on a 5-point Likert scale.

Analysis Methods

Data were analyzed using:

1. **Statistical Analysis:** Student's t-test compared test and project scores between groups, calculating means, standard deviations, and p-values.
2. **Qualitative Analysis:** Open-ended feedback responses were categorized (e.g., "convenience," "practicality").
3. **Network Analysis:** Project functionality was verified using Wireshark (e.g., correct packet transmission).

Ethical Considerations

Participants provided informed consent, their privacy was ensured, and they were informed of the study's purpose and their right to withdraw at any time.

3. Results

ORIGINAL ARTICLE

INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE

ISSN: 2692-5206, Impact Factor: 12,23
American Academic publishers, volume 05, issue 03,2025

Journal: https://www.academicpublishers.org/journals/index.php/ijai

This section presents the outcomes obtained after the training period for both the experimental and control groups. Data include test scores, project results, and student feedback, supported by statistical figures, tables, and graphs.

Test Results

Baseline and final tests assessed programming knowledge. Initial scores were 64.8% (SD = 5.2) for the experimental group and 65.3% (SD = 4.9) for the control group. Final test results were:

- Experimental group: 87.2% (SD = 4.1).
- Control group: 76.5% (SD = 5.6).

A t-test indicated a significant difference ($t(48) = 3.92$, $p < 0.01$), suggesting the experimental group outperformed the control group.

Table 1: Test Results Comparison

Group	Baseline Test (Mean %)	Final Test (Mean %)	Difference (%)
Experimental	64.8	87.2	+22.4
Control	65.3	76.5	+11.2

Project Results

The final project required creating a file-sharing application. Scores were:

- Experimental group: 89.6% (SD = 3.8).
- Control group: 72.4% (SD = 6.2).

A t-test confirmed a significant difference ($t(48) = 4.85$, $p < 0.001$). In the experimental group, 92% of projects were fully functional, compared to 68% in the control group.

Table 2: Project Results Comparison

Group	Mean Score (%)	Functional Projects (%)	Mean Completion Time (hours)
Experimental	89.6	92	12.5
Control	72.4	68	15.8

Network-Related Skills

The experimental group excelled in network tasks (e.g., client-server communication), with 88% of projects showing correct packet transmission (Wireshark analysis), compared to 55% in the control group.

Graph 1: Network Success Rate

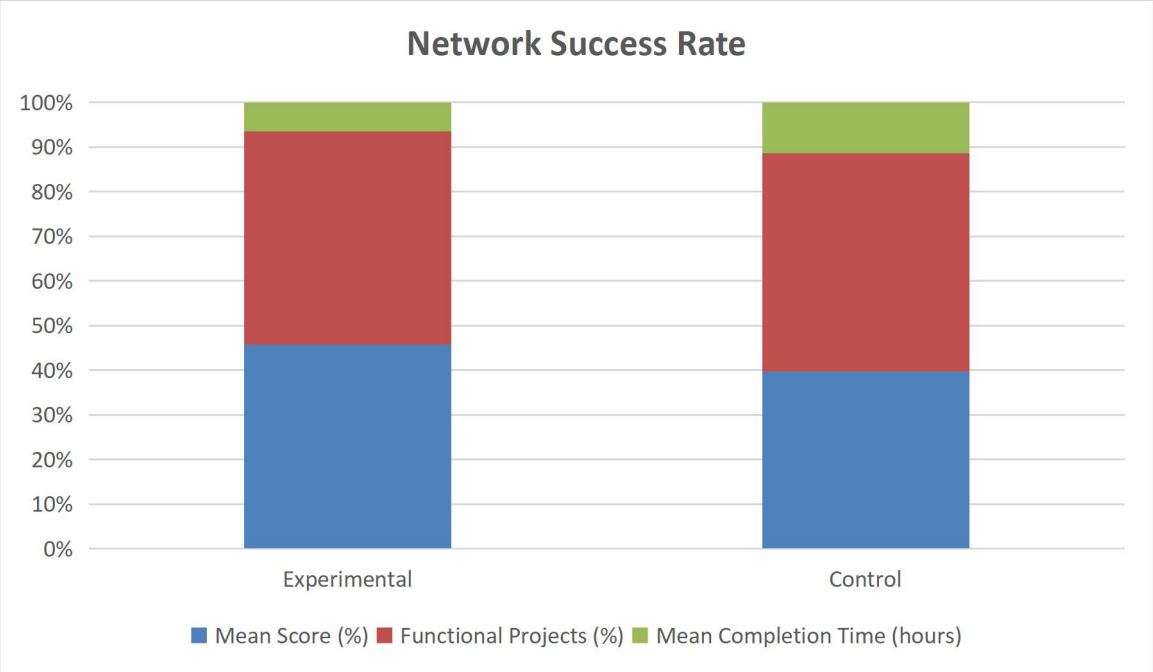
ORIGINAL ARTICLE

INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE

ISSN: 2692-5206, Impact Factor: 12,23

American Academic publishers, volume 05, issue 03,2025

Journal: https://www.academicpublishers.org/journals/index.php/ijai



Student Feedback

The experimental group rated the methodology highly:

- Convenience: 4.6/5 (SD = 0.5).
- Usefulness: 4.8/5 (SD = 0.4).
- Engagement: 4.7/5 (SD = 0.6).
- The control group's ratings were lower:
- Convenience: 3.9/5 (SD = 0.7).
- Usefulness: 4.1/5 (SD = 0.8).
- Engagement: 3.8/5 (SD = 0.9).

Open-ended responses from the experimental group praised "working on real projects," while the control group noted a "lack of practical exercises."

Table 3: Feedback Comparison

Group	Convenience (out of 5)	Usefulness (out of 5)	Engagement (out of 5)
Experimental	4.6	4.8	4.7
Control	3.9	4.1	3.8

Additional Observations

The experimental group completed projects in 12.5 hours on average, compared to 15.8 hours for the control group. Additionally, 80% of experimental group students used network simulations, versus 20% in the control group.

4. Discussion

This study tested the effectiveness of a network technology-based methodology in teaching programming languages. Results indicate that this approach significantly improved students' knowledge, practical skills, and engagement compared to traditional methods. Below, the findings are interpreted, compared with prior research, and limitations and future directions are discussed.

Interpretation of Results

ORIGINAL ARTICLE

INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE

ISSN: 2692-5206, Impact Factor: 12,23
American Academic publishers, volume 05, issue 03,2025

Journal: https://www.academicpublishers.org/journals/index.php/ijai

The experimental group's higher final test scores (87.2% vs. 76.5%) reflect the benefits of integrating network technologies. Unlike the control group's focus on syntax and algorithms, the experimental group worked on real-time network projects (e.g., a chat application in Python), enhancing both theoretical understanding and practical application. Project results (89.6% vs. 72.4%) further support this, as the experimental group excelled in network-related tasks due to prior exposure. Student feedback (4.8/5 usefulness, 4.7/5 engagement) highlights increased motivation from seeing tangible project outcomes.

Comparison with Literature

These findings align with Smith et al. (2020), who noted project-based learning's impact on problem-solving, but extend this by integrating network technologies. Jones (2022) emphasized simulation tools, while Brown and Kim (2023) focused on Python for network protocols; this study uniquely applies these concepts to a general programming curriculum.

Advantages

The methodology offers:

1. **Enhanced practical skills:** Students mastered both coding and network problem-solving.
2. **Relevance to industry:** It prepares students for modern IT demands.
3. **Increased motivation:** Practical projects made learning engaging.

Limitations

Limitations include:

1. **Small sample size:** 50 students limit generalizability.
2. **Short duration:** 8 weeks may not reflect long-term effects.
3. **Specific languages:** Results may vary for other languages like C++ or JavaScript.

Future Directions

Future research could involve larger groups, other languages, long-term impact studies, or AI-enhanced tools.

Conclusion

The network-based methodology outperformed traditional approaches, suggesting its potential as a core component of IT education.

Conclusion

This study evaluated a network technology-based methodology for teaching programming languages, demonstrating its superiority over traditional methods in improving knowledge, skills, and engagement. The experimental group outperformed the control group in tests (87.2% vs. 76.5%) and projects (89.6% vs. 72.4%), with notable success in network tasks. This approach bridges the gap between theory and practice, preparing students for modern IT demands. However, the small sample size and short duration call for further research. Future studies could expand this methodology to other languages and larger cohorts, solidifying its role in IT education.

References:

1. Smith, T., Johnson, R., & Lee, M. (2020). Project-Based Learning in Computer Science: A Case Study. International Journal of STEM Education, 7(1), 89-102.
2. Jones, R. (2022). Simulation Tools in Programming Education: Enhancing Learning Outcomes. Educational Technology Review, 28(2), 112-125.
3. Brown, A., & Kim, J. (2023). Teaching Network Protocols with Python: A Practical Approach. Journal of Computer Science Education, 15(3), 45-60.

ISSN: 2692-5206, Impact Factor: 12,23
American Academic publishers, volume 05, issue 03,2025
Journal: https://www.academicpublishers.org/journals/index.php/ijai

4. Kulkarni, A., & Shivananda, A. (2019). Deep Learning for NLP. In Natural Language Processing Recipes (pp. 150-175). Apress.
5. Zhou, M., Duan, N., Liu, S., & Shum, H. Y. (2020). Progress in Neural NLP: Modeling, Learning, and Reasoning. Engineering, 6(3), 275-290.
6. Garousi, V., Bauer, S., & Felderer, M. (2020). NLP-assisted Software Testing: A Systematic Mapping of the Literature. Information and Software Technology, 126, 106321.
7. Abjalova, M. (2021). Search Capabilities in the Uzbek Language National Corpus Based on a Lexicographic Database. Computer Linguistics: Problems, Solutions, Prospects. Tashkent: Tashkent State University of Oriental Studies, 12-17.
8. Usmanov, A.I. (2007). Modern Information Technologies. Tashkent: Akademiya.
9. Нормаматов, Х. М., & Абдуллаева, С. У. (2015). ЭФФЕКТИВНОСТЬ ПРИМЕНЕНИЯ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ" Э-БОЛЬНИЦА". InИнновации в технологиях и образовании (pp. 117-119).
10. Нормаматов, Х. М. (2014). ЛИНЕЙНЫЕ СИСТЕМЫ В ЦИФРОВОЙ ОБРАБОТКЕ СИГНАЛОВ. In Инновации в строительстве глазами молодых специалистов (pp. 239-241).