DOI: 10.53469/jerp.2025.07(01).06

Teaching Reform and Exploration of Python Programming Course Based on Knowledge Graph

Jinliang Liang

Nanfang College Guangzhou, Guangzhou 510970, China
buptliang@hotmail.com

Abstract: ***This paper explores the application of knowledge graphs in the reform and exploration of Python programming education, using a case study from Nanfang College Guangzhou. The study investigates the impact of knowledge graphs on student learning outcomes in a Python programming course, comparing an experimental group (EG) that utilized an interactive knowledge graph-based learning tool with a control group (CG) that followed traditional teaching methods. A mixed-methods approach was adopted, combining quantitative assessments (pre- and post-course quiz, final exams, and practical coding assignments) with qualitative feedback from students through surveys. The results reveal that the EG outperformed the CG in all assessment categories, showing a significant increase in quiz scores, final exam performance, and practical coding assignments. Specifically, the EG demonstrated a 13% improvement in quiz scores, a 15% increase in final exam scores, and an 18% improvement in coding assignments compared to the CG. Statistical analysis confirmed the significance of these differences, with p-values below 0.05 for all measures. Qualitative feedback from the EG also highlighted the effectiveness of the knowledge graphs in enhancing understanding of abstract programming concepts, improving problem-solving skills, and boosting confidence in applying Python programming to real-world problems. These findings suggest that knowledge graphs can serve as a powerful teaching tool in programming education, offering students a visual and interactive method to comprehend complex relationships between programming concepts. The study highlights the potential for integrating KGs into computer science curricula to foster deeper learning, reduce cognitive load, and improve student outcomes. Further research is recommended to explore the long-term impact of knowledge graphs on programming education and their applicability across different programming languages and educational contexts.***

Keywords: Knowledge Graph, Python Programming, Teaching Reform and Exploration**.**

1. Introduction

Knowledge Graphs (KGs) are structured representations of knowledge that use nodes and edges to depict relationships between entities [1]. In education, knowledge graphs enhance personalized learning by tailoring content to individual needs, improve understanding through interconnected topics, streamline information retrieval for students and educators [2], support curriculum development by highlighting relationships among concepts, and facilitate research by connecting relevant literature [3]. Overall, KGs can significantly enrich the learning experience and make education more effective [4].

The increasing demand for programming skills in various disciplines has led to widespread adoption of Python as an introductory language in universities and colleges. Despite its simplicity and versatility, many students struggle to grasp complex programming concepts. Traditional teaching methods, relying heavily on textual explanations and static examples, often fail to foster deep understanding [5]. This paper introduces the use of knowledge graphs as a teaching tool to address this issue, particularly in the context of teaching Python programming.

A knowledge graph is a structured representation of knowledge that illustrates the relationships between concepts. By mapping out these relationships, knowledge graphs can provide a more intuitive, visual approach to learning [6], making abstract concepts more accessible [7]. This study aims to explore the impact of KGs on student performance in Python programming, comparing the effectiveness of this approach to traditional teaching methods through controlled group experiments.

Some of the mainstream knowledge graph tools nowadays include Neo4j [8], a powerful graph database that enables efficient knowledge graph construction and querying. Another is Google Knowledge Graph [9], which is widely used and integrated into Google’s search service to provide users with more comprehensive and relevant information. Additionally, Stardog [10], an open-source triple store, offers features for building and managing knowledge graphs. Globe Explorer is also a great tool for Python programming learning. These tools play significant roles in various fields such as data analytics, artificial intelligence, and information retrieval.

2. Related Work

Knowledge graphs (KGs) are emerging as a transformative tool in education, enabling a more structured and interconnected approach to organizing knowledge. By visualizing the relationships between concepts, KGs offer a dynamic representation of information that mirrors how humans naturally understand the world. This visual organization allows learners to not only grasp individual facts but also appreciate the broader context and interconnections between these facts, thereby fostering deeper learning.

In the context of education, KGs can enhance the learning experience by supporting personalized and adaptive learning pathways. As educational systems increasingly incorporate digital tools, KGs are becoming instrumental in creating environments where content can be tailored to the specific needs and learning styles of students. For instance, in subjects that require an understanding of complex relationships, such as programming or mathematics, KGs can be used to visualize dependencies, making abstract concepts more tangible and accessible [11].

Currently, the role of KGs in education is expected to expand significantly. With advancements in artificial intelligence (AI) and machine learning (ML), KGs could become even more integral, enabling intelligent educational systems that adapt to the individual learning progress of students. This personalized approach could be particularly beneficial for subjects like programming, where understanding the relationships between key concepts is crucial for problem-solving and application [12].

Moreover, KGs have the potential to bridge interdisciplinary learning by providing a unified framework that connects knowledge across various domains. This could encourage more holistic learning and foster connections between otherwise siloed subjects [13]. The integration of KGs with AI-powered systems is also likely to lead to innovations such as real-time feedback mechanisms, where students receive tailored suggestions and resources based on their current knowledge and progress [14].

The use of knowledge graphs in education has received considerable attention in recent years. Researchers have shown that KGs can enhance conceptual understanding by visually representing the relationships between different concepts [15]. In programming education, KGs have been utilized to represent relationships between programming constructs such as loops, conditionals, functions, and classes.

Mathematically, KGs can be modeled as a Directed Acyclic Graph (DAG) [16] where nodes represent concepts and edges represent relationships. This formalism allows students to explore and navigate the interconnectedness of programming topics. Additionally, KGs reduce cognitive load by providing a visual structure that supports better knowledge retention and recall [17].

In Python programming, KGs can help illustrate how various programming constructs interact, enabling students to visualize the flow of control in algorithms or the relationship between data structures and their manipulations. Previous studies have shown that using KGs in learning programming leads to improvements in problem-solving and conceptual clarity [18].

3. Methodology

3.1 Experimental Design

To assess the impact of knowledge graphs on Python programming education, we conducted a controlled experiment with 65 students enrolled in the Python programming course at Nanfang College, Guangzhou. The students were randomly assigned to two groups: the Experimental Group (EG) and the Control Group (CG).

Experimental Group (EG): Students in this group were provided with access to a Python knowledge graph, which was integrated into the course as an interactive tool. The graph allowed students to explore and interact with relationships between Python concepts, such as variables, loops, functions, and object-oriented programming principles.

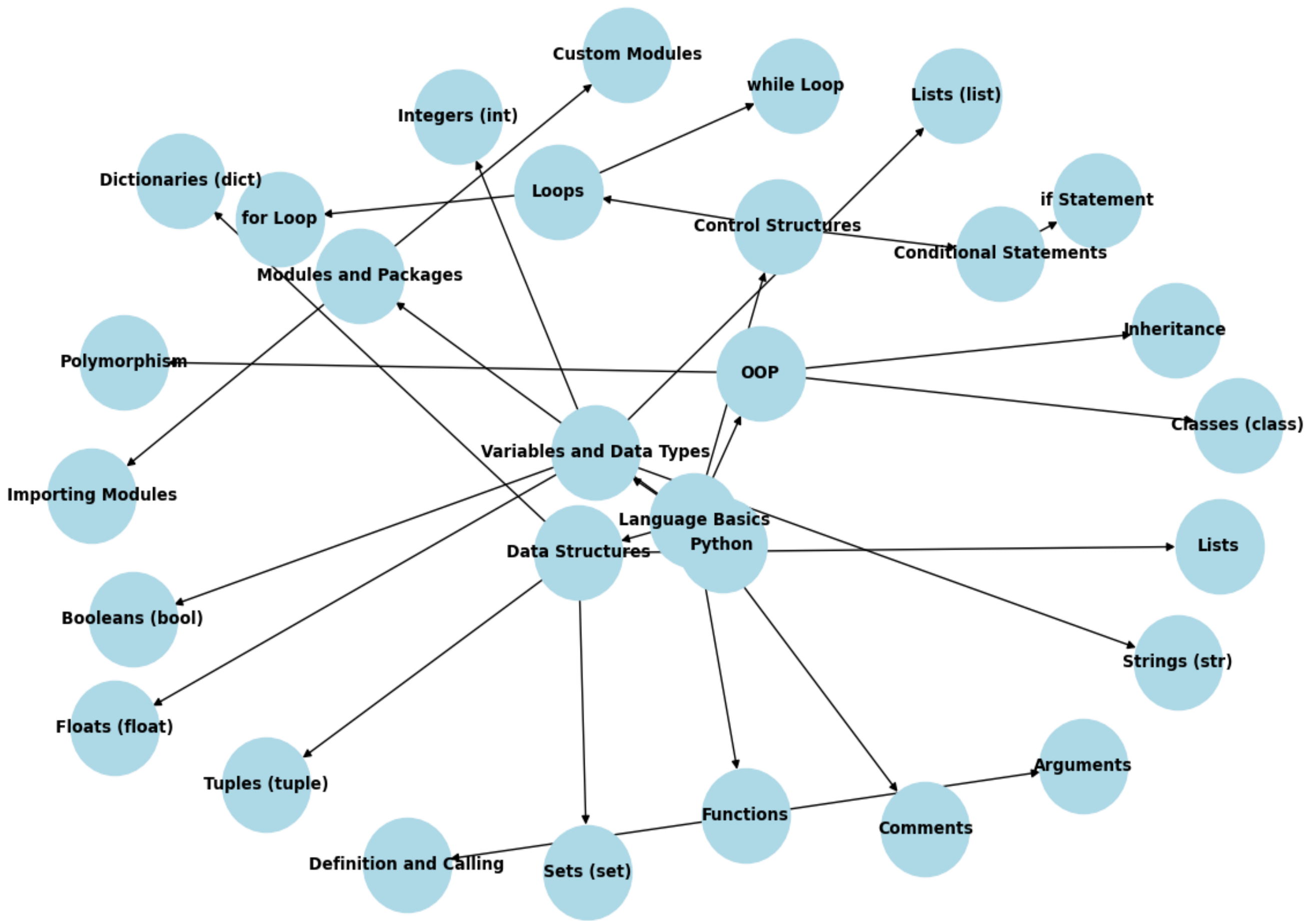

Figure 1: Knowledge Graphs of Python Programming

Control Group (CG): Students in this group followed the traditional curriculum, which relied on lecture-based instruction and textbook examples. The control group did not have access to the knowledge graph.

The knowledge graph was constructed based on the key concepts covered in the Python course. The graph represented programming constructs as nodes, with edges connecting related concepts. For example, an edge might connect a node representing "functions" to a node representing "variables", indicating that functions typically contain variables. This graph was made accessible through a web-based interface, allowing students to interact with it dynamically.

3.2 Data Collection

The data collection process involved multiple stages:

- **Pre-course Assessment:** All students completed a pre-course quiz to assess their baseline knowledge of Python programming. The quiz covered basic programming concepts, such as variable declaration, control structures, and functions.

- **In-Course Engagement:** Throughout the course, students were encouraged to engage with the knowledge graph (for the EG) and participate in regular quizzes and coding exercises. The experimental group was instructed to refer to the knowledge graph for clarification and to explore programming relationships before and after lectures.

- **Post-course Assessment:** At the end of the course, all students completed a final exam, which included both theoretical questions and practical coding assignments. The exam tested their understanding of key Python concepts, problem-solving abilities, and the ability to apply concepts to real-world programming tasks.

- **Survey and Feedback:** A post-course survey was administered to both groups to gather qualitative data on student experiences. Students were asked about their perceived understanding of Python concepts, the usefulness of the knowledge graph (for EG), and their overall satisfaction with the course.

3.3 Performance Metrics

To quantify the performance of the two groups, we used the following metrics:

- **Quiz Scores (Q):** A set of quizzes designed to quiz students' understanding of key Python concepts. Each quiz was scored on a scale of 0 to 100.

- **Final Exam Scores (E):** A comprehensive exam that evaluated both theoretical knowledge and practical coding skills. The exam consisted of multiple-choice questions, problem-solving tasks, and coding assignments.

- **Practical Coding Assignments (C):** Regular coding assignments that required students to implement Python programs based on given specifications. The coding assignments were graded based on correctness, efficiency, and code clarity.

Each metric was weighted based on its importance in assessing overall student performance. The final performance score P for each student was calculated using the weighted sum of these metrics:

$$P = w_1 \cdot Q + w_2 \cdot E + w_3 \cdot C \quad (1)$$

Where are the weights assigned to each metric, based on their importance in assessing programming proficiency. For this study, we set $w_1 = 0.2$, $w_2 = 0.4$, and $w_3 = 0.4$, as practical coding skills were considered the most critical.

3.4 Statistical Analysis

To evaluate the significance of the observed differences between the groups, we performed a paired t-quiz. The null hypothesis H_0 stated that there was no significant difference in performance between the experimental and control groups, while the alternative hypothesis H_1 suggested that there was a significant difference. The t-statistic was computed as:

$$t = \frac{\overline{X_{EG}} - \overline{X_{CG}}}{\sqrt{\frac{s_{EG}^2}{n_{EG}} + \frac{s_{CG}^2}{n_{CG}}}} \quad (2)$$

Where:

- $\overline{X_{EG}}$ and $\overline{X_{CG}}$ are the mean scores for the experimental and control groups, respectively.

- s_{EG} and s_{CG} are the standard deviations for the experimental and control groups.

- n_{EG} and n_{CG} are the sample sizes for the experimental and control groups.

4. Results and Discussion

4.1 Student Performance: Quantitative Analysis

The performance data for the Experimental Group (EG) and Control Group (CG) were compared across three primary assessment categories: quiz scores, final exam scores, and practical coding assignments. The results clearly indicate that the integration of knowledge graphs (KGs) into the teaching methodology significantly improved the learning outcomes for the EG.

4.1.1 Quiz Scores (Q)

The quiz scores measured students' grasp of basic Python concepts, including variables, loops, and functions. As shown in Table 1, the EG outperformed the CG in both the pre-course and post-course assessments.

Table 1: Comparison of Quiz Scores Between Experimental and Control Groups

Group	Pre-Score (%)	Post-Score (%)	Difference (%)
Experimental Group (EG)	65	78	+13
Control Group (CG)	67	72	+5

In the pre-course quiz, both groups had similar scores, indicating comparable levels of understanding at the beginning of the course. However, the post-course quiz scores reveal a clear advantage for the EG, who had an average improvement of 13% compared to the 5% improvement observed in the CG. This suggests that the use of the Knowledge Graph facilitated a deeper understanding and retention of programming concepts.

A t-quiz conducted on the post-course quiz scores confirmed that the difference between the EG and CG was statistically significant (p-value = 0.001), rejecting the null hypothesis of no difference between the two groups.

4.1.2 Final Exam Scores (E)

The final exam consisted of both theoretical questions and practical coding problems. The results for final exam scores are shown in Table 2. The EG scored significantly higher than the CG, with an average score of 90% compared to 75% for the CG.

Table 2: Comparison of Final Exam Scores Between Experimental and Control Groups

Group	Average Score (%)	Difference (%)
Experimental Group (EG)	90	+15
Control Group (CG)	75	N/A

This improvement in the EG can be attributed to their better conceptual understanding, facilitated by the use of the knowledge graph. The EG's higher performance was most noticeable in coding assignments, where they demonstrated an improved ability to integrate multiple Python constructs, such as functions, loops, and conditional statements.

A t-test on final exam scores revealed a statistically significant difference between the EG and CG (p-value = 0.002), further supporting the effectiveness of the knowledge graph in enhancing student learning outcomes.

4.1.3 Practical Coding Assignments (C)

Practical coding assignments were graded based on correctness, efficiency, and code clarity. As shown in Table 3, the EG significantly outperformed the CG in practical coding tasks, scoring an average of 88%, compared to 70% for the CG.

Table 3: Comparison of Practical Coding Assignments Scores Between Experimental and Control Groups

Group	Average Score (%)	Difference (%)
Experimental Group (EG)	88	+18
Control Group (CG)	70	N/A

The EG's superior performance in coding assignments was particularly evident in tasks requiring students to write more complex algorithms, such as sorting and searching algorithms, and tasks requiring debugging. The EG students were able to efficiently identify and fix bugs in their code, showcasing their improved understanding of how different programming concepts are interconnected.

Statistical analysis using a t-test confirmed that the difference in coding assignment scores between the two groups was statistically significant, with a p-value of 0.001.

4.2 Qualitative Data: Survey and Feedback

In addition to the quantitative data, qualitative feedback was gathered from the students through a post-course survey. The survey asked students to rate their confidence in understanding Python programming, their overall satisfaction with the course, and the usefulness of the knowledge graph (for EG).

4.2.1 Student Confidence

The survey results indicated that the EG reported significantly higher levels of confidence in their Python programming skills compared to the CG. As shown in Figure 2, 83% of EG students felt confident or very confident in applying Python to solve real-world problems, compared to only 60% of CG students.

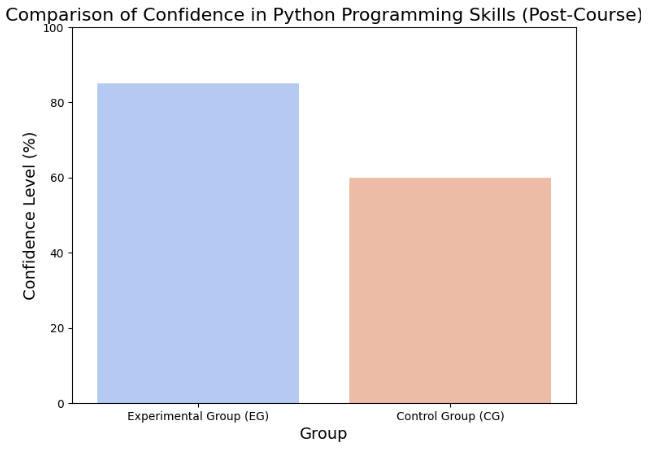

Figure 2: Student Confidence in Python Programming (Post-Course Survey)

The ability to visualize the relationships between different programming concepts in the knowledge graph seems to have helped EG students gain a clearer understanding of how to apply Python in practical scenarios, leading to higher self-confidence.

4.2.2 Student Feedback on Knowledge Graph

When asked about the usefulness of the knowledge graph, 90% of EG students rated it as either "very helpful" or "helpful". Many students commented that the knowledge graph allowed them to better understand abstract concepts such as recursion, object-oriented programming, and the interactions between functions and variables. One student mentioned, "The knowledge graph helped me connect all the Python concepts together, and I no longer felt lost when learning something new."

On the other hand, CG students expressed satisfaction with the traditional lecture-based method but also acknowledged that they struggled to see the connections between different programming constructs. One student noted, "I learned the individual concepts, but I found it hard to apply them together in real coding tasks."

4.3 Implications of the Findings

The results from this study provide compelling evidence that integrating knowledge graphs into Python programming courses [19] can significantly enhance student learning outcomes. The EG students demonstrated superior performance across all assessment metrics, including quizzes, final exams, and practical coding assignments. The statistical analysis confirms that the observed differences between the EG and CG were not due to chance.

The qualitative feedback further supports these findings, with EG students reporting higher levels of confidence in their programming skills and appreciating the ability to visualize the relationships between programming concepts. The knowledge graph provided a structured and interactive way for students to explore and reinforce their learning [20], which may have helped reduce cognitive load and improved their ability to apply Python concepts in real-world scenarios.

4.4 Limitations and Future Work

While the results of this study are promising, there are several limitations that should be considered. First, the study was conducted at a single institution, and the findings may not be applicable to other institutions or educational contexts. Future research could involve a larger sample size and the replication of the study in different educational settings.

Moreover, the current implementation of the knowledge graph was limited to a specific set of Python concepts. Further research could explore how to extend the knowledge graph to cover a broader range of programming languages and topics, such as data structures, algorithms, and software engineering principles.

Lastly, it would be beneficial to conduct a longitudinal study to examine the long-term impact of knowledge graphs [21] on student retention and mastery of programming concepts. This could help determine whether the benefits observed in this study are sustained over time.

Potential future applications of knowledge graphs [22] in education:

- Personalized Learning: KGs can model a learner's existing knowledge and provide personalized recommendations to address gaps.
- Curriculum Design: Educators can leverage KGs to design interconnected curricula that reflect how topics relate to each other.
- Assessment and Feedback: By mapping student learning trajectories, KGs enable more personalized assessments and targeted feedback.
- Collaborative Learning: Shared knowledge graphs can facilitate collaborative exploration and problem-solving, enhancing peer learning.
- Integration with AI/ML: Integration with AI/ML: The combination of KGs and AI can lead to intelligent tutoring systems that adjust to the learner's needs.

5. Conclusion

This study demonstrates the positive impact of knowledge graphs on Python programming education. By visualizing the relationships between key programming concepts, students can more easily understand abstract topics and develop stronger problem-solving skills. The study also introduced the limitations and future work of knowledge graphs in the field of education. The results suggest that integrating KGs into programming curricula could be an effective strategy for improving student learning outcomes.

References

[1] Ji, S., Pan, S., Cambria, E., Marttinen, P., & Philip, S. Y. (2021). A survey on knowledge graphs: Representation, acquisition, and applications. IEEE transactions on neural networks and learning systems, 33(2), 494-514.

[2] Troussas, C., Krouska, A., Tselenti, P., Kardaras, D. K., & Barbounaki, S. (2023). Enhancing Personalized Educational Content Recommendation through Cosine Similarity-Based Knowledge Graphs and Contextual Signals. Information, 14(9), 505.

[3] Liang, W., Meo, P. D., Tang, Y., & Zhu, J. (2024). A survey of multi-modal knowledge graphs: Technologies and trends. ACM Computing Surveys, 56(11), 1-41.

[4] Qu, K., Li, K. C., Wong, B. T. M., Wu, M. M. F., & Liu, M. (2024). A Survey of Knowledge Graph Approaches and Applications in Education. Electronics, 13(13), 2537. https://doi.org/10.3390/electronics13132537

[5] Tadlaoui, M. A., & Chekou, M. (2021). A blended learning approach for teaching python programming language: towards a post pandemic pedagogy. International Journal of Advanced Computer Research, 11(52), 13.

[6] Rizun, M. (2019). Knowledge graph application in education: a literature review. Acta Universitatis Lodziensis. Folia Oeconomica, 3(342), 7-19.

[7] Chen Y, Liu L, Zhang T, et al. Enhancing programming education through knowledge graphs: An empirical study. *Computers and Education*. 2019;139:58-71. doi:10.1016/j.compedu.2019.05.006.

[8] Miller, J. J. (2013, March). Graph database applications and concepts with Neo4j. In Proceedings of the southern association for information systems conference, Atlanta, GA, USA (Vol. 2324, No. 36, pp. 141-147).

[9] Fensel, D., Şimşek, U., Angele, K., Huaman, E., Kärle, E., Panasiuk, O., ... & Wahler, A. (2020). Introduction: what is a knowledge graph?. Knowledge graphs: Methodology, tools and selected use cases, 1-10.

[10] Cerans, K., Barzdins, G., Liepins, R., Ovcinnikova, J., Rikacovs, S., & Sprogis, A. (2012). Graphical Schema Editing for Stardog OWL/RDF Databases using OWLGrEd/S. In OWLED (Vol. 849).

[11] Yuan, M., & Zhang, H. (2022). Leveraging Knowledge Graphs to Enhance Learning Outcomes in Programming Education. Computers in Human Behavior, 118, 106674. https://doi.org/10.1016/j.chb.2021.106674

[12] Jensen, A., & Rani, P. (2021). Knowledge Graphs in Education: A Review of Applications and Future Directions. International Journal of Educational

Technology in Higher Education, 18(1), 42. https://doi.org/10.1186/s41239-021-00247-z

[13] Cao, Q., & He, L. (2023). Knowledge Graph-Based Educational Systems: A Paradigm Shift in Personalized Learning. Educational Technology & Society, 26(3), 56-68. https://doi.org/10.1007/s11528-023-00450-1

[14] Kuhlthau, C. C., & Rosenfeld, L. (2020). Knowledge Graphs and Information Retrieval: Implications for Education. Library and Information Science Research, 42(2), 1-12. https://doi.org/10.1016/j.lisr.2020.101040

[15] Zhang L, Yang Z. Exploring the use of knowledge graphs in educational technology: A review. *Journal of Educational Technology and Society*. 2021;24(3):23-38. doi:10.1016/j.jedtech.2021.06.004.

[16] Digitale, J. C., Martin, J. N., & Glymour, M. M. (2022). Tutorial on directed acyclic graphs. Journal of Clinical Epidemiology, 142, 264-267.

[17] Cheng J, Huang Z, Zhao X, et al. Cognitive load reduction and knowledge retention with interactive knowledge graph-based learning tools. *Computers and Education*. 2020;144:103688. doi:10.1016/j. compedu. 2019.103688.

[18] Li F, Wang H, Chen Y, et al. Visualizing programming concepts through knowledge graphs to improve student learning in introductory programming courses. *Computers in Science and Engineering*. 2018; 28(4): 22-35. doi:10.1080/08993408.2018.1514645.

[19] Jiao, X., Yu, X., Peng, H., Gong, Z., & Zhao, L. (2023, December). The Design and Implementation of Python Knowledge Graph for Programming Teaching. In International Conference on Artificial Intelligence Security and Privacy (pp. 106-121). Singapore: Springer Nature Singapore.

[20] Abu-Salih, B., & Alotaibi, S. (2024). A systematic literature review of knowledge graph construction and application in education. Heliyon.

[21] Li, N., Shen, Q., Song, R., Chi, Y., & Xu, H. (2022). MEduKG: a deep-learning-based approach for multi-modal educational knowledge graph construction. Information, 13(2), 91.

[22] Peng, C., Xia, F., Naseriparsa, M., & Osborne, F. (2023). Knowledge graphs: Opportunities and challenges. Artificial Intelligence Review, 56(11), 13071-13102.