Article │ Received 19 January 2025; Accepted 3 March 2025; Published 12 March 2025
https://doi.org/10.55092/pcs20250004

# High-quality giving-a-fish or teaching-to-fish: AI service empowerment in university programming education research

**Yunyan Liao[1,2], Xinyue Shu[2], Rongdan Shi[2], Yang Jiang[2], Qing Huang[1,2] and Fenge Wang[3,*]**

[1] College of Computer and Information Engineering, Jiangxi Normal University, Nanchang, Jiangxi
[2] College of Digital Industry, Jiangxi Normal University, Shangrao, Jiangxi
[3] College of Foreign Languages, Jiangxi Normal University, Nanchang, Jiangxi

* Correspondence author; E-mail: 202341600145@jxnu.edu.cn.

**Abstract:** Traditional university programming education follows a "theory-practice" model, aiming to cultivate students' computational thinking. However, research has shown that teachers can only impart pseudo-code and use it for extremely simple cases, limiting the scope of practice. In response to this issue, a "practice-theory-practice" teaching model has been proposed. The research team developed two AI service tools, Cheng Xiaoming and Code Monkey, based on the artificial intelligence generation platform Prompt Sapper, which combine providing specific code examples (teaching people to fish) with offering methods for writing code (teaching people how to fish). Through designed experimental schemes, the traditional university programming education model was compared with the new AI-powered programming education model. The new teaching model, assisted by AI, has been validated in practical teaching applications to enhance programming skills and cultivate computational thinking, demonstrating its effectiveness and impact.

**Keywords:** information technology in education; computational thinking; artificial intelligence; programming education

## 1. Introduction

In the early stage, the computer programming language teaching of primary and university in China was mainly in information technology courses. Entering the era of intelligence, school education was faced with the severe challenge of teaching paradigm transformation: intelligent machines designed by human beings had already been in deep learning, while there was still a large degree of "shallow learning" in school classrooms. The form of programming education mainly relies on STEM education, maker education, student associations and science and technology competition and other carriers, that is, teachers first teach the theoretical knowledge of information technology in the classroom, and then students use STEM Education, Maker Education, Student Associations and Science and Technology competition to conduct practical operations, so as to learn from "Doing". However, this teaching mode of "Theory—Practice" with theory before practice still stays at the ideological level of teaching

programming knowledge, and some people even think that the accumulation of knowledge points is the core of programming 0. This teaching mode is difficult to cultivate students' computational thinking, logical thinking, problem-solving thinking and innovative thinking.

## 2. Research status analysis

As early as a few years ago, the craze of artificial intelligence began to spread, becoming a widely used and deeply researched field and showing vigorous development. With the arrival of a brand new era of AI, programming education has become a hot spot of education research at home and abroad. Research indicates that developed nations like the United States, the United Kingdom, Australia, and the European Union have integrated youth Physical Education (PE) into their national strategic planning and action plans.

Fortunately, current AI technology has advanced to the level of 'No-Code programming'. It can automatically generate code, thereby assisting people in learning programming. Leveraging AI's capability to generate code, we propose a new teaching model: 'Practice-Theory-Practice.' Initially, students use AI to generate code, directly observing its execution. This approach stimulates their interest in learning and allows them to experience the practical fun and value of programming firsthand, rather than just being taught programming code passively. This method breaks free from the traditional limitation of first learning dry, theoretical knowledge and code syntax. Subsequently, AI personalizes the teaching of code syntax and logic based on individual students' learning progress and needs, enhancing their theoretical understanding. Finally, equipped with AI tools as their 'fishing gear,' students can apply their knowledge practically, thereby consolidating and enhancing their computational thinking and programming skills 0.

## 3. Difficult problems and thinking of the development of traditional middle school programming education

Many scholars generally agree that cultivating computational thinking, logical thinking, and innovative thinking are the core concepts and objectives of university programming education. For instance, Professor Yizhen Zhou posits that computational thinking encompasses a range of thinking activities that apply the fundamental concepts of computational science to problem-solving, system design, and understanding human behavior. Professor Jing Yang maintains that the essence of programming education lies in fostering students' innovative thinking and ability to solve practical problems, rather than merely teaching them how to write code. It's about nurturing computational thinking, innovation, and an understanding of the correct approach for the future intelligent era. In pursuit of this educational objective, most schools adopt a traditional pedagogical approach in their university programming courses, primarily using the lecture method. Teachers employ various means such as multimedia, blackboards, and programming tools to teach in alignment with the programming curriculum's knowledge system. This pedagogical approach predominantly depends on textbooks and instructor guidance. Nonetheless, due to time constraints and the intricacy of programming language syntax, educational resources and programming tools frequently employ pseudocode for language instruction. As a result, the instructional effectiveness tends to be confined to conveying programming concepts and coding logic to students.

Therefore, the prevailing teaching model, which adheres to the 'Theory-Practice' paradigm, falls short in fulfilling the needs of modern students in programming education. It does not effectively accomplish the objective of nurturing students' computational thinking within the realm of teaching practice (Figure 1). To rectify this issue, domestic scholars such as Sun Lihu, Jiang Yan, and Bing Jie have developed a variety of programming education models, each tailored to specific programming content and platforms. Nonetheless, research in this domain remains somewhat limited and fragmented. Consequently, this study concentrates on delineating a novel teaching model that more effectively caters to the educational requirements of university students in programming 0. Additionally, it examines the potential of AI technology in facilitating teaching at various stages.

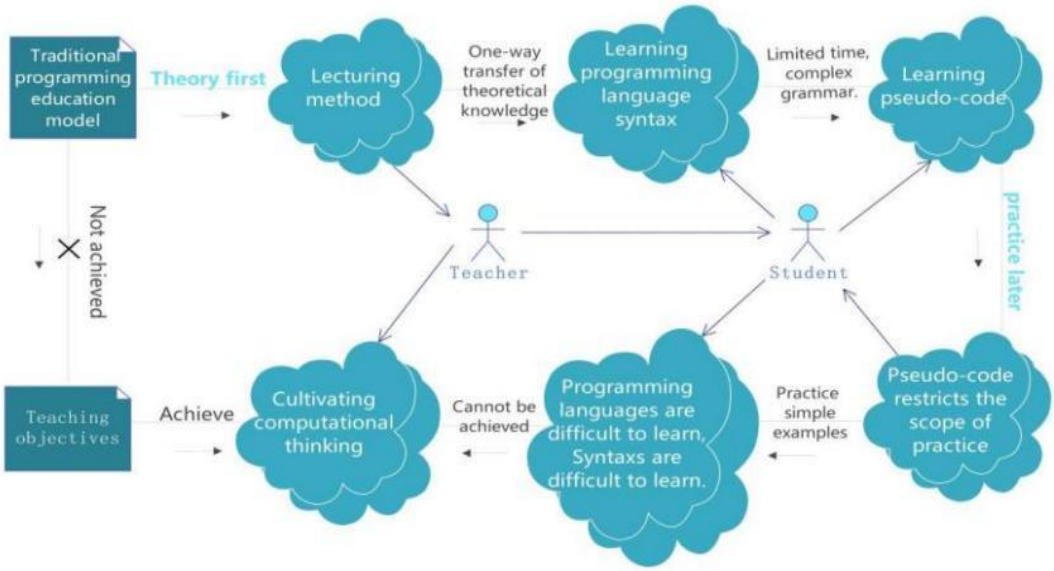


**Figure 1.** Traditional programming education development model.

*Thinking under the help of new teaching*

The development of Artificial Intelligence (AI) is one of the most notable advancements in the technology sector in recent years. The progress of AI in the field of education has been significant, bringing about many new possibilities and opportunities. There has been a proliferation of practical cases in AI-assisted programming education: for instance, Grasshopper, developed by Google, is a mobile application that utilizes AI technology to provide an interactive programming learning experience; AI4K12 aims to introduce AI education into middle school classrooms and has developed a set of AI courses and educational resources tailored for secondary school students, using intelligent educational systems and machine learning algorithms to offer personalized learning support and practical projects. These practical cases offer new insights for our research. Based on the challenges outlined in Section 2.1 and the broader context of AI technology, we propose the following three areas of consideration:

How can AI services be utilized to implement a "practice-theory-practice" programming learning model?

How can AI technology assist middle school students in writing programming languages?

How can AI technology aid middle school students in learning code?

In the following Section 3, we will delve into these three considerations and present solutions.

## 4. Services empower higher education programming education: teach people to fish and teach people to fish

### *4.1. The teaching mode of "practice-theory-practice" is proposed*

We investigated the specific implementation of the solution to the problem: When utilizing AI-related tools for instruction, AI provides specific programming code and demonstrates it in the programming tool, offering students a reference to help them engage directly in practice 0. Students encounter problems during practice, which are then feedback to the instructor. Next, the instructor delivers theoretical explanations and teaches the syntax and logic of the code, enabling students to enhance their theoretical knowledge based on their understanding of the code. Finally, with the assistance of AI services as a "fishing tool," students apply their learned knowledge to attempt autonomous programming and practice programming solutions to specific problems. This forms a new programming education model, namely the "Practice-Theory-Practice" model.

### *4.2. The new teaching mode is realized*

To implement the proposed new teaching model as shown in Figure 2, our laboratory has developed a series of AI teaching assistant programming products, including Code Monkey and Cheng Xiaoming, on the Prompt Sapper platform. Code Monkey is designed to focus on the instruction and writing of code, reinforcing students' programming knowledge by directly displaying specific code through the tool, laying the groundwork for code writing and providing students with the "fish." Cheng Xiaoming, on the other hand, is aimed at focusing on the learning of code, teaching students how to learn code and giving them the "fishing rod." These two AI teaching assistant programming "fishing tools" offer middle school students an intuitive, simple, and engaging way to learn and write code, enhancing the programming education for middle school students 0.

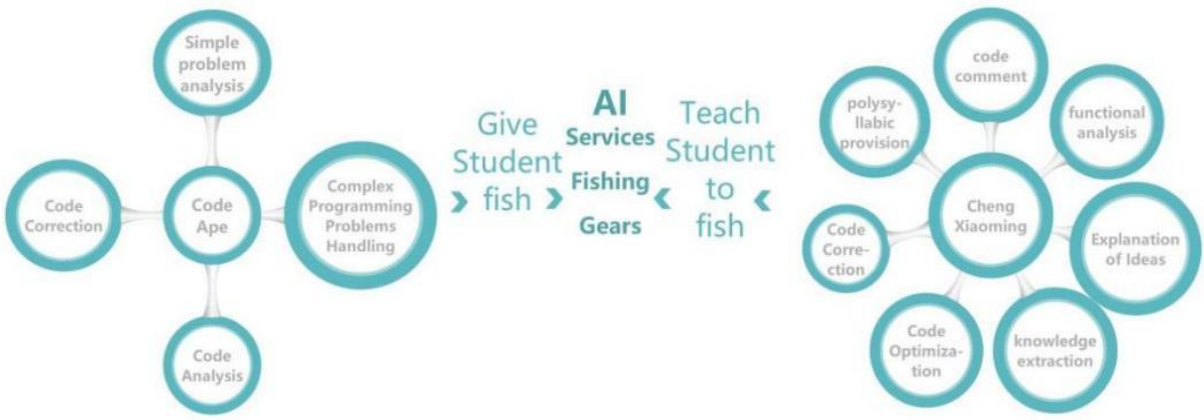


**Figure 2.** Correspondence between "fish" and "fishing" of Code Ape and Cheng Xiaoming and their core functions.

By using the AI assistant Code Monkey to help write code and Cheng Xiaoming to assist in learning code, we have overcome two major challenges in the traditional development of programming education in middle schools, realizing a teaching model of "Practice-Theory-Practice." in Figure 3. This means AI services empower middle school programming education: through the AI-generated platform Prompt Sapper and its AI service products, Code Monkey and Cheng Xiaoming, we assist middle school students in their programming education, achieving a combination of teaching people to fish and teaching people how to fish.

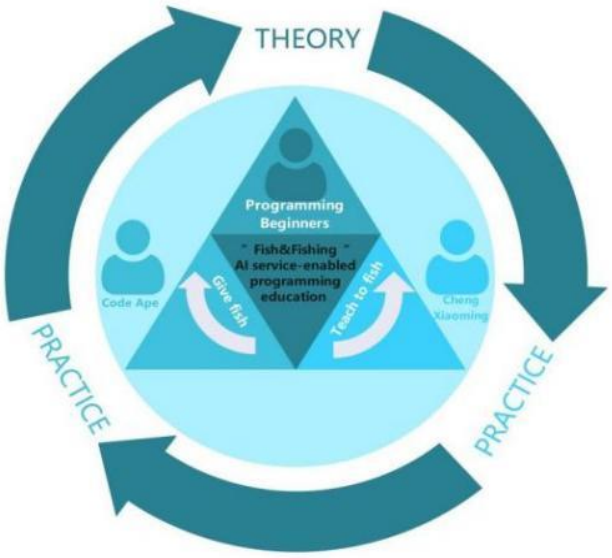


**Figure 3.** The "pyramid" of AI service-enabled secondary programming education.

*4.3. The cultivation of computational thinking*

Research has shown that early traditional programming education in middle schools had issues such as an incomplete cultivation of computational thinking and uneven effectiveness in the use of programming tools. The study's proposed Cheng Xiaoming and Code Monkey take into account the age appropriateness of the users and are capable of cultivating computational thinking from multiple perspectives and aspects, achieving the teaching objectives of programming education. This demonstrates the value of AI services in empowering middle school programming education. The following section will provide a theoretical argument for this.Based on the introduction of Cheng Xiaoming and Code Monkey's functions in Section A of the previous text, the process of middle school students using Cheng Xiaoming to learn code and Code Monkey to write code can be summarized into six stages: idea explanation, code error correction and optimization, code analysis, programming problem-solving, providing multiple solutions, and knowledge extraction, as shown in Figure 4. This process shares similarities with the problem-solving flow of computational thinking 0.

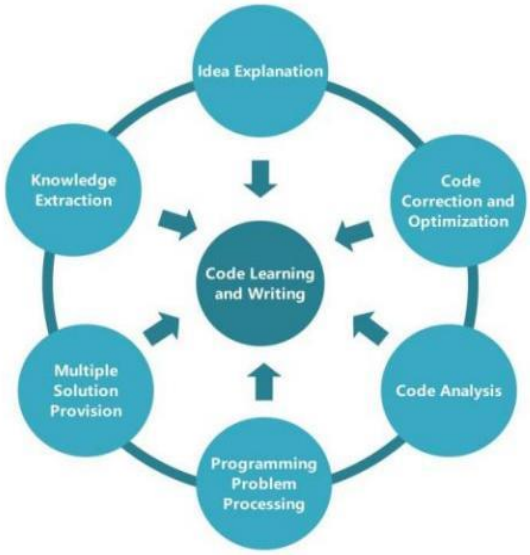


**Figure 4.** Main aspects of learning and writing code by using Cheng Xiaoming and Code Ape.

## 5. Experiment

After conducting a baseline test of programming skills among students from a major at a university in Nanchang, the research team selected two classes with similar programming foundations, totaling 100 students as the research subjects. Class A, with 51 students, was designated as the experimental group, while Class B, with 49 students, served as the control group. The average scores of the baseline test for both Class A and Class B were the same to ensure fairness in subsequent tests. From the selected 100 students, 70 who were currently studying or intended to learn programming languages other than Python were chosen as Group C. The baseline test scores of these 70 students were all around the average.

### *5.1. Experimental design*

Programming ability is the capacity to apply computational thinking to practical program writing and problem-solving, which can foster the development of computational thinking. Therefore, the objective of this experiment by the research team is to comparatively evaluate the effectiveness of traditional middle school programming education and AI-powered programming education, where Code Monkey provides the "fish" and Cheng Xiaoming teaches how to "fish," in actual teaching. The evaluation metrics include enhancing programming ability and cultivating computational thinking. To this end, we have designed an experiment lasting four months, from early February 2023 to early June 2023, to compare the effects of these two methods on these two metrics.

For Groups A and B, the Python course was the subject of the experiment. In the experimental group, teachers applied Cheng Xiaoming and Code Monkey to the teaching of the Python course in their class, allowing students to use these tools for learning and practicing programming both before and after class. In the control group, teachers used traditional teaching methods to instruct the Python course. Both groups received a total of 36 hours of instruction, with 2 hours per week. At the end of the experimental period, we conducted different tests to assess improvements in programming ability and the cultivation of computational thinking.

For Group C, these 70 students were allowed to use Cheng Xiaoming and Code Monkey to learn an additional one, two, or three programming languages outside of their regular Python course, without affecting normal teaching. The experimental period was the same as above. At the end of the experiment, we tested the learning effects of these 70 students in multiple programming languages.

Experimental Design for Improving Programming Skills: This study will test the programming ability of experimental students from four aspects: programming ability, code quality, basic knowledge of programming grammar, and multi-language learning effect.

We designed a set of Python programming test questions, which covered five categories of Python related knowledge, including basic grammar and data types, functions and modules, string processing, file processing, data structure and algorithm. Each category set 10 questions and 50 programming questions in five types, to test the programming ability of the two groups of students. At 14:00–18:00 PM, June 5, 2023, we conducted online testing with the students in groups A and B. We could compile the source code submitted by students online and generate executable files. It also evaluates the quality of the code from correctness, runtime, memory consumption and return results based on set test cases. In addition, we also designed a Python test paper to master the basic knowledge of Python programming and grammar in both groups A and B.

Experimental Design for Cultivating Computational Thinking: In this study, pre-tests and post-tests on computational thinking were conducted for groups A and B using computational thinking test questions. The evaluation framework for computational thinking utilized in this study is a localized evaluation system developed by Xingzhi Chen. The primary indicators of this framework are categorized into computational thinking attitude and computational thinking skills. The computational thinking attitude includes three secondary indicators: emotional attitude, thinking quality, and cooperative learning. The computational thinking skills encompass five secondary indicators: (abstraction, lysis, generalize, assess, algorithm).

### *5.2. Data collection and analysis*

The experimental data for Cheng Xiaoming, Code Ape and enhancing programming skills consist of four groups: programming design ability—Python programming test accuracy, code quality—Python programming test source code quality, programming syntax knowledge—Python exam scores, and the effectiveness of learning multiple languages - average scores in various programming languages for each subject.

#### *5.2.1. Programming design ability - analysis of Python programming test accuracy*

Firstly, we compiled the distribution of correctness rates for the five major question types in the Python programming test for both groups of students in the experiment (see, for example, Figure 5). The “Basic Syntax and Data Types” category includes questions on variable definition and usage, data type conversions, conditional statements (if-else), loop statements (for loop, while loop), list, dictionary, set operations, and other fundamental syntax and data type manipulation questions. The “Functions and Modules” category encompasses questions on function definition and invocation, parameter passing, handling return values, function scope, and related topics. It also includes questions on module importing, usage, and module creation. The “String Manipulation” category covers topics such as string concatenation, slicing, replacement, character or substring search, case conversion, and other related

questions. The “File Processing” category involves questions on file reading and writing, parsing and processing CSV or JSON formatted data, log file analysis, and related topics 0. The “Data Structures and Algorithms” category comprises operations on data structures such as lists, dictionaries, sets, sorting algorithms, search algorithms, recursion, and related topics.

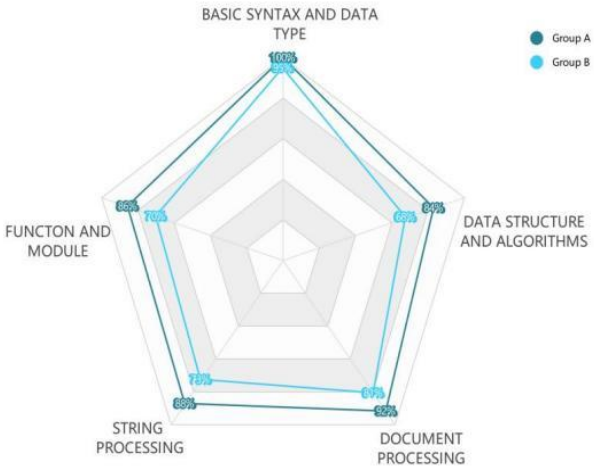


**Figure 5.** Radar chart of correctness rates for different question types in the Python programming test.

*5.2.2. Code quality - analysis of Python programming test source code quality*

We collected the source code submitted by the experimental students and combined it with the evaluation data from the platform to analyze the code quality of the two groups based on five indicators: maintainability, reliability, reusability, security, and standardization (Figure 6).

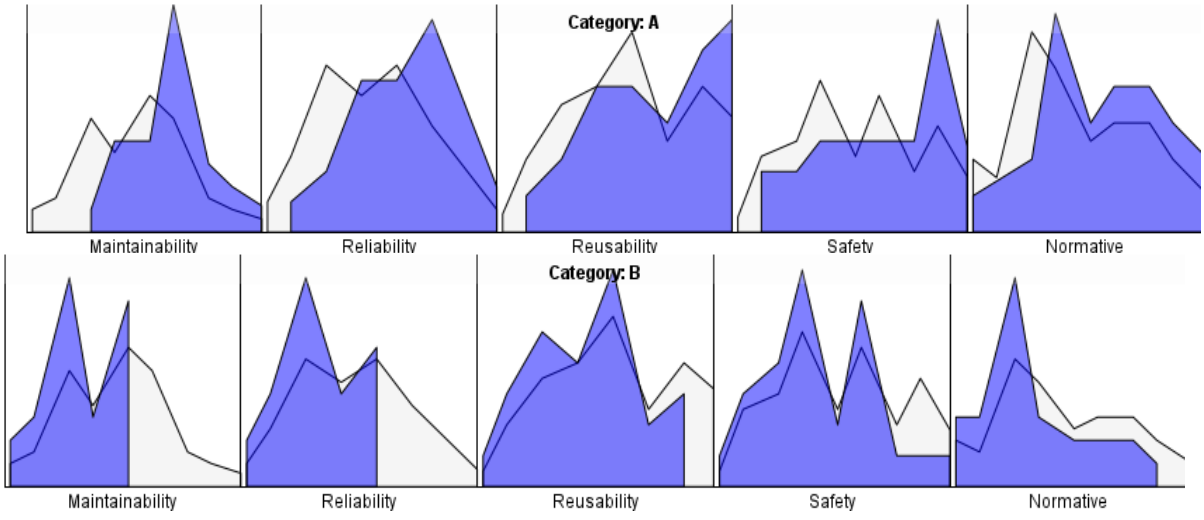


**Figure 6.** Subgroup comparison of code quality analysis for the two groups.

*5.2.3. Experimental data for cultivating computational thinking*

Firstly, examining the mean scores of the two groups from Table 1, the mean score for Group A is 75.2549, while the mean score for Group B is 68.3673, indicating that Group A significantly outperforms Group B. Secondly, an independent samples t-test was conducted in conjunction with

the data from Table 2. With a Significance 0.379, which is greater than 0.05, and assuming equal variances, the two-sided P-value is 0.002, which is less than 0.05, suggesting that a statistically significant difference exists between the two groups' scores. The data from both groups indicate that the experiment has also played a beneficial role in reinforcing students' mastery of Python programming syntax.

**Table 1.** The statistical mean scores of the two groups.

| | Group | N | Mean | Standard Deviation | Standard Error of the Mean |
|---|---|---|---|---|---|
| Score | A | 51 | 75.2549 | 9.81803 | 1.3748 |
| | B | 49 | 68.3673 | 11.27071 | 1.6101 |

**Table 2.** Independent samples T for two groups' scores.

| | | Levene's Test for Equality of Variances | | T for Equality of Means | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | F | Significance | T | Degree of freedom | One-sided P-value | Two-sided P-value | Difference in means | Standard Error Estimate |
| Score | Assuming isotropic | 0.78 | 0.379 | 3.262 | 98 | <0.001 | 0.002 | 6.887 | 2.111333 |
| | Not Assuming isotropic | | | 3.253 | 95.019 | <0.001 | 0.002 | 6.887 | 2.11719 |

df: degrees of freedom; P: significance 0.379 > 0.05; T: t-test.

Before the experimental course, we administered a computational thinking test to the two groups to assess their initial levels. Following the course, we conducted another computational thinking test to measure the improvement in computational thinking skills for both groups. The computational thinking questions used in our assessment were derived from the "Bebras International Computational Thinking Challenge" questions from previous years. As the most prestigious computational thinking competition in the world, the Bebras challenge ensures that the questions are authoritative and reasonable, providing a scientific method to evaluate students' computational thinking abilities. Figure 7 illustrates the changes in computational thinking skills across the five dimensions—abstraction, decomposition, pattern recognition, evaluation, and algorithmic thinking—for the two groups of students.

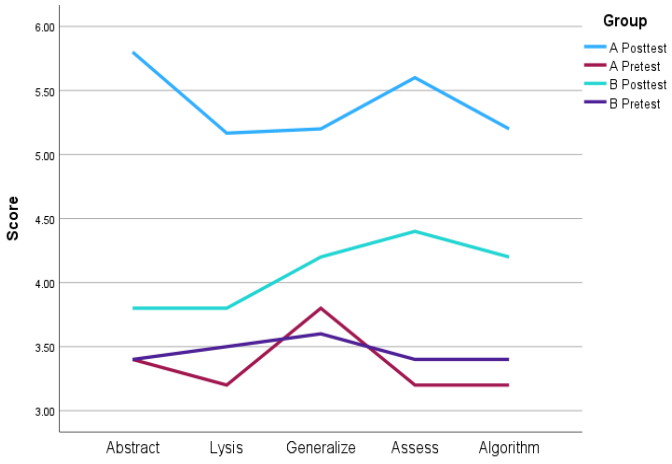


**Figure 7.** Changes in pre-tests and post-tests of students' computational thinking in the two groups.

## 6. Conclusion

Based on the data collection and analysis, we have derived the following two experimental conclusions and educational implications for programming education in secondary school students:

*6.1. The auxiliary Python courses taught by Cheng Xiaoming and Code Monkey can enhance the programming abilities of beginners.*

With respect to program design, code quality, and syntax knowledge, the empirical data reveals a superior performance by the experimental group compared to the control group. This improvement is likely due to the introductory level of programming beginners, necessitating innovative instructional strategies and tool-supported teaching to facilitate their transition from beginner to advanced levels. Traditional programming pedagogies are inadequate to fulfill this requirement (as discussed in Section 2). The application of artificial intelligence, particularly AI-assisted programming tools, has been proven effective in bolstering students' programming capabilities, as indicated by this research. Consequently, it is essential for educators to proactively integrate AI tools into programming education, thereby establishing a robust platform for the cultivation of students' programming skills.

*6.2. The auxiliary Python courses taught by Cheng Xiaoming and Code Monkey have a significant effect on cultivating computational thinking in programming beginners.*

Both Cheng Xiaoming and Code Ape have the functions of code optimization, code annotation, and code error correction, *etc.* Students in the experimental group learn and write code by using both of them, and the quality of the designed code is significantly better than that of the control group. However, code quality is often neglected in the programming teaching of secondary school students, focusing more on the teaching of theoretical knowledge. However, beginners are at the stage of laying the foundation, and the standardized learning of code quality assessment is the basis for further subsequent learning 0. Therefore, the importance of code quality should be emphasized in programming education for secondary school students to ensure that they are able to write high-quality, easy-to-read and maintainable code in practice.

This paper aims to realize the teaching goal of modern programming education, and puts forward the teaching mode of "Practice-Theory-Practice". In order to solve the problems encountered in this model, we innovatively chose the use of artificial intelligence technology as the breakthrough point, closely combining the pain points of traditional programming education with the technical characteristics of artificial intelligence. These tools can optimize language communication and effectively assist students in programming learning and practice. Using Cheng Xiaoming and the code little ape to assist students 'programming learning has played an obvious effect, which can better improve students' programming ability and cultivate students' computing thinking, and truly achieve the teaching goal. Based on the inspiration of this research and the rapid development of artificial intelligence technology, in the future programming education practice, we will study more AI-assisted programming products on the Prompt Sapper platform, so that students can next become the programming leaders with the help of these "fishing gear".

## Acknowledgments

This work was funded by the Jiangxi Province Higher Education Teaching Reform Research Project with grant number JXJG-23-2-42.

## Conflicts of interests

The authors declare no conflict of interest.

## Authors' contribution

Conceptualization, Yunyan Liao and Qing Huang; methodology, Fenge Wang; Rongdan Shi; validation, Xinyue Shu, Yang Jiang; formal analysis, Xinyue Shu; investigation, Xinyue Shu; resources, Fenge Wang; data curation, Xinyue Shu; writing—original draft preparation, Xinyue Shu; writing—review and editing, Yunyan Liao; visualization, Xinyue Shu; supervision, Qing Huang; project administration, Yunyan Liao; funding acquisition, Yunyan Liao. All authors have read and agreed to the published version of the manuscript.

## References

[1] Liu L, Dong Y. Research on teaching models in artificial intelligence education (In Chinese). *Fujian Comput*. 2022, 38(09):115–118.

[2] Li Y, Yu C, Cui Y. TPCaps: a framework for code clone detection and localization based on improved CapsNet. *Appl. Intell*. 2022, 53(13):16594–16605.

[3] Cañadas L. Contribution of formative assessment for developing teaching competences in teacher education. *Eur. J. Teach. Educ.* 2023, 46(3):516–532.

[4] Wu J. The training strategy of creative talents in the teaching of computer software development (In Chinese). *Adult High. Educ.* 2023, 5(6):2523–5826.

[5] Macrides E, Miliou O, Angeli C. Programming in early childhood education: a systematic review. *Int. J. Child-Comput. Interact.* 2022, 32:100396.

[6] Huang Y, Liu W. How to improve the teaching efficiency of higher mathematics in private colleges. *Adv. Educ. Technol. Psychol.* 2021, 5(11):2371–9400.

[7] Wang K, Ruan S, Fang Y. Core issues of programming languages in integrated teaching method (In Chinese). *Comput. Educ.* 2022(09):162–165.

[8] Pham STH, Sampson PM. The development of artificial intelligence in education: a review in context. *J. Comput. Assist. Learn* 2022, 38(5):1408–1421.