

Research on the Application of AI Large Models in Python Teaching

Yan Ding^{1*}

¹ Guangzhou College of Technology and Business, Guangzhou, China

Email Address

407730436@qq.com (Yan Ding)

*Correspondence: 407730436@qq.com

Received: 25 October 2025; **Accepted:** 12 November 2025; **Published:** 12 December 2025

Abstract:

With the rapid development of Generative Artificial Intelligence (AIGC) technology, the education sector is undergoing a profound transformation from traditional “knowledge impartation” to “capability cultivation”. As a general-purpose and widely applicable programming language, Python teaching faces the challenge of how to adapt to the talent cultivation needs of the intelligent era. Based on constructivist learning theory and the concept of human-machine collaborative education, this paper proposes an innovative framework for Python teaching that integrates AI technology, covering five core stages: pre-class preparation, classroom interaction, post-class exercises, project practice, and assessment evaluation. It also clarifies the ethical norms for AI use and the path for teacher role transformation. This framework achieves a triple shift: from “static lesson preparation” to “dynamic generation”, from “one-way lecturing” to “two-way exploration”, and from “uniform assignments” to “personalized pathways” through dynamically generating teaching resources, constructing interactive classrooms, designing personalized learning paths, creating open collaborative projects, and innovating competency-oriented assessment. Practice shows that this teaching model can effectively stimulate students' learning interest and enhance their programming practical ability and higher-order thinking literacy.

Keywords:

AI-empowered Teaching, Python Teaching, Generative Artificial Intelligence, Teaching Paradigm Transformation

1. Introduction

Python, with its concise syntax, ease of learning, and rich ecosystem, has become the programming language of choice for general education and foundational courses in many university majors, widely applied in teaching across disciplines such as computer science, accounting, economics, and biology [1]. However, traditional Python teaching has several pain points: homogenized teaching materials that are difficult to adapt to the background needs of students from different majors; classroom teaching dominated by one-way lectures, lacking interactivity; monotonous forms of post-class exercises, lacking personalized guidance; assessment methods

focusing on knowledge memorization and code writing, neglecting the cultivation of higher-order thinking skills.

The emergence of generative AI large models provides a new technological path to address the above problems. These tools possess core capabilities such as natural language interaction, code generation and debugging, and multimodal explanation, enabling them to be deeply integrated into the entire teaching process as teaching aids. However, the application of AI in teaching is not simply a technological add-on but involves a systematic transformation of teaching philosophies, methods, and evaluation systems [2]. How to normatively and efficiently integrate AI tools into Python teaching to achieve “human-machine collaboration” rather than “human-machine replacement” has become a crucial issue in current university programming education.

Based on years of Python teaching practice and combined with the characteristics of AI technology, this paper constructs a full-process AI-empowered Python teaching framework aimed at promoting the shift in teaching paradigm from “instructor-centered knowledge transmission” to “learner-centered, AI-assisted” approach, providing support for cultivating innovative and interdisciplinary talents required in the intelligent era.

2. Theoretical Foundation of AI-Empowered Python Teaching

2.1. Constructivist Learning Theory

Constructivism posits that knowledge is not unilaterally transmitted from teacher to student but is actively constructed by students through interaction with the environment in specific contexts. AI tools provide ideal technical support for constructivist learning: by dynamically generating personalized cases and exercises, they create learning situations tailored to students’ professional backgrounds and interests; through real-time code debugging, Socratic questioning, and other interactive forms, they guide students to actively explore knowledge; through project collaboration and problem-solving, they facilitate the internalization and transfer of knowledge in practice.

A research report from the “Connotation, Cultivation, and Innovative Design of Smart Courses” thematic presentation held by Chongqing Jiaotong University pointed out that smart courses adhere to the “student development-centered” philosophy, integrate cutting-edge technologies such as generative AI and big data, and utilize carriers like online platforms, AI teaching assistants, and smart classrooms to conduct deep blended learning designs, achieving data-driven continuous iterative optimization [3]. This aligns highly with the core concepts of constructivist learning theory.

2.2. Human-Machine Collaborative Education Concept

The human-machine collaborative education concept emphasizes that AI, as an intelligent tool, its core value lies in extending and enhancing human cognitive abilities, not replacing humans. In Python teaching, AI assumes the roles of “intelligent teaching assistant” and “collaborative partner”: responsible for repetitive, mechanical tasks, freeing teachers to focus on cultivating higher-order abilities; simultaneously helping students break through cognitive limitations and engage in exploratory learning within a broader knowledge space.

The “Teacher-Student-Machine” synergy concept proposed by Guangzhou City Polytechnic innovatively upgrades artificial intelligence from a “tool carrier” to a “collaborative participant”, reshaping the temporal and spatial boundaries of the classroom and the distribution of authority and responsibility among subjects[4]. This concept provides theoretical guidance and a practical framework for human-machine collaboration in Python teaching.

2.3. Higher-Order Thinking Skills Cultivation Objective

In the context of the intelligent era, the core objective of programming education is no longer merely code-writing skills but higher-order thinking literacies such as critical thinking, innovative thinking, problem-solving ability, and collaborative ability. AI tools can quickly provide code answers, rendering memorization-based, repetitive knowledge assessment meaningless. This necessitates a shift in teaching focus towards cultivating students’ abilities in code comprehension, review, optimization, and innovative application—core competencies that are difficult for AI to replace.

Scholar Hu Xiaoyong et al. pointed out in their research that teachers should become supervisors of technological risks, anticipating risks and conducting behavioral interventions to reduce the harm caused by adverse risks to education [5]. This requires that Python teaching, while utilizing AI tools, must emphasize the cultivation of students’ critical thinking and ethical awareness.

3. Practical Framework for AI-Empowered Python Teaching

The AI-empowered Python teaching framework constructed in this paper covers five stages: pre-class, in-class, post-class, project practice, and assessment evaluation. These stages connect and work synergistically to form a full-process closed-loop teaching system.

3.1. Pre-class Preparation and Course Design: From “Static Preparation” to “Dynamic Generation”

Traditional lesson preparation models rely on the accumulation of individual teacher experience, resulting in slow updates to teaching materials and low levels of personalization. AI tools enable the dynamic generation and optimization of teaching resources, improving preparation efficiency and quality.

3.1.1. Personalized Lesson Plan and Case Generation

By precisely designing prompts and inputting key information such as teaching objectives, student backgrounds, and key/difficult teaching points into the AI large model, teachers can quickly generate personalized lesson plans and teaching cases. For example:

For accounting majors: “Please design a lesson plan on ‘Lists and Loops’ for an introductory Python course for 53 accounting major undergraduates, including one social media data analysis case that incorporates accounting data processing thinking, with clear and operable steps.”

For function teaching: “Generate 5 Python function exercises of increasing difficulty for the function chapter, covering knowledge points such as function definition, parameter passing, return values, and recursive calls, accompanied by

reference answers, detailed analysis, and common student error analysis, suitable for non-computer science beginners.”

AI-generated teaching materials are not used directly but serve as an “inspiration library” for teacher preparation. Teachers need to select, optimize, and integrate based on actual teaching conditions to ensure the scientific nature, interest, and suitability of the cases. This model not only reduces the teacher’s preparation burden but also makes teaching cases more relevant to the cognitive backgrounds of students from different majors, enhancing learning relevance.

3.1.2. “Code Story” Visualization of Abstract Concepts

Abstract concepts in Python are teaching difficulties, and students easily fall into the predicament of “knowing how to use but not understand”. AI can transform dry technical logic into vivid stories, metaphors, or visual charts, lowering the comprehension threshold. For example:

Explaining bubble sort: “Generate a story about ‘elves lining up for a magic ceremony’, integrating each step of the bubble sort into the storyline, using personified language to explain the sorting principle.”

Explaining recursion: “Use examples from daily life to metaphorize the ‘self-calling’ and ‘termination condition’ of recursion, and generate a flowchart showing the execution process of a recursive function.”

Through storified and visualized presentation methods, abstract concepts become concrete and perceptible, catering to the needs of students with different learning styles and improving knowledge transfer efficiency.

3.2. Classroom Teaching and Interaction: From “One-way Lecturing” to “Two-way Exploration”

The core of classroom teaching is to construct multi-directional interaction among teachers, students, and between humans and machines, breaking the traditional “teacher lectures, students listen” one-way transmission model, and cultivating students’ active exploration ability and human-machine collaboration ability.

3.2.1. Real-time Code Demonstration and Debugging

During classroom code demonstration sessions, teachers publicly use AI tools for positive demonstration and reverse debugging, visually demonstrating the efficient mode of human-machine collaboration:

Positive demonstration: “We need to write a crawler to scrape weather data for a certain city over the past 30 days for subsequent data analysis. First, let the AI generate the basic code framework, then we analyze the code logic together—why is the request header setting important here? How to handle anti-crawling mechanisms? How to optimize the data storage format?”

Reverse debugging: “This is a typical erroneous code from last lesson’s student assignment. Let’s ask the AI to analyze the cause of the error. Look, the AI points out an error in the loop condition setting. So how do we verify if this diagnosis is correct? How can we modify it to be more efficient?”

This teaching method clarifies the positioning of AI—“a powerful copilot” rather than an “answer generator”—allowing students to understand that AI is a tool to

improve efficiency, while core logical analysis and optimization decisions still require human leadership.

3.2.2. Cultivating Precise Questioning Ability

During course review or knowledge point exploration sessions, teachers do not provide answers directly but guide students to explore knowledge by asking questions to the AI, focusing on cultivating students' questioning ability and information retrieval skills:

Example teaching scenario:

Student: "What is the use of the 'init' method in Python?"

Teacher: "That's a great question! But asking directly like this, the AI's response might be too general. How can we refine the question to get a more precise answer? For instance, can we add the context 'in the process of class instantiation'? Can we explicitly request 'explain in language understandable by beginners and provide an example'?"

Guided by the teacher, the student optimizes the question to: "Please explain the role of the 'init' method in the class instantiation process in language understandable by Python beginners, and provide one simple example code".

This model not only helps students deepen their understanding of knowledge but also cultivates their core skill for efficient interaction with AI—precise questioning ability, which is an indispensable survival skill in the intelligent era [6].

3.2.3. Adapting to Differentiated Learning Needs

For difficult concepts in Python, a single explanation method can hardly meet the needs of all students. AI can provide multi-angle, multi-modal explanation schemes:

- Metaphorical explanation: "Use 'gift wrapping and decoration' as a metaphor for the function of decorators—adding extra functionality without changing the original function code."
- Flowchart explanation: Generate a flowchart of the decorator execution process, showing the function call sequence and parameter passing path.
- Pseudocode explanation: Use simplified pseudocode to show the core logic, stripping away syntactic details and focusing on the conceptual essence.
- Example demonstration: Generate multiple example codes with progressive difficulty, from basic applications to advanced scenarios, gradually deepening understanding.

Multi-dimensional explanation methods can cater to students with different learning styles, allowing each student to find their own path to understanding.

3.3. Student Learning and Practice: From "Uniform Assignments" to "Personalized Pathways"

Traditional post-class exercises are mostly uniform, making it difficult to accommodate differences in student foundations and professional needs, easily leading to the dilemma where "students with strong foundations find it easy, while those with weak foundations find it difficult". AI tools enable the customization of exercises and provision of instant feedback, constructing adaptive learning paths.

3.3.1. Personalized Exercise Generation

Teachers provide students with prompt templates, guiding them to generate exclusive exercises based on their professional background, interests, and knowledge weaknesses by prompting the AI:

Economics major student: “Please generate one Python exercise requiring writing a function to calculate the GDP growth rate of a region over the past 5 years, input is a list of 5 years of GDP data, output is a list of growth rates, and include test cases.”

Biology major student: “Please design one Python exercise requiring the use of a dictionary to store 3 DNA sequences and their corresponding protein names, implementing add, query, modify, and delete functions, and handle input errors.”

Student with weak foundation: “Please generate 2 Python loop exercises with low difficulty, focusing on traversing lists with for loops and while loop termination conditions, with detailed step-by-step hints.”

Personalized exercises enhance learning relevance and interest, allowing students to feel the close connection between programming and their own majors and interests, thereby stimulating learning motivation.

3.3.2. AI-Assisted Self-Directed Learning and Instant Feedback

Students are encouraged to first attempt collaborative inquiry with the AI when encountering problems, and are provided with standardized prompt templates to guide them in efficiently obtaining help rather than directly asking for answers:

- Code comprehension template: “You are a patient Python tutor. Explain the execution logic of the following code in language understandable by beginners, analyze the function line by line, and avoid piling up professional terms: code snippet”.

- Error troubleshooting template: “I encountered an error while running the following code. Please do not provide the modified code directly, but tell me the error type, possible causes, and give 3 troubleshooting directions: code snippet”.

- Optimization suggestion template: “This is my completed Python code ([code snippet]). Please provide optimization suggestions from three aspects: code style, execution efficiency, and readability, explain the reasons for optimization, and provide optimization ideas rather than directly modifying the code.”

Through this approach, students can receive instant feedback, cultivate self-directed learning and problem-solving abilities, while avoiding over-reliance on AI.

3.3.3. AI-Assisted Assignment Pre-review and Optimization

After completing assignments, students can first submit their code to the AI for preliminary review. The AI provides feedback from the following dimensions:

- Syntax errors: Point out syntax issues in the code and suggest modifications.
- Logical flaws: Hint at potential logical errors.
- Code style: Check compliance with PEP 8 standards.

Efficiency optimization: Suggest more efficient algorithms or data structures.

Students modify and optimize their work based on AI feedback before submitting it to the teacher. This process not only improves assignment quality but also deepens students' understanding of knowledge through revision; simultaneously, it reduces the teacher's repetitive grading burden, allowing them to focus their energy on higher-order ability evaluation and personalized guidance.

3.4. Project and Practice: From “Working Behind Closed Doors” to “Open Collaboration”

Project practice is a core part of Python teaching, cultivating students' comprehensive application ability and collaborative skills. AI tools can provide full-process support for project practice, simulating real software development environments, and enhancing project quality and learning outcomes.

3.4.1. Project Brainstorming and Planning

In the initial stage of group projects, each group collaborates with AI to complete the preliminary project planning:

- Idea generation: “We are accounting major students and want to do a Python data analysis project with a theme related to financial data. Please generate 5 specific project ideas that are feasible and practical, and explain the core functions of each project.”
- Technology selection: “Our project needs to implement Excel data reading, data cleaning, visual analysis, and report generation functions. Please recommend suitable Python libraries and explain the advantages and usage scenarios of each library.”
- Task breakdown: “Our project is a listed company financial indicator analysis system. Please break down the project into specific task modules, clarifying the function, technical requirements, and timeline for each module, suitable for a 4-person group collaboration.”

The planning schemes generated by AI provide students with a reference framework. Students discuss, adjust, and optimize based on this, stimulating innovative thinking and cultivating project planning skills.

3.4.2. AI “Consultant” Support During Project Implementation

During project implementation, students can use the AI as a “project consultant” for consultation to solve technical difficulties:

- Function implementation: “How to use pandas to read multiple sheets in an Excel file and merge them into one DataFrame? Please provide the core code and parameter explanations.”
- API usage: “How to set Chinese display in matplotlib charts? How to adjust axis ticks and legend position? Please provide example code.”
- Error troubleshooting: “When running the code, a ‘ValueError: cannot reindex from a duplicate axis’ error occurred. What is the reason? How to troubleshoot?”

AI's instant support allows students to quickly break through technical bottlenecks, focusing more energy on the project's creative design and functional optimization, while simulating the working mode of using tools to solve problems in real software development.

3.5. Assessment and Evaluation: From “Code Writing” to “Capability Orientation”

Traditional Python assessments mostly consist of code-writing questions, which can easily be answered directly by AI, making it difficult to truly reflect students’ ability levels. The assessment system in the AI era should focus on higher-order thinking skills, achieving a shift from “result-oriented” to “process + capability-oriented”. The AI-empowered Python teaching assessment system is shown in Table 1.

Table 1. AI-Empowered Python Teaching Assessment System.

Assessment Type	Assessment Content	Ability Assessment Focus
Project Report	Requires elaboration on project design ideas, reasons for technology selection, challenges encountered and solutions, functional innovations, reflections and improvement directions.	System design ability, problem-solving ability, innovative thinking, written expression ability.
Code Review	Provides AI-generated code with potential issues, requiring students to identify problems, analyze causes, and optimize.	Code comprehension ability, critical thinking, optimization ability.
Live Debugging	In a computer lab exam, given a program with multiple bugs, requires students to troubleshoot, fix, and explain the problems on the spot.	Debugging ability, problem localization ability, knowledge application ability.
Prompt Engineering Assessment	Presents complex requirements, requires students to design prompts to make the AI generate code that meets the requirements.	Requirement analysis ability, precise expression ability, human-machine collaboration ability.

The new assessment system reduces the proportion of memorization-based, repetitive questions, focusing on assessing students’ higher-order abilities in understanding, design, review, and problem-solving, which better aligns with the talent cultivation needs of the intelligent era.

4. Ethical Norms and Risk Prevention in AI-Empowered Teaching

While the application of AI in teaching brings convenience, it also faces risks such as academic integrity, over-reliance, and ability weakening. Therefore, clear ethical norms and risk prevention mechanisms must be established to ensure the healthy development of AI-empowered teaching [7].

4.1. Establishing Clear AI Use Policies

At the beginning of the course, teachers need to clearly define the boundaries and norms of AI use for students, forming a written “AI Use Guide” including:

- Permitted use scenarios: Pre-class preview, post-class exercises, technical consultation in project practice, code optimization suggestions, concept understanding assistance, etc.
- Restricted use scenarios: Closed-book quizzes, final exams, and other assessment sessions requiring independent completion, where direct use of AI to generate answers is prohibited.
- Citation norms: If AI-generated code or ideas are used in assignments or projects, the AI tool name, usage time, and generated content must be clearly indicated; otherwise, it is considered academic misconduct.

Clear use policies can prevent improper use behaviors caused by students' vague understanding, maintaining the fairness and seriousness of teaching.

4.2. Teaching the Core Skill of “Prompt Engineering”

The ability to interact efficiently with AI is a core skill in the intelligent era. Teachers should incorporate “prompt engineering” into the teaching content, teaching students to design clear, specific, and contextualized prompts, including:

- Clear objectives: Clearly state the task the AI needs to complete, such as “explain concept”, “generate code”, “troubleshoot error”.
- Supplemental context: Provide necessary contextual information, such as “for Python beginners”, “adapted for accounting major scenarios”.
- Setting constraints: Specify output requirements, such as “code needs detailed comments”, “do not give the answer directly, only provide hints”.
- Iterative optimization: Gradually adjust prompts based on the AI's initial output to obtain more accurate results.

4.3. Strengthening Critical Thinking Cultivation

AI-generated content is not absolutely correct and may contain syntax errors, logical flaws, or fail to meet requirements. Teachers need to repeatedly emphasize the principle of “trust but verify”, cultivating students' ability to critically review AI output:

- Code verification: For AI-generated code, require students to analyze the logic line by line and verify its correctness by running test cases.
- Logical questioning: Guide students to think “Is the AI's solution optimal? Are there other implementation methods?”
- Error analysis: Collect error cases generated by AI and conduct collective analysis in class, letting students understand the limitations of AI.

4.4. Preventing Over-reliance Risks

To prevent students from becoming over-reliant on AI, leading to the weakening of basic programming abilities, the following measures should be taken:

- Setting “AI-free” basic exercises: In the learning stage of core knowledge points (e.g., variables, loops, conditional judgments), arrange exercises that must be completed independently, prohibiting AI assistance, to consolidate basic abilities.
- Increasing manual debugging sessions: Incorporate manual error troubleshooting exercises in classroom teaching, allowing students to master basic debugging methods and skills.
- Strengthening conceptual understanding: Use classroom Q&A, group discussions, etc., to assess students' understanding of core concepts, not just focusing on code writing results.

5. Teacher Role Transformation: From “Knowledge Transmitter” to “Guide in the Intelligent Era”

The integration of AI does not diminish the role of teachers but places higher demands on it. Teachers need to transform from traditional “sole knowledge transmitters” into the following four roles:

5.1. Designer of Learning Experiences

Teachers are no longer mere transporters of teaching materials but designers and organizers of learning activities. They need to combine the characteristics of AI tools to design engaging learning scenarios, project tasks, and interactive sessions, stimulating students’ learning interest and exploration desire, and constructing immersive, personalized learning experiences.

5.2. Guide for Human-Machine Collaboration

Teachers need to guide students to correctly understand the value and limitations of AI, master methods for efficient collaboration with AI, and help students establish a “human-machine synergy” mindset. When students encounter difficulties using AI, provide targeted guidance to steer them from “passively accepting answers” to “actively using tools to explore answers”.

5.3. Stimulator of Higher-Order Thinking

Teachers’ core energy should focus on cultivating students’ higher-order thinking skills. Through questioning guidance, case analysis, project reviews, etc., stimulate students’ critical thinking, innovative thinking, and problem-solving abilities. For example, in project reviews, focus not only on functional implementation but also on the innovativeness of the design ideas and the rationality of the solutions.

5.4. Guardian of Academic Norms

Teachers need to strengthen academic integrity education, clarify the ethical boundaries of AI use, and establish sound assessment and supervision mechanisms to prevent academic misconduct. Simultaneously, pay attention to students’ learning processes, and through classroom interaction, assignment feedback, project communication, etc., promptly identify students’ learning difficulties and improper use behaviors, providing targeted guidance and correction.

6. Teaching Practice Effects and Reflection

The aforementioned teaching framework was applied in the “Python Programming” course in the second semester of the 2024-2025 academic year (targeting 117 non-computer science undergraduate students from majors including accounting, economics, and biology). Evaluation was conducted through questionnaires, grade analysis, and student interviews, with the following results:

6.1 Teaching Effects

- Significant increase in learning interest: Questionnaire surveys showed that 85% of students believed that “personalized cases and exercises” enhanced their learning interest, and 78% of students indicated that “AI-assisted learning made programming simpler and more fun.”

- Noticeable enhancement of practical ability: In the project practice at the end of the course, the quality of students' projects improved significantly compared to before. 90% of projects achieved the expected functions, and 65% of projects had innovative extended functions, representing increases of 25% and 30% respectively compared to the previous academic year.

- Improvement in higher-order thinking skills: Assessment results from code review and live debugging sessions showed that 72% of students could accurately identify potential issues in AI-generated code and propose optimization solutions, an increase of 40% compared to the traditional teaching mode.

- Formation of human-machine collaboration ability: In interviews, 80% of students stated that they had mastered methods for efficient collaboration with AI, could use AI to solve technical problems in learning and practice, while maintaining independent thinking ability.

6.2. Existing Problems and Improvement Directions

Although the teaching practice achieved certain results, some problems still require continuous improvement:

- Uneven quality of AI output: Some AI-generated lesson plans or codes contained errors, requiring teachers to spend time screening and optimizing. In the future, output quality can be improved by optimizing prompt design and using more professional programming-specific AI tools.

- Uneven collaboration effects due to student ability differences: Students with weak foundations might have difficulty obtaining effective help when using AI due to inability to accurately express needs. In the future, layered prompt templates need to be designed to provide targeted guidance for students with different foundations.

- Increased difficulty in academic integrity supervision: Although AI use norms were established, there is still a risk of students concealing AI use. In the future, comprehensive judgment can be made by combining code similarity detection tools and process evaluation data (e.g., learning logs, debugging records).

7. Conclusions

The development of AI technology brings revolutionary opportunities to Python teaching and also presents new challenges. The AI-empowered Python teaching framework proposed in this paper achieves a systematic transformation of the teaching paradigm through dynamically generating teaching resources, constructing interactive classrooms, designing personalized learning paths, creating open collaborative projects, and innovating competency-oriented assessment. The core value of this framework lies in positioning AI as an "intelligent auxiliary tool" rather than a "replacer", improving teaching efficiency and quality through human-machine collaboration, while focusing on the cultivation of higher-order thinking skills, providing support for talent cultivation in the intelligent era.

In the process of AI-empowered teaching, clear ethical norms, effective risk prevention, and the active transformation of teacher roles are key to ensuring teaching quality [8]. In the future, with the continuous development of AI technology, Python teaching will face further changes, requiring teachers to continuously explore, innovate courageously, and constantly optimize teaching models to cultivate more innovative and interdisciplinary talents that meet the needs of the intelligent era.

Conflicts of Interest

The author declares that there is no conflict of interest regarding the publication of this article.

Funding

This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

References

- [1] Xinhua News. AI Subject Simulation Training Agent Released, Enabling Each Student to Have a Dedicated AI Training Tutor. Department of Education of Inner Mongolia Autonomous Region. Available online: https://jyt.nmg.gov.cn/jydt/mtjj1/202511/t20251103_2811766.html (accessed on 1 November 2025).
- [2] Chen, X.Q.; Min, L.B. Generative Artificial Intelligence Empowers Kindergarten Teacher Professional Development: Value, Risks, and Countermeasures. *Teacher Education Research*, 2025, 37(04), 33-39.
- [3] Su, Y.K. Connotation, Cultivation and Innovative Design of Smart Courses Thematic Report. Chongqing Jiaotong University “AI Empowers Teacher Development” Series Activities, Chongqing, China, 2025-10-20.
- [4] Wang, J.; Xiao, Y.J.; Wang, Y.C. et al. “Teacher-Student-Machine” Synergy, “Teaching-Learning-Assessment” Integration: Exploration and Practice of the Smart Teaching Ecosystem Construction at Guangzhou City Polytechnic. Guangzhou: Guangzhou City Polytechnic, 2023.
- [5] Hu, X.Y.; Huang, J.; Lin, Z.R. et al. Educational AI Ethics: Connotational Framework, Cognitive Status Quo, and Risk Aversion. *Modern Distance Education Research*, 2022, 34(02), 21-28+36.
- [6] Wang, Y.M.; Wang, D.; Liang, Y.W. The Value Implication and Cultivation Path of Learners’ Questioning Ability in the Context of “AI+Education”. *Journal of Distance Education*, 2024, 42(1), 66-75.
- [7] Zheng, X.D.; Zhou, Z.B.; Fan, Y.J. Ethical Governance of Educational Artificial Intelligence: Realistic Dilemmas and Breakthrough Paths. *Modern Distance Education Research*, 2024, 36(4), 12-21.
- [8] Zhu, Z.T.; Dai, L. Research on the Construction of an Ethical Norm Framework for the Application of Artificial Intelligence in Education. *E-Education Research*, 2023, 44(5), 5-12.



© 2025 by the author(s); licensee International Technology and Science Publications (ITS), this work for open access publication is under the Creative Commons Attribution International License (CC BY 4.0). (<http://creativecommons.org/licenses/by/4.0/>)