

Practical Exploration of AIGC-Enabled Teaching Reform in Python Programming Courses

Liangtao Yang

Shanghai Polytechnic University, Shanghai 201209, China

Abstract: With the rapid development of artificial intelligence technology, generative AI represented by ChatGPT has demonstrated tremendous potential in the field of education. Based on a comprehensive analysis of the challenges posed by AIGC to Python programming courses, this paper delves into the new situations and issues faced by Python programming courses, and proposes to reform traditional teaching modes by utilizing generative AI technology. At the specific implementation level, the reform content covers reconstructing teaching modes, enriching teaching resources, realizing personalized learning, and establishing a multi-dimensional evaluation mechanism. Teaching practice has shown that integrating AIGC into the teaching reform of Python programming courses has not only achieved good teaching results but also received positive feedback from students, providing a reference for innovation in higher education teaching modes.

Keywords: AIGC, Large Language Model (LLM), Python Programming, Teaching Reform.

1. Introduction

In recent years, with the rapid rise of generative artificial intelligence (AI) technologies represented by ChatGPT, the ability of AI to empower education and teaching has become increasingly prominent, driving comprehensive and deep-seated reforms in the field of higher education[1]. The deep integration of intelligent technology and education and teaching has become one of the core trends in the reform and development of higher education in the new era. Against this backdrop, national and educational departments at all levels have attached great importance to the construction of smart education and the digital transformation of education, successively issuing a number of guiding policy documents to provide solid policy support and development directions for AI-enabled classroom teaching reform. In 2022, the State Council issued the "14th Five-Year Plan for the Development of the Digital Economy," explicitly proposing to further promote the construction of smart education and empower high-quality education with digital technology[2]. In the same year, the National Education Work Conference officially deployed the implementation of the National Education Digitalization Strategy Action Plan, comprehensively promoting the digitalization and intelligent teaching reform of various courses. In 2025, the Ministry of Education released the "White Paper on Smart Education in China," further clarifying the comprehensive integration of AI technology into all elements and processes of education and teaching, and promoting the systematic reshaping of teaching modes, teaching resources, and evaluation systems. Python programming, which combines theoretical and practical aspects, is a key course for cultivating students' programming thinking, engineering practice ability, and innovative literacy. Therefore, exploring the deep integration path of generative AI and higher education teaching, and fully introducing intelligent technology into Python programming course teaching, is not only an inevitable requirement for implementing the construction of smart education but also an important path for cultivating composite innovative talents that meet the development needs of industries in the new era.

2. The Challenges Brought by AIGC to Python Programming Teaching

2.1. Students are Prone to Excessive Dependence, Leading to a Decline in Their Autonomous Learning Ability

With the rapid iteration and popularization of large language model technology, the threshold for current AIGC programming tools continues to decrease. Practitioners in the industry without programming background can quickly build simple applications with the help of models. In the teaching scenario of Python programming in universities, relying on large models to complete code writing, homework answering, and course report writing has become a common learning method for students[3].

Python, as an introductory programming language, features course exercises and homework assignments that primarily focus on basic syntax application and simple logic implementation. Most programming problems can quickly obtain standard answers through AIGC tools. This has led to a lazy mindset among most students, who directly rely on large models to generate code regardless of the difficulty of the problem, completely skipping the learning process of independent thinking, logical reasoning, manual coding, and debugging[4]. This overly dependent learning mode has far-reaching negative impacts: on the one hand, students lack practical training in handwriting code and independent debugging for a long time, resulting in superficial understanding of Python syntax rules, program logic, and code operation principles. Their ability to write code and debug in practice is difficult to steadily improve, making it impossible to form a solid programming foundation; on the other hand, students' independent thinking ability, logical reasoning ability, and innovative thinking continue to weaken, gradually losing their core competencies of independently exploring and solving problems.

2.2. The Difficulty of Learning Effect Evaluation has Increased, Making it Difficult to Guarantee the Authenticity of Assessments

The traditional evaluation system for Python programming teaching primarily relies on forms such as homework, classroom exercises, and unit tests, assessing learning outcomes through student code completion, correct operation, and format standardization. However, the widespread adoption of AIGC technology has completely disrupted the traditional assessment and evaluation logic, rendering the original assessment standards and methods gradually ineffective and posing significant challenges to teachers' teaching evaluation work. Currently, mainstream large models such as Doubao, KIMI, and the GPT series can accurately match the basic Python teaching content in universities, quickly generating homework code with standardized format, complete logic, and direct operability. This code can easily pass all test cases in the course and is highly likely to achieve full marks[5].

Python code possesses a standardized syntax system, fixed operational logic, and unified formatting conventions. Code generated by AI is neat and rigorous, without obvious personal writing characteristics or distinct "AI traces". Even if teachers add manual review and line-by-line verification as assessment steps, it is difficult to effectively distinguish whether the code was written by students independently or generated by AI. It is impossible to judge students' true mastery level through homework results. Ultimately, this leads to the false phenomenon of "high scores but low abilities" in teaching assessment, making it impossible for teachers to accurately grasp students' knowledge weaknesses and difficult to adjust the teaching pace and content accordingly[6]. This seriously affects the authenticity, objectivity, and effectiveness of teaching evaluation.

2.3. There are Defects in the Code Quality of Large Models, Which Mislead Students' Cognition and Learning Logic

Currently, AIGC programming tools exhibit high adaptability to simple and standardized Python basic programming tasks, efficiently completing introductory-level code writing work such as variable definition, loop conditions, and basic function calls. However, when faced with programming tasks that involve complex logic, special scenarios, and require multi-dimensional logical deduction, the limitations of the model become fully apparent. The generated code commonly suffers from issues such as logical loopholes, parameter errors, insufficient compatibility, and algorithm implementation failures, making it unable to operate stably and normally.

For professional developers, they can accurately identify vulnerabilities and defects in AI code, and correct and optimize it with their professional knowledge. However, for students who have no foundation in Python and are in the introductory stage, their own knowledge reserves are weak and their programming experience is insufficient, making them completely incapable of discerning errors in AI code and identifying logical vulnerabilities. Students generally have misconceptions, believing that code that runs is correct code, and they regard code generated by AIGC as the standard answer, blindly copying and applying it. From the perspective of learning patterns, Python programming learning needs to

start with basic cases, accumulating experience through independent coding, manual debugging, and deep exploration of principles, gradually achieving knowledge transfer and mastering complex algorithms and project development capabilities. However, students who rely too much on AI code with hidden vulnerabilities, skipping the basic accumulation and thinking training process, will not only form incorrect programming cognition and solidify erroneous coding habits, but also lose the ability to transfer knowledge and tackle problems independently, laying serious hidden dangers for subsequent advanced programming learning and algorithm learning.

2.4. Traditional Teaching Models are Imbalanced, and the Value of Classroom Education is Weakened

Python programming teaching is not only about imparting syntactic knowledge and programming skills, but more importantly, its core educational goal is to cultivate students' computational thinking, logical thinking, rigorous programming literacy, and the scientific research spirit of tackling difficulties. Traditional classrooms achieve the dual goals of knowledge impartation and ability cultivation through a closed-loop model of "teacher explanation - student practice - classroom Q&A - review of incorrect questions". However, the comprehensive penetration of AIGC tools has completely disrupted the balance of traditional classroom teaching, significantly weakening the dominant role of teachers and the value of practical training in the classroom. Traditional teaching models are difficult to adapt to new-era teaching scenarios.

Throughout the entire process of pre-class preview, in-class practical operation, and post-class consolidation, students can quickly complete learning tasks with the help of AI tools, without having to follow the teacher's thinking and gradually code. In the classroom, students may appear to be able to quickly complete practical tasks and keep up with the teaching progress, but in reality, they are not participating in thinking training and practical exercises, and have only a rudimentary understanding of the key and difficult points explained by the teacher and the programming logic. Meanwhile, in traditional programming teaching, the process of students repeatedly reporting errors and debugging is a crucial step in understanding program logic and accumulating programming experience, as well as an important process in cultivating students' patience, rigor, and problem-solving abilities. However, AIGC directly avoids all trial and error and debugging processes, resulting in students lacking the core experience of programming learning, unable to appreciate the logical charm of programming learning, and continuously reducing their learning initiative and exploration desire.

3. Teaching Reform Content of Python Programming Course

To effectively implement the reform objectives of the Python programming course, and address the pain points in traditional course teaching, such as monotonous teaching formats, outdated and rigid resources, poor adaptability to learning situations, and one-sided evaluation systems, this course leverages generative artificial intelligence technology to reshape the course teaching system from four core dimensions: teaching mode, teaching resources, learning

methods, and evaluation mechanisms. This aims to comprehensively enhance the quality of course teaching and students' programming practice and innovative application abilities[7].

3.1. Reconstruction of Teaching Mode from One-Way Lecture to Two-Way Interaction

Traditional Python programming courses often adopt a "spoon-feeding" one-way teaching model, centered around theoretical explanations and code demonstrations by teachers. Students passively receive knowledge, resulting in generally low classroom participation and learning initiative. At the same time, programming knowledge is highly abstract and code logic is obscure and difficult to understand. Traditional static teaching methods struggle to demonstrate the underlying principles and dynamic processes of program execution, leading to students' incomplete understanding of core knowledge points and weak practical coding abilities. Overall, the teaching effect is unsatisfactory.

Leveraging generative artificial intelligence technology, this course thoroughly disrupts the traditional one-way teaching model, reconstructing a new classroom paradigm featuring bidirectional interaction between teachers and students, and human-machine collaboration and empowerment. This approach fully stimulates students' enthusiasm and initiative for autonomous learning. During classroom instruction, teachers can utilize generative artificial intelligence to swiftly create interactive dynamic animations of Python programs, visualizing core code logic such as abstract variable operations, loop nesting, function calls, and data structures. This approach comprehensively reconstructs the entire process of program execution, underlying logic, and intermediate operational results, intuitively resolving difficulties in learning programming knowledge and assisting students in accurately grasping knowledge points.

Meanwhile, teachers can leverage large AI models to generate and distribute targeted classroom exercises in real-time, based on the key and difficult points taught in class. After students complete the online exercises, the AI system can provide instant feedback, correct errors automatically, and accurately calculate the correct answer rate, common mistake points, and weak knowledge areas for the entire class, forming a visual learning analysis report. Relying on big data analysis results, teachers can grasp the overall classroom learning effect in real-time, dynamically adjust the teaching pace, supplement explanations of key and difficult points, and conduct targeted question-answering and review, achieving teaching tailored to learning.

3.2. Dynamic Enrichment and Updating of Teaching Resources Enabled by Intelligence

Traditional Python course teaching resources face issues such as being rigid and outdated, lacking diversity, and insufficient relevance. The update cycle of textbooks and courseware is long, the cases are outdated, and they are disconnected from the actual application scenarios of various majors, making it difficult to adapt to the new era of programming teaching needs and the learning demands of students from different majors. Generative artificial intelligence technology provides a new path for the dynamic construction, precise updating, and personalized adaptation of teaching resources, enabling the intelligent, dynamic, and professional development of teaching resources.

On the one hand, generative large models can automatically generate basic teaching resources such as courseware, knowledge point handouts, and practical guidance manuals based on the curriculum syllabus, talent cultivation objectives, and classroom teaching progress. These resources are structured in a standardized way, highlight key points, and are suitable for classroom teaching, significantly improving the efficiency of teaching resource production. At the same time, they can update content in real-time according to technological development, avoiding the problem of outdated traditional resources. On the other hand, given the highly practical nature of Python courses, artificial intelligence can be combined with industry scenarios from different majors such as economics and management, engineering, liberal arts, and art to customize specialized application cases. For example, differentiated cases such as data analysis, web scraping, visual graphics, intelligent computing, and automated office work can be generated, making programming teaching meet professional application needs and solving the problem of the disconnect between theory and practice.

In addition, generative artificial intelligence can achieve precise and intelligent push of learning resources based on multi-dimensional user personas such as students' learning history, knowledge mastery level, practical operation level, and learning weaknesses. By combining high-quality learning platform resources across the entire network, it can accurately recommend Python online tutorials, CSDN technical blogs, GitHub open-source projects, practical training question banks, and other expansion resources that are suitable for students' learning levels, building a three-dimensional resource system of "basic resources + professional resources + expansion resources" to meet the layered learning needs of different students.

3.3. Personalized Learning Upgrade from Standardized Indoctrination to Customized Cultivation

Traditional classroom teaching is constrained by teaching resources, faculty strength, and class duration, and generally adopts a standardized teaching model with unified syllabus, content, and progress. This undifferentiated and one-size-fits-all approach may achieve widespread large-scale teaching, but it completely ignores individual differences among students. There is a significant gap in computer basics, logical thinking ability, and programming talent among different students. Students with no foundation find it difficult to keep up with the class progress, while those with programming background encounter the problem of "not learning enough or deeply", failing to achieve individualized teaching and restricting the development of students' personalized abilities.

Based on the intelligent analysis and customized empowerment capabilities of generative artificial intelligence, the course completely breaks the limitations of standardized teaching and builds a personalized learning system that is hierarchical, classified, and precisely adapted. Through pre-course assessments and analysis of historical learning data, artificial intelligence can accurately assess each student's programming foundation, learning ability, and knowledge gaps, and tailor a unique learning plan and tiered learning objectives for students.

3.4. Construction of a Multi-dimensional Evaluation Mechanism Shifting from Result-oriented to Process-oriented

The traditional Python course evaluation model centers on the final written exam and the outcomes of the final project, representing a singular, outcome-oriented summative evaluation. This evaluation approach solely focuses on students' final learning outcomes, overlooking the entire process of students' pre-class preparation, classroom interaction, daily practical operations, problem correction, and ability growth. It cannot objectively reflect students' true learning status and comprehensive abilities, making it difficult to accurately identify problems in students' learning process. Furthermore, it fails to effectively motivate students to continuously improve, and the scientificity, comprehensiveness, and guidance of the evaluation are severely lacking.

Relying on the full-process data collection and intelligent analysis capabilities of generative artificial intelligence, the curriculum constructs a multi-dimensional and three-dimensional evaluation mechanism combining "process evaluation + summative evaluation", realizing the transformation of evaluation from "emphasizing results and ignoring process" to "emphasizing the whole process, all aspects, and abilities". Artificial intelligence can collect students' learning behavior data throughout the entire process, covering classroom interactive answering, quality of homework completion, accuracy of code practice, review of incorrect questions, duration of resource learning, frequency of question and communication, and other full-process data, achieving dynamic tracking and precise recording of students' learning process.

4. Exploration of the Application of AIGC in the Course of "Python Programming"

In response to a series of practical problems brought by AIGC to Python programming teaching, such as student dependence, difficulty in evaluation, unreliable code quality, and imbalanced teaching system, the course group where the author is located, based on the pain points of undergraduate programming course teaching and the development trend of smart education, successfully applied for and was approved for the university level undergraduate smart course construction reform project using the course "Python Programming Design" as a carrier. This project relies on the Chaoxing Smart Teaching Platform and deeply integrates generative artificial intelligence technology. It focuses on the reconstruction of course teaching resources, visualization of knowledge systems, construction of intelligent teaching carriers, multi scenario AI teaching applications, and optimization of teaching data loops to carry out systematic teaching reform practices. It gradually builds a human-machine collaborative teaching system that is suitable for undergraduate programming teaching in the new era. The core construction contents of the course exclusive knowledge base, structured knowledge graph, and full scenario intelligent teaching tool matrix have been preliminarily completed, effectively solving the shortcomings of traditional Python teaching and empowering the improvement of course teaching quality and efficiency.

4.1. Construction of Precise Teaching Resources based on Exclusive Knowledge Base

In response to the common problems of knowledge illusion, knowledge point deviation, and insufficient adaptation of professional content in general large-scale models, the project team abandons the boundaryless generation mode of general AI and builds a localized exclusive knowledge base for courses based on the Python programming course syllabus, talent training objectives, and undergraduate students' cognitive laws, ensuring the accuracy, professionalism, and adaptability of AI teaching output content from the source. The project team systematically sorted out all the knowledge points, key and difficult points, and easy to make mistakes in Python courses, integrated course standards, self written textbooks, teaching cases, practical exercises, final exams, experimental guidance and other teaching resources of all categories, and built a Python course exclusive knowledge resource library with clear hierarchy, complete content, and in line with the actual teaching situation of the school, completely avoiding problems such as content generated by general large models being disconnected from the course system, outdated knowledge points, and incorrect answers. In addition, the platform has achieved precise correlation and matching between knowledge base resources and course knowledge graphs. Each knowledge point is matched with corresponding three-dimensional resources such as theoretical texts, teaching micro courses, in class exercises, textbook slices, and practical cases, forming an integrated resource system of "knowledge points resources exercises", providing solid resource support for hierarchical teaching, self-directed learning, and intelligent Q&A.

4.2. Construction of Layered and Progressive Curriculum Knowledge Graph

To solve the problems of fragmented knowledge points, fuzzy knowledge logic, and difficulty for students to build a systematic knowledge system in traditional Python teaching, the project team used the self written textbook "Python Language Programming" as the core blueprint and relied on AI visualization technology to construct a fully covered, layered, and structured Python course knowledge graph, realizing the visualization, structured, and networked presentation of the course knowledge system, and helping teachers and students clearly grasp the internal logic and progressive relationship of knowledge points.

The construction of knowledge graph is divided into two stages: automatic generation and manual optimization. Firstly, relying on the electronic textbook directory structure and course syllabus, the system automatically generates an initial knowledge graph with "Python Programming Course" as the root node, quickly building the overall knowledge framework. On this basis, the project team combined years of frontline teaching experience to manually optimize and adjust the hierarchical structure and logical relationship of each module and knowledge point, focusing on sorting out the progressive relationship of core teaching modules such as "Python language overview, syntax foundation, sequential branch loop structure, combined data types, functions and modules, file operations, and exception handling", clarifying the prerequisite relationship, subsequent expansion relationship, and parallel correlation relationship of all knowledge points. Each knowledge point node in the graph is equipped with

standardized names, core definitions, and explanations of teaching difficulties. Through exclusive relationship annotation functions, the logical relationships between knowledge points are clearly defined. For example, it is clear that "function definition and parameter settings" are the prerequisite for "function calling and recursive functions", and "basic syntax of lists and dictionaries" is the core prerequisite for "data preprocessing case practice".

4.3. Application of Multi scenario Artificial Intelligence Tools in Stereoscopic Teaching Practice

Based on the constructed knowledge base and knowledge graph, the project team has built a multi scenario AI intelligent teaching tool matrix, covering the entire teaching scenario of student self-learning, teacher lesson preparation and teaching, personalized teaching implementation, etc., and constructing a new teaching mode of "human-machine collaboration and personalized teaching", fully empowering the complete teaching loop of pre class lesson preparation, in class teaching, and post class consolidation. The specific application scenarios are divided into three modules.

Intelligent learning companions achieve 24/7 personalized Q&A guidance. Intelligent learning companion tools break the limitations of traditional teaching time and space, providing students with personalized learning guidance services 24/7, anytime and anywhere, effectively addressing the shortcomings of limited classroom teaching Q&A time and insufficient teacher-student interaction. Teachers can customize and build a dedicated instruction library based on the teaching characteristics and learning needs of Python courses, with preset multi-functional instruction groups such as "grammar knowledge Q&A, code error debugging, typical case analysis, in class exercise generation, and summary of key and difficult points". Students can directly call the corresponding functions to carry out independent learning.

Intelligent lesson preparation assistant helps teachers efficiently generate teaching resources. In response to the problems of time-consuming and laborious traditional lesson preparation, lagging resource updates, and insufficient case innovation, the platform is equipped with an intelligent lesson preparation assistant, which supports teachers to describe teaching needs through natural language, quickly and automatically generate standardized courseware, classroom practical cases, in class exercises, unit test questions, experimental task books and other teaching resources, greatly reducing teachers' repetitive lesson preparation workload and improving lesson preparation efficiency and resource quality.

Intelligent learning situation analysis, dynamic planning, personalized learning path. The system relies on big data technology to collect data on the entire process of student course learning, including knowledge graph learning progress, digital human video viewing time and completion rate, correct answer rate of exercises, distribution of wrong questions, interaction records with intelligent learning partners, and completion status of practical tasks. Through data modeling, an accurate student learning profile is constructed to accurately assess each student's knowledge mastery level, learning shortcomings, and ability levels. Based on the learning profile, the system can automatically plan dynamic personalized learning paths for students, achieving hierarchical teaching and precise education.

4.4. Reform Implementation Effectiveness and Teaching Reflection

The smart course teaching reform of "Python Programming" has achieved a comprehensive upgrade of the course teaching system and completed three core teaching changes: firstly, the upgrading of the form of course resources, breaking the limitations of traditional static textbooks and fixed courseware, and constructing a dynamic, interrelated, and progressive visual knowledge network; The second is the reform of teaching mode, transforming the traditional "one-way teaching by teachers and passive acceptance by students" indoctrination teaching into a new intelligent teaching mode of "top-level design by teachers, human-machine collaboration empowerment, and student autonomy"; The third is to upgrade the educational goals, from simply mastering Python syntax knowledge and completing basic programming tasks, to cultivating students' computational thinking, innovation ability, and engineering practice ability to collaboratively solve complex programming problems using artificial intelligence tools. Most students expressed that with the assistance of generative artificial intelligence tools, the difficulty of course learning has been effectively reduced, the channels for self-directed learning have become more diverse, and the motivation, confidence, and problem-solving ability of programming learning have been significantly improved. The teaching reform practice has achieved good teaching and educational results.

At the same time, the practice of educational reform has also exposed new educational challenges under the new teaching mode, which need to be continuously optimized and improved in subsequent teaching. Firstly, it is necessary to remain vigilant and avoid students' excessive dependence on AI tools, guide students to use intelligent tools reasonably, adhere to independent thinking and coding; Secondly, it is necessary to cultivate students' AI content identification and code correction abilities on a regular basis, helping them accurately identify vulnerabilities and deficiencies in AI generated content and form rigorous programming thinking; Thirdly, it is necessary to continuously iterate and update the course knowledge base, knowledge graph, and Q&A resources, keep up with the trend of Python technology ecosystem updates, and ensure the forefront and practicality of teaching content.

5. Conclusion

The rapid development of generative large language models has provided a new technological support for the innovation of Python programming course teaching, effectively empowering the entire teaching process and significantly enhancing the overall teaching effectiveness of the course. However, it also brings new challenges and tests to college programming course teaching. At present, the integration of generative artificial intelligence and course teaching is still in the initial exploratory stage, and there are still many urgent issues to be addressed in practical teaching applications, including the risk of teacher-student technological dependence, potential security risks of teaching data privacy, and uneven abilities of teachers in applying intelligent teaching methods. Looking ahead, Python programming course teaching needs to continue to deeply explore the path of deep integration between artificial intelligence and teaching. While consolidating students' basic programming knowledge and core skills, it should focus on

cultivating students' AI collaborative application abilities and independent thinking abilities. It should identify the balance point between basic teaching and intelligent empowerment, prevent students from becoming overly dependent on artificial intelligence and reducing to mere "code porters", and strive to cultivate compound technical talents with solid computational thinking, independent development ability, and innovative literacy. At the same time, it should continue to deepen teaching reforms and truly empower high-quality education with intelligent technology.

References

- [1] Zhang, T. (2026). Ministry of Education introduces the progress of promoting artificial intelligence education in China. *Science Education and Culture Review*, (2), 1.
- [2] Qiu, F., & Wu, Y. D. (2024). Core elements analysis of educational innovation driven by generative artificial intelligence. *Education Development Research*, 44(Z1), 9-16.
- [3] Li, H., & Li, K. (2026). The connotation of artificial intelligence literacy for college students in the era of generative AI: Research on training paths. *Information and Computer*, 38(2), 248-250.
- [4] Xu, H., Ju, X. L., & Wang, H. C. (2023). Challenges and countermeasures of programming teaching under large models. *Computer Education*, 21(11), 60-64.
- [5] Ju, X. L., Zhang, Y. M., Wang, H. C., et al. (2024). Exploration of Python programming course teaching assisted by large language models. *Computer Education*, 22(9), 33-37.
- [6] Wu, J. (2026). Teaching quality assurance of secondary colleges in universities under the "Double First-Class" initiative: System improvement countermeasures-taking the School of Artificial Intelligence at Southwest University as an example. *Science Jifeng*, (3), 159-161.
- [7] Xie, H. B., Liu, F., & Qin, H. X. (2024). Exploring the application of large language models in the reform of foundational programming education. *Computer Knowledge and Technology*, 20(8), 39-42.