Frontiers in Arts, Humanities
& Social Sciences

Research on the Application of Python Project-Based Textbooks in Artificial Intelligence Education

Xiedong Song

School of Computer Science and Engineering, Jining University, Jining,273155,China.

Abstract The rapid evolution of artificial intelligence technology has made Python programming competency a critical demand in talent cultivation. However, the "knowledge-oriented" compilation logic and linear grammatical arrangement of traditional textbooks fail to bridge the gap between theoretical instruction and practical application. Targeting this structural contradiction, this study systematically explores the design philosophy, teaching practices, and application effects of Python project-based textbooks in artificial intelligence education. Grounded in constructivist learning theory, project-based learning theory, and the "learning by doing" educational philosophy, the study proposes a textbook design framework characterised by "project orientation, task driving, and unity of knowledge and action," which organically integrates the Python knowledge system and AI application scenarios into graded project tasks. The study comprehensively employs methods such as questionnaire surveys, final grade comparative analysis, and project work evaluation to assess teaching effectiveness. The results indicate that students' learning interest and programming self-efficacy have significantly improved, and most students are able to independently use AI algorithm libraries such as Pandas and Matplotlib to complete data processing and visualisation tasks. The study concludes that project-based textbooks, through the mechanism of "learning programming by doing projects and using AI in programming," effectively enhance students' programming practical abilities and AI application literacy, providing a referable practical pathway and theoretical reference for textbook reform in programming education in the AI era.

Keywords Python programming; project - based textbooks; artificial intelligence education; project-driven teaching; teaching reform

Received: May 11, 2026
Revised: June 23, 2026
Accepted: June 25, 2026
Published: July 21, 2026

DOI:10.67148/fahss-2026-229

1. Introduction

1.1 Research Background

The rapid advancement of artificial intelligence technology is profoundly reshaping the global industrial landscape and the ecosystem of higher education. From breakthroughs in deep learning algorithms to the emergence of large language models,

AI technologies have transitioned from laboratory settings to large-scale application scenarios, generating continuously rising social demand for interdisciplinary talents equipped with both programming skills and AI literacy [1]. In this technological wave, Python, with its notable advantages of concise syntax, a rich ecosystem, and an active community, has firmly established its core position as the preferred programming language in the field of artificial intelligence [2]. Whether in machine learning, data science, or deep learning frameworks, Python has become an almost indispensable foundational tool. In other words, to cultivate high-quality talents capable of adapting to the AI era, the Python programming course has inevitably become a critical component that computer science and related majors in higher education institutions cannot bypass.

However, there often exists a considerable gap between ideal and reality. The current teaching practices of Python programming courses in universities generally face several vexing issues. First, the updating of teaching content lags seriously behind. Many classrooms still mechanically teach grammar and concepts by rote, while the industry has already advanced several steps ahead in the field of deep learning—this disconnect leads to graduates finding that what they have learned does not align with actual job requirements [3]. Second, the practical teaching component is quite weak. Programming, after all, cannot be learned solely by listening to lectures; one must genuinely write code, debug, and run projects. Nevertheless, many courses still emphasize theoretical instruction, offering students pitifully few hands-on opportunities, to the extent that they cannot even produce a complete project by the time of their graduation thesis [4]. Third, teaching methods are relatively outdated, lacking both interest and challenge. The traditional trilogy of "grammar lectures—exercise drills—final examinations" struggles to ignite students' enthusiasm for learning, let alone cultivate their ability to solve complex engineering problems [5]. The accumulation of these issues results in an awkward situation: enterprises cannot find suitable candidates, and students cannot secure satisfactory employment.

Where does the root of the problem lie? To put it bluntly, the traditional textbook compilation logic and teaching model follow an outdated "knowledge-oriented" path—systematically teaching all grammatical features of the language first, and then expecting students to figure out how to apply them on their own. This approach might have worked in the early days of the information age, but in today's rapidly evolving AI landscape, it has clearly fallen behind the times [2][4]. What students need is an experience of "learning by doing" in real-world projects, rather than struggling in the vast ocean of grammatical details.

It is precisely based on this practical dilemma that an increasing number of educators are turning their attention to project-based learning (PBL). Project - based learning emphasizes using authentic projects as carriers, allowing students to actively construct knowledge and develop abilities in the process of solving practical problems. Introducing this concept into the compilation of Python textbooks and course design is expected to fundamentally bridge the gap between theoretical instruction and practical application [6]. In recent years, domestic scholars have adopted project - driven approaches to integrate typical machine learning cases into Python language teaching in a layered and gradual manner [4]; other researchers have proposed a "four-stage, dual-chain, three-loop" teaching model empowered by generative AI, exploring the potential

for deep integration of AI technology with Python courses [3]. These explorations provide valuable references for the compilation of Python project-based textbooks.

Nevertheless, frankly speaking, current research mostly remains at the level of pedagogical model discussions, lacking in-depth theoretical construction and empirical verification regarding how to systematically design a Python project-based textbook truly suitable for artificial intelligence education contexts [7]. This precisely constitutes the starting point of the present study.

1.2 Research Purpose and Significance

This study aims to systematically explore the design philosophy, compilation framework, and teaching application effects of Python project-based textbooks in artificial intelligence education, attempting to address three core questions: First, what design philosophy and compilation principles should Python project-based textbooks for AI education follow? Second, how can the Python knowledge system and AI application scenarios be organically integrated into the structural design of project-based textbooks? Third, can teaching practices based on project-based textbooks effectively enhance students' programming abilities and AI literacy [1][3]?

From a theoretical perspective, this study attempts to provide a referable paradigm for programming textbook compilation in the AI era. The compilation logic of traditional textbooks emphasises systematicity and completeness of knowledge, whereas project-based textbooks underscore the applicability and contextualization of knowledge—how to balance and transform these two logics is an issue worthy of in-depth exploration in textbook construction theory [5]. From a practical perspective, the findings of this study can serve as reference cases and operational pathways for the textbook construction and teaching reform of Python programming courses in higher education institutions [2], helping frontline teachers more clearly understand "what a project-based textbook should look like" and "how to use it effectively."

1.3 Domestic and International Research Status

The concept of project-based learning can be traced back to Dewey's educational philosophy of "learning by doing," and after decades of development, it has gained widespread recognition and application globally [6]. In the field of programming education, PBL is generally considered an effective approach to cultivating students' computational thinking and problem-solving abilities [4][7].

Internationally, research on Python project-based teaching has shown clear growth in recent years. Researchers have explored the topic from multiple perspectives: some have focused on integrating project-based learning with AI literacy cultivation, using authentic projects such as drone control and image recognition to enhance students' computational thinking and AI application capabilities, with empirical studies demonstrating that such immersive project experiences significantly deepen students' understanding of AI concepts [6]; others have examined the integration of generative AI tools (e.g., GitHub Copilot, ChatGPT) in project-based programming teaching, investigating how to leverage AI to assist students in completing complex programming tasks while simultaneously maintaining the cultivation of critical thinking and avoiding cognitive laziness caused by technological dependence [7]; still others have proposed a

human-computer collaborative practical teaching model, embedding AI tools into the knowledge acquisition, practical training, evaluation feedback, and personalised tutoring phases of project-based learning, thereby forming a relatively complete instructional loop [6][7]. Furthermore, some scholars, from the perspective of immersive learning strategies, have investigated the integration of virtual simulation environments, gamified programming platforms, and project-based learning in AI certification programmes, finding that project-based learning significantly enhances learning motivation and knowledge transfer capabilities [6].

[Insert Figure 1: Development Trends of Domestic and International Research on Python Project-Based Textbooks and Teaching since 2010] – displaying the evolution trajectories of domestic research from "project-driven teaching exploration" to "AI-empowered PBL" and then to "systematic design of project-based textbooks," as well as the international progression from "application of PBL in programming education" to "AI-assisted project-based learning" and then to "human-computer collaborative teaching models," highlighting the similarities, differences, and convergence points of research hotspots between domestic and international contexts through comparative presentation.

Domestically, related research has similarly exhibited a diversified landscape. In terms of Python teaching reform, Lu Haili et al. [4] adopted a project-driven approach to integrate typical machine learning cases into Python language instruction in a layered and progressive manner, providing sufficient programming practice to help learners master Python and acquire preliminary capabilities for developing AI systems; their teaching practice demonstrated that hierarchical project design effectively reduces cognitive load and enhances learners' sense of achievement. In the realm of AI-empowered teaching, some scholars have proposed a generative AI-enabled "four-stage, dual-chain, three-loop" teaching model, exploring pedagogical innovation in Python courses from dimensions such as teaching model construction, resource system development, and assessment paradigm creation, providing an actionable framework for the deep integration of AI technology and course instruction [3]. In the area of project-based textbook development, existing textbooks have adopted the "project-oriented, task-driven" compilation philosophy, organising Python fundamentals progressively through multiple projects and tasks, enabling students to gradually acquire programming skills while completing specific tasks [5]. Other studies have proposed a PBL-driven "AI-assisted learning, progressive tasking, practice-based competence" three-dimensional teaching model, validated through Python course practice, with results indicating positive effects on students' programming self-efficacy and project quality. At the curriculum reform level, some research takes typical work projects as carriers, constructs course modules tailored to the characteristics of AI technology applications, and employs task-driven and contextualised teaching methods, providing industry-based demand references for the content selection of project-based textbooks.

In summary, both domestic and international research have accumulated considerable achievements in the integration of Python project-based teaching and AI education. However, most existing studies focus on pedagogical models and teaching methods, lacking sufficiently in-depth theoretical construction and empirical research on the systematic design of "textbooks" as the core instructional vehicle in project-based

reform – particularly regarding how to organically integrate the Python knowledge system, AI application scenarios, and project-based learning logic [7]. This precisely identifies the entry point of the present study.

2. Design Philosophy and Compilation Framework of Python Project-Based Textbooks

As one of the most central carriers of teaching activities, the design philosophy and compilation framework of a textbook directly determine the pathways and effectiveness of instructional implementation.

2.1 Overall Design Philosophy of the Textbook

The design of this textbook is guided by the core philosophy of "project-oriented, task-driven, and integrating knowledge with action," embodying the educational thought of "student-centred and competence-based." The establishment of this philosophy stems from a reflection on the traditional logic of textbook compilation – conventional textbooks tend to follow the disciplinary knowledge system as the main thread, unfolding chapter by chapter according to the difficulty of grammatical rules, yet after learning all the grammar, students still do not know how to apply this knowledge to solve a complete problem. Project-based textbooks attempt to reverse this situation: making projects the starting point rather than the end point of learning, and making tasks the opportunity for introducing knowledge rather than mere appendages for knowledge application.

"Student-centred" means that the compilation logic of the textbook should shift from the "perspective of teaching" to the "perspective of learning" – not considering "what is convenient for the teacher to deliver," but rather "what is effective for the student to learn." Every project and every task in the textbook should start from the learner's cognitive starting point, respect their prior knowledge and experience, and set appropriate cognitive challenges within their "zone of proximal development" [8]. "Competence-based" implies that the goal of the textbook is not to have students memorise as many grammatical items as possible, but to enable them to develop programming ability, problem-solving skills, and AI application capabilities through the process of completing projects [9]. Knowledge remains important, but it serves as a means to foster competence, rather than being the ultimate goal of instruction.

In terms of content organisation, the textbook strictly adheres to learners' cognitive laws, adopting the principle of progressing from the simple to the complex and from the shallow to the deep [10]. Beginners should not be thrown into complex comprehensive projects before they have established basic programming intuition; therefore, the early projects focus on focused training on single knowledge points and the independent completion of simple tasks. As learning deepens, project difficulty gradually increases, knowledge integration intensifies, and ultimately leads to full system development and algorithmic innovation practice.

2.2 Structured Design of the Textbook Content System

2.2.1 Reconstruction Logic of the Knowledge System

Traditional Python textbooks generally follow the "linear arrangement of

grammatical knowledge points" paradigm – starting with environment setup, then sequentially introducing data types, operators, flow control, functions, modules, object-orientation, file operations, exception handling, and so on. Each chapter is relatively independent, and there is a lack of organic project threads to connect them. Although this arrangement is convenient for teachers to teach step by step, it makes it difficult for students to understand the collaborative relationships among various knowledge points in actual development.

This textbook breaks this linear arrangement pattern by fully integrating Python's core knowledge points into project tasks. The knowledge system covered by the textbook includes: Python development environment setup and IDE usage, variables and data types, operators and expressions, flow control structures (branching and looping), composite data types (lists, tuples, dictionaries, sets), function definitions and module imports, file read/write operations, exception handling mechanisms, object-oriented programming (classes and objects, inheritance and polymorphism), and more. These knowledge points are not presented in isolation according to traditional chapter sequences, but are introduced "on demand" at corresponding task nodes according to the actual needs of project development – for example, introducing branching and looping structures in the "supermarket shopping checkout system" project, introducing list operations and file reading/writing in the "student grade management system" project, and introducing object-oriented programming and data analysis libraries in the "e-commerce user profiling" project. This "knowledge follows tasks" reconstruction approach enables students to understand the application scenarios and value of knowledge in specific programming practices, avoiding the dilemma of "learning but not knowing how to use it."

2.2.2 Gradient Setting and Layered Design of Projects

The gradient setting of projects is one of the most carefully planned aspects of project-based textbook design. Whether the gradient setting is reasonable directly affects whether students can gain sustained achievement and a sense of growth in project learning. Following the three-stage progressive approach of "basic grammar → comprehensive projects → algorithmic innovation," this textbook divides all projects into three levels.

Level 1: Basic Grammar Layer. Projects at this level focus on single-skill training and simple combinations of Python core grammatical knowledge. The projects are relatively small in scale and have relatively single tasks, aiming to help beginners establish basic programming intuition and confidence. Typical projects include "temperature conversion tool," "calculator program," "guess the number game," etc. These projects share common characteristics: clear knowledge objectives, moderate code volume, short completion cycles (1-2 class hours), and suitability for students with zero programming background to get started quickly.

Level 2: Comprehensive Project Layer. Projects at this level require students to comprehensively apply multiple knowledge points to complete application systems of certain complexity. The projects are of medium scale, involving knowledge integration and system design, aiming to cultivate students' modular programming abilities and problem-decomposition skills. Typical projects include "student information

management system," "library borrowing management system," "takeaway ordering system," etc. These projects usually cover multiple knowledge modules such as function encapsulation, file persistence, exception handling, and object-oriented design, with completion cycles of 3-5 class hours.

Level 3: Algorithmic Innovation Layer. Projects at this level point towards AI applications and algorithm design, requiring students, based on the programming abilities accumulated in the previous two stages, to further engage with cutting-edge content such as data acquisition, data analysis, and machine learning modelling. Typical projects include "recruitment data analysis based on web crawling," "sales data visualisation based on Pandas," "Iris classification based on Scikit-learn," and so on. These projects not only test students' fundamental programming skills but also demand a certain level of data thinking and algorithmic understanding.

The three levels form a clear progressive relationship: the project outcomes of the preceding level lay the knowledge and ability foundation for the projects of the subsequent level, while the projects of the latter level deepen and extend the learning of the former.

2.3 Modular Structure Design of the Textbook

The structural organisation of the textbook is a concrete manifestation of its design philosophy. This textbook adopts a modular project arrangement, with each project unit following a uniform structural framework. Specifically, each project includes the following seven sections，as show in Fig 1:

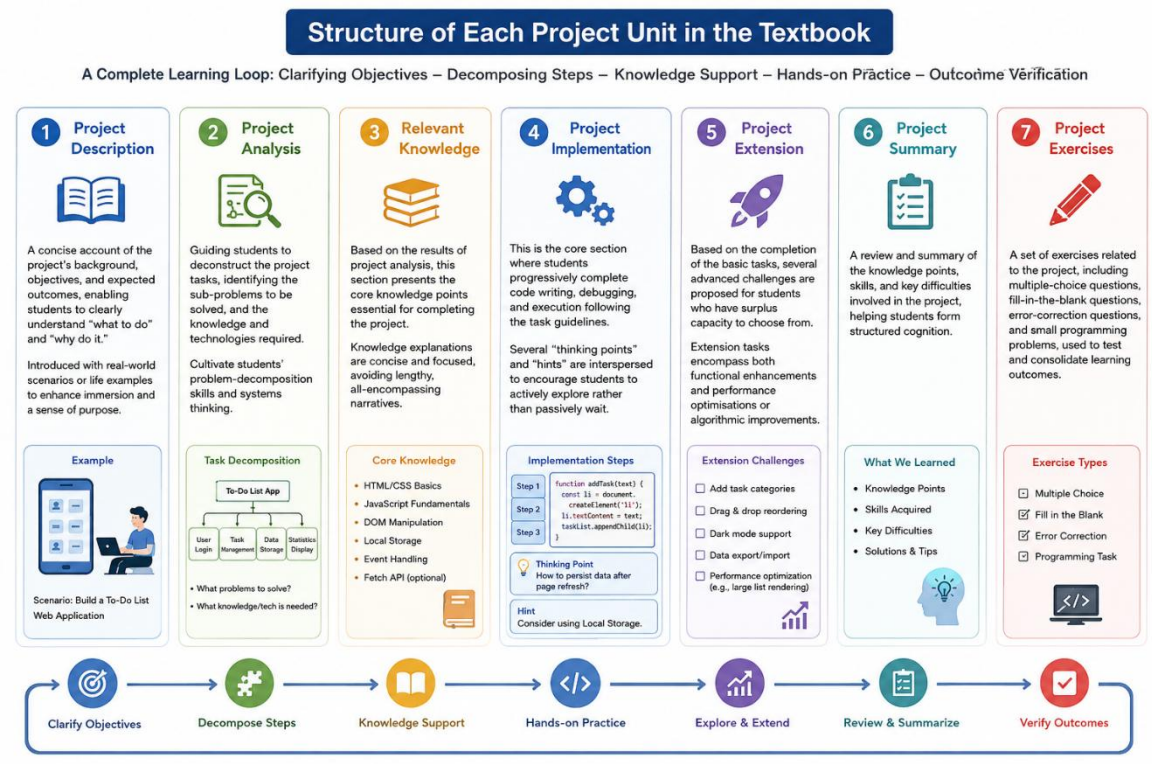

Fig 1. Structure of Each Project Unit in the Textbook

Project Description – A concise account of the project's background, objectives, and expected outcomes, enabling students to clearly understand "what to do" and "why do

it." The project description is usually introduced with real-world scenarios or life examples to enhance immersion and a sense of purpose in learning.

Project Analysis – Guiding students to deconstruct the project tasks, identifying the sub-problems to be solved, and the knowledge and technologies required. This section aims to cultivate students' problem-decomposition skills and systems thinking.

Relevant Knowledge – Based on the results of project analysis, this section presents the core knowledge points essential for completing the project. Knowledge explanations are concise and focused, avoiding lengthy, all-encompassing narratives.

Project Implementation – This is the core section of each project, where students progressively complete code writing, debugging, and execution following the task guidelines. Several "thinking points" and "hints" are interspersed throughout the implementation process to encourage students to actively explore rather than passively wait when encountering difficulties.

Project Extension – Based on the completion of the basic tasks, several advanced challenges are proposed for students who have surplus capacity to choose from. Extension tasks encompass both functional enhancements and performance optimisations or algorithmic improvements.

Project Summary – A review and summary of the knowledge points, skills, and key difficulties involved in the project, helping students form structured cognition.

Project Exercises – A set of exercises related to the project, including multiple-choice questions, fill-in-the-blank questions, error-correction questions, and small programming problems, used to test and consolidate learning outcomes.

These seven sections together constitute a complete learning loop of "clarifying objectives – decomposing steps – knowledge support – hands-on practice – outcome verification."

3. Teaching Practice of Python Project-Based Textbooks in Artificial Intelligence Courses

3.1 Design of the Teaching Practice

3.1.1 Teaching Subjects and Curriculum Settings

The teaching practice of this study selected second-year students majoring in Computer Science and Technology at an application-oriented undergraduate university in Shandong Province, China, as the implementation subjects, involving two natural classes with a total of 77 students. The selection of this grade level was based on two considerations: on the one hand, the students had already completed the "University Computer Fundamentals" course and possessed basic computer operation skills and information literacy, though the vast majority had not yet systematically studied programming; on the other hand, the sophomore year coincides with the critical transitional period from foundational professional courses to core professional courses. Introducing Python project-based teaching at this stage can not only lay the programming foundation for subsequent AI-oriented courses such as "Machine Learning" and "Data Mining," but also help students establish an initial understanding of artificial intelligence technologies in the early stages of their professional studies.

The course is titled "Python Programming" and is designated as a compulsory

professional foundation course, with a total of 64 class hours, of which 48 hours are allocated to theoretical instruction and 16 hours to project practice, spanning a teaching period of 16 weeks. Instruction is conducted in smart classrooms equipped with multimedia teaching facilities and internet-connected computers, with each student using an independent computer to facilitate in-class coding practice.

3.1.2 Teaching Implementation Process

The teaching implementation based on project-based textbooks is designed with reference to the generative AI-empowered "four-stage, dual-chain, three-loop" teaching model. The "four stages" refer to the four progressive phases of instruction – project introduction, knowledge construction, collaborative implementation, and outcome presentation; the "dual chains" refer to the parallel advancement and mutual support of the "knowledge learning chain" and the "project practice chain"; the "three loops" refer to the closed-loop structure of "pre-class preview – in-class practice – after-class extension" formed within each project unit. This framework not only highlights the distinctive feature of project-based teaching of "taking projects as the main thread" but also accommodates the systematic construction of the knowledge system.As show in Fig 2.

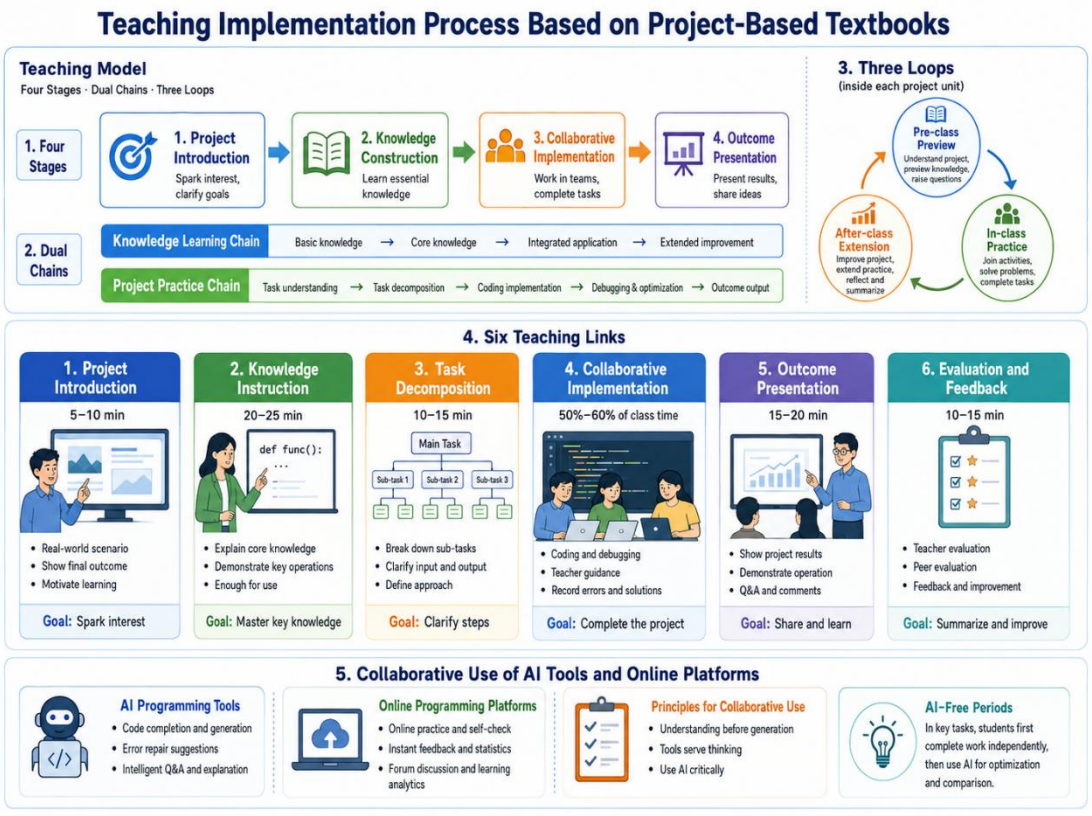

Fig 2 Teaching Implementation Process Based on Project-Based Textbooks

Specifically, the teaching implementation of each project follows the following six links.

Link 1: Project Introduction. The teacher introduces the project theme through real-world scenarios or life examples, stimulating students' curiosity and desire for completion by demonstrating the final outcomes of the project (such as a running

program, a visualisation report, etc.). This introduction session lasts approximately 5 to 10 minutes, focusing on "igniting interest" rather than "instilling knowledge."

Link 2: Knowledge Instruction. The teacher delivers concise explanations and demonstrations on the core knowledge points involved in the project. The instruction follows the principle of "sufficiency for use" – teaching only the knowledge necessary for completing the project, without addressing "standby knowledge" that is not immediately applicable. For example, in the "Data Visualisation" project, emphasis is placed on Pandas data reading and Matplotlib's basic plotting functions, rather than expanding into Pandas' advanced indexing or Matplotlib's complex customisation. The instruction duration is controlled within 20 to 25 minutes.

Link 3: Task Decomposition. The teacher guides students to break down the project into several executable sub-tasks, clarifying the inputs, outputs, and implementation approaches for each. This session can be assisted by mind maps or flowcharts. Task decomposition serves both as cognitive training – cultivating students' decomposition thinking skills – and as a teaching management strategy – transforming large projects into manageable steps to reduce cognitive load and learning anxiety.

Link 4: Collaborative Implementation. Students progressively complete code writing, debugging, and execution according to the results of task decomposition. The teacher circulates within the classroom for guidance, providing collective explanations for common issues and one-on-one troubleshooting for individual problems. Collaborative implementation is the most time-consuming phase of the project, typically accounting for 50% to 60% of the total project class hours. During this process, students are encouraged to follow the implementation guidelines in the textbook and are required to document the errors encountered and their corresponding solutions.

Link 5: Outcome Presentation. Each group selects a representative to present the project outcomes, including operational demonstrations and explanations of key code segments. During the presentation, other groups may ask questions and provide comments. The outcome presentation serves both as an inspection of project quality and as an opportunity for mutual learning.

Link 6: Evaluation and Feedback. The teacher provides summative evaluation of project completion, identifying common issues and directions for improvement, while also organising intra-group and inter-group peer evaluations. The evaluation dimensions cover code standardisation, functional completeness, innovativeness, and teamwork. Feedback is delivered through a combination of written and oral means to ensure that each student receives targeted suggestions for improvement.

3.2 Collaborative Application of Textbooks and AI Tools

Against the backdrop of the rapid development of generative artificial intelligence, the ecosystem of programming education is undergoing profound changes. This textbook fully considers the rational utilisation of AI-assisted programming tools in its design, and explores the collaborative application model of textbooks and AI tools in teaching practice.

In terms of the application of AI-assisted programming tools, two types of AI-assisted programming tools were introduced during the teaching process. One type

is code completion and generation tools, which students can use to obtain code completion suggestions, function usage hints, and error repair recommendations while writing code. The other type is intelligent question-answering tools, which students can consult through natural language queries to obtain explanations and guidance when encountering conceptual difficulties or debugging challenges. To ensure equal access for all students, the course uniformly recommended free or educationally discounted versions of AI tools and arranged training on tool registration and basic usage during the first week of instruction.

In terms of the application of online programming platforms, the course incorporated online programming practice platforms (such as Python123, etc.), where students could complete project exercises from the textbook. The platforms automatically performed code correctness checks and style standardisation inspections, providing immediate feedback. The discussion forum functionality on the platforms offered students a space for asynchronous communication, and teachers could also access backend data to understand each student's code submission frequency, error type distribution, and other learning behaviour information.

In terms of the guiding principles for collaborative application, the teacher consistently emphasised three principles to students in the collaborative use of textbooks and AI tools. First, "understanding precedes generation" – students must be able to read and comprehend every line of AI-generated code, understanding its logic and function, rather than simply copying and pasting. Second, "tools serve thinking" – the value of AI tools lies in improving efficiency and reducing the burden of mechanical labour, but they cannot replace students' higher-order thinking activities such as algorithmic design, problem decomposition, and logical reasoning. Third, "critical use" – AI-generated code may contain errors or suboptimal solutions, and students need the ability to judge and correct, rather than blindly trusting. To implement the above principles, the teacher established some "AI-free periods" during the project implementation phase – in key tasks (such as the implementation of core algorithms), students were required to complete the code independently before using AI tools for optimisation comparison, thereby ensuring that students experienced the complete thinking process.

4. Teaching Effectiveness Evaluation and Analysis

The evaluation of teaching effectiveness is a critical link in examining the outcomes of textbook reform. This chapter, starting from the design of the evaluation indicator system, systematically presents the evaluation methods and data collection processes, then analyses and discusses the evaluation results across dimensions such as academic performance, learning attitudes and interests, practical abilities, and AI literacy, ultimately forming a comprehensive judgement on the teaching effectiveness of the project-based textbook.

4.1 Design of the Evaluation Indicator System

To comprehensively and objectively evaluate the teaching effectiveness of the Python project-based textbook, this study constructed a multi-dimensional evaluation indicator system. The selection of indicators followed three principles: comprehensiveness – covering multiple aspects such as knowledge, abilities, and

attitudes; operability – indicators should be observable, quantifiable, and data-accessible; pertinence – closely corresponding to the core objectives of project-based teaching and the specific requirements of AI education.

The evaluation indicator system encompasses five first-level dimensions, with several second-level indicators under each dimension.

Dimension 1: Knowledge Mastery. This assesses students' level of understanding and retention of Python foundational grammar and core concepts. Second-level indicators include: grammar knowledge mastery (measured through multiple-choice and fill-in-the-blank questions), program reading ability (measured through code-reading questions), and depth of conceptual understanding (measured through short-answer questions). This dimension focuses on achievement at the cognitive level.

Dimension 2: Practical Ability. This assesses students' capability to use Python to solve real programming tasks. Second-level indicators include: code writing standardisation (measured through project code review), program debugging and troubleshooting ability (measured through error-correction questions and debugging tasks), and project completion quality (measured through project work evaluation).

Dimension 3: Computational Thinking. This assesses students' computational thinking abilities, including problem decomposition, abstract modelling, algorithmic design, and iterative optimisation. Second-level indicators include: problem decomposition ability (evaluated through project analysis reports), algorithm design and expression ability (evaluated through core algorithm implementation), and systems thinking level (evaluated through overall project architecture).

Dimension 4: Learning Interest and Attitude. This assesses students' learning motivation, interest, and self-efficacy in Python programming and the AI field. Second-level indicators include: course learning interest (measured through questionnaire surveys), programming self-efficacy (measured through questionnaire surveys), and learning initiative (measured through classroom participation and extracurricular practice volume).

Dimension 5: AI Application Ability. This assesses students' ability to use AI-related tools and libraries in the Python ecosystem to complete foundational AI tasks. Second-level indicators include: mastery of AI fundamentals (measured through knowledge tests), AI algorithm library application ability (evaluated through library usage in projects), and appropriate use of AI tools (evaluated through the quality of AI usage descriptions). Regarding the weight allocation across dimensions, considering the "competence-based" core orientation of project-based teaching, the practical ability and computational thinking dimensions each account for 25%, knowledge mastery accounts for 20%, learning interest and attitude accounts for 15%, and AI application ability accounts for 15%. The weights were determined through discussion by three computer education experts with associate professor or higher titles using the Delphi method.

4.2 Evaluation Methods and Data Collection

This study comprehensively employs multiple evaluation methods to collect teaching effectiveness data from different perspectives, striving to form triangulation.

In terms of questionnaire surveys, questionnaires were distributed twice, at the beginning of the semester (Week 1) and at the end of the semester (Week 16). The

questionnaire contents included: a learning interest scale (5 items, using a 5-point Likert scale), a programming self-efficacy scale (6 items), and an AI cognition and attitude scale (4 items). The questionnaires were distributed and collected online through the "Wenjuanxing" platform. At the beginning of the semester, 74 valid questionnaires were collected (response rate 96.1%), and at the end of the semester, 72 valid questionnaires were collected (response rate 93.5%).

In terms of final grade comparative analysis, the final assessment adopted a structure of "theoretical written examination (40%) + project practice assessment (60%)." The theoretical written examination covered multiple-choice questions, fill-in-the-blank questions, program reading questions, and short-answer questions, with a full score of 100. The project practice assessment required students to independently complete a comprehensive programming project and submit a report based on the projects learned during the semester. The final grades of the experimental class (n=38) using the project-based textbook were compared with those of the control class (n=39) using the traditional textbook during the same period. The control class maintained consistency with the experimental class in terms of course objectives, class hour arrangements, and assessment methods, with the only difference being the textbook type and the corresponding teaching approach.

In terms of project work evaluation, four major project works completed by students during the semester were evaluated. Each project was independently scored by two teachers, and the mean value was taken. The scoring criteria covered four dimensions: functional completeness (30 points), code standardisation (25 points), algorithmic and design rationality (25 points), and innovativeness (20 points).

In terms of classroom observation and in-depth interviews, the course instructor recorded classroom observation logs each teaching week, focusing on students' classroom participation, group collaboration quality, and common types of difficulties. At the end of the semester, 12 students were randomly selected (covering high, medium, and low academic achievement levels) for semi-structured in-depth interviews, each lasting approximately 20 to 30 minutes, covering learning experiences, textbook usage perceptions, and AI tool usage.

4.3 Evaluation Results and Analysis

4.3.1 Comparative Analysis of Academic Performance

The final grade statistics revealed that the experimental class's average score was 8.5 points higher than that of the control class. The independent samples t-test indicated that the difference between the two groups was statistically significant ($t=3.87$, $p<0.01$). In terms of score distribution, the excellence rate (≥90 points) of the experimental class was 23.7%, compared to 10.3% in the control class; the pass rate (≥60 points) of the experimental class was 97.4%, compared to 84.6% in the control class. The proportion of students in the low-score segment (<60 points) was 2.6% in the experimental class, significantly lower than the 15.4% in the control class. This distribution pattern indicates that the project-based textbook not only improved students' overall academic performance but also effectively reduced the proportion of academically struggling students. Further analysis revealed that the gap between the experimental and control

classes was primarily reflected in the project practice assessment component – the experimental class averaged 86.2 points in project practice, while the control class averaged 73.8 points, a difference of 12.4 points; the gap in the theoretical written examination component was relatively smaller (78.5 points in the experimental class vs. 74.6 points in the control class). This suggests that the advantages of the project-based textbook are primarily manifested in the cultivation of practical abilities, rather than mere knowledge memorisation.

4.3.2 Changes in Learning Attitudes and Interests

Based on the paired-samples t-test results from the pre-semester and post-semester questionnaire data, students in the experimental class showed significant changes across multiple attitudinal dimensions. In terms of learning interest, the mean value was 3.42 at the beginning of the semester (on a 5-point Likert scale) and increased to 4.18 at the end of the semester, with an improvement of 0.76 ($t=5.21$, $p<0.001$). In interviews, several students expressed that "they used to think programming was about memorising grammar, but now they find that programming can really solve practical problems" – this cognitive shift from "learning grammar" to "solving problems" represents an important psychological mechanism for the enhancement of learning interest. In terms of programming self-efficacy, the mean value was 3.15 at the beginning of the semester and increased to 3.89 at the end, with an improvement of 0.74 ($t=4.93$, $p<0.001$). Notably, the improvement in self-efficacy among students with zero prior programming experience (1.02) was significantly higher than among those with prior programming experience (0.41), indicating that the "progressive" design of the project-based textbook is particularly friendly to beginners – they gradually build the confidence of "I can do it" through completing one project after another. In terms of learning initiative, the end-of-semester questionnaire showed that 72.2% of students reported "frequently writing code practice outside of class," compared to only 29.7% at the beginning of the semester. The classroom observation logs also confirmed this change – during the second half of the semester, there was a notable increase in students actively discussing programming problems during breaks and spontaneously organising study groups after class.

4.3.3 Enhancement of AI Literacy

In terms of AI application ability, the average correct rate on the end-of-semester AI fundamentals test was 73.6%, indicating that students had established an initial cognitive framework for basic machine learning concepts (such as training/testing set partitioning, the distinction between supervised and unsupervised learning, etc.). In terms of AI algorithm library application, 86.1% of students were able to independently use Pandas for data cleaning and aggregation operations in their projects, 79.2% were able to use Matplotlib for basic data visualisation, and 61.1% were able to use Scikit-learn to complete simple classification tasks. Regarding the appropriate use of AI tools, the quality evaluation of the end-of-semester "AI Usage Descriptions" showed that 68.1% of students were able to clearly indicate the AI-assisted generated portions and explain their own modifications, indicating that most students had developed an awareness and habit of "critically using AI tools."

5. Conclusions and Prospects

5.1 Research Conclusions

Centering on the core theme of the application of Python project-based textbooks in artificial intelligence education, this study has progressed through the articulation of textbook design philosophy, the construction of a compilation framework, the implementation of teaching practice, and the evaluation of teaching effectiveness, leading to the following main conclusions.

First, the design philosophy of project-based textbooks effectively responds to the structural dilemmas of programming education in the AI era. The "project-oriented, task-driven, and integrating knowledge with action" design philosophy proposed in this study breaks away from the path dependence of traditional textbooks that linearly arrange content according to grammatical knowledge points, embedding knowledge learning within the process of completing real-world projects. Teaching practice has demonstrated that this design philosophy offers significant advantages in reducing beginners' cognitive load and enhancing their sense of meaning and achievement in learning. The project-based textbook is not a simple patch to traditional textbooks, but rather a fundamental re-answer to the question of "how knowledge should be taught and learned" in programming education.

Second, project-based teaching significantly enhances students' programming practical abilities and AI application literacy. Comparative data between the experimental class and the control class show that the experimental class using the project-based textbook achieved significantly higher final overall scores than the control class (with an average score 8.5 points higher, $p<0.01$), and the gap was primarily reflected in the project practice assessment dimension. Project work evaluations indicate that students' scores across the four projects showed a progressive upward trend, demonstrating that the gradient design of the project-based textbook effectively supports the gradual development of competencies. In terms of AI application ability, the majority of students were able to independently use AI algorithm libraries such as Pandas and Matplotlib in their projects to complete data processing and visualisation tasks, and initially established a cognitive framework for machine learning workflows.

Third, the project-driven model effectively bridges the gap between theoretical instruction and practical application. The common dilemma in traditional programming teaching of "learning but not knowing how to use it" has been substantially alleviated under the project-based teaching framework. In the process of completing authentic projects, students naturally encounter and resolve the application of grammatical knowledge in specific contexts. Knowledge is no longer an abstract object of memorisation, but rather a tool for solving problems. The cognitive shift from "learning grammar" to "solving problems" constitutes the psychological mechanism underlying the significant enhancement of students' programming self-efficacy.

Fourth, the collaborative application of textbooks and AI-assisted tools requires clear norms and boundaries. The practice of this study indicates that the appropriate use of AI-assisted programming tools can significantly improve coding efficiency; however, in the absence of clear regulatory constraints, students may easily fall into the trap of "over-dependence." Through the implementation of an "AI Usage Description" system

and the establishment of "AI-free periods" for key tasks, a balance can be achieved between efficiency enhancement and thinking cultivation.

5.2 Limitations and Deficiencies of the Study

Although this study has achieved certain results in both theoretical exploration and teaching practice, several unavoidable limitations remain.

First, the representativeness of the sample is limited. This study only targeted 77 students majoring in Computer Science at one application-oriented undergraduate university, with a relatively small sample size involving only one institution and one discipline. Students from different types of institutions (research universities, vocational colleges) and different disciplinary backgrounds (humanities, engineering, sciences) may exhibit differences in acceptance and adaptability to project-based textbooks, and the external validity of the study's conclusions requires further testing.

Second, the practice period is relatively short. This study only implemented one semester (16 weeks) of teaching practice. The formation of programming ability and the cultivation of AI literacy are long-term cumulative processes. A shorter practice period may not fully capture the impact effects of project-based textbooks on students' long-term development, nor can it comprehensively assess the textbook's performance in terms of knowledge retention rates and sustained learning willingness.

Third, the evaluation instruments require further refinement. In this study, dimensions such as computational thinking and innovative transfer ability lack mature and standardised measurement tools. Although project work scoring adopted a double-blind independent evaluation mechanism, subjective bias remains difficult to completely avoid. Moreover, the evaluation dimensions for AI literacy are relatively preliminary, failing to comprehensively cover deeper competencies such as AI ethical awareness and data critical thinking.

Fourth, the generalisability of the textbook needs further validation. The project-based textbook designed in this study was developed for a specific course (Python Programming) and a specific target group (second-year Computer Science students). Whether its project selection, difficulty gradient, and modular structure are applicable to other course contexts remains to be further tested through practice.

5.3 Future Research Directions

Based on the findings and limitations of this study, subsequent research can be further pursued in multiple directions. At the level of textbook content, an AI-empowered "textbook-technology" collaborative update mechanism could be explored, utilising artificial intelligence tools to automatically track technological evolution and industry demand changes, assisting compilation teams in promptly updating project cases and knowledge content to maintain the textbook's dynamic vitality. At the level of learning pathways, learning analytics and recommendation algorithms could be integrated to develop an intelligent learning path recommendation system, dynamically delivering differentiated project sequences and learning resources according to students' knowledge levels, learning styles, and project completion status, thereby achieving genuine individualised instruction. At the level of teaching resources, project-based textbooks could transcend the limitations of paper media by integrating multimodal

resources such as micro-video lectures, interactive coding exercises, virtual simulation experiments, and AI-powered intelligent Q&A into a multidimensional learning support system, evolving the textbook from a "reading text" into a "learning environment." At the level of application scope, interdisciplinary project-based textbooks could be developed for various disciplinary backgrounds including data science, financial analysis, bioinformatics, and digital humanities, using authentic disciplinary problems as project contexts and Python as the problem-solving tool to simultaneously enhance programming ability and professional competence. Against the backdrop of the increasing prevalence of AI-assisted programming, traditional programming assessment methods are facing fundamental challenges; future research urgently needs to explore new evaluation paradigms – shifting from "examining code writing ability" to "examining problem-definition ability, algorithm design ability, code comprehension ability, and critical evaluation ability" – so that assessment truly targets the irreplaceable core competencies of human intelligence. The above directions are interrelated and mutually supportive, collectively pointing toward the evolutionary trajectory of project-based textbooks from "static texts" to "intelligent ecosystems."

Funding: This study was supported by the following research grants:

Innovation and Research on Project-Based Teaching Materials for Python Courses for Artificial Intelligence Education (2025 "Project of Artificial Intelligence-Enabled Higher Vocational Education Curriculum and Teaching Material System Construction", Informatization Teaching Steering Committee for Vocational Colleges of Ministry of Education of China, Grant No. KT2505072)

Research on Practical Application and Quality Evaluation of Multimodal AI Teaching Assistants in Blended Learning for Adult Higher Education (Project of Shandong Adult Higher Education Research Association, Grant No. 2025CRGDJY057)

References

[1] Wang, S. (2026). PBL-driven teaching reform and practice of Python programming course – Based on the "AI-assisted learning, progressive tasking, practice-based competence" three-dimensional teaching model. China Information Technology Education, (6), 92-95.

[2] Li, L., Li, D., Yang, H., & Zhang, Y. (2023). Curriculum reform of "Python Programming" in the context of artificial intelligence. Modern Information Technology , 7(17), 178-182.

[3] Chen, X., Huang, Z., & Hu, H. (2025). Teaching design and practice of Python programming course empowered by generative AI. Computer Education , (12), 109-113.

[4] Lu, H., Wang, B., & Gao, X. (2022). Design and practice research of project-based learning in the intelligent era – Taking PBL teaching of Python course as an example. Heilongjiang Education (Higher Education Research and Evaluation) , (2). https://mall.cnki.net/magazine/article/HLKB202202009.htm

[5] Ge, D., Jin, L., Wang, C., Liu, E., & Luo, H. (2021). Exploration of Python language teaching for artificial intelligence. Journal of Electrical and Electronic Teaching , 43(3), 5.

[6] Wang, S. I. C., Liu, E. Z. F., Huang, Y. Y., & Sang, H. Y. (2024). When drone meets AI education: Boosting high school students' computational thinking and AI literacy. 2024 Pacific Neighborhood Consortium Annual Conference and Joint Meetings (PNC) , 45-52.

[7] Han, N., Zhang, F., Lan, S., Li, Y., & Chen, D. (2024). Construction and implementation of human-machine

collaborative practice teaching mode based on AI – Take the "Python and Web Crawler" lesson as an example. Frontiers in Artificial Intelligence and Applications .

[8] Yang, L., & Li, T. (2024). Exploration of project-based loose-leaf textbook development in vocational colleges – Taking information technology majors as an example. New Curriculum Teaching (Electronic Edition) , (7), 176-178.

[9] Wang, H. (2023). Exploration and practice of computer general education textbook construction from the perspective of "New Engineering." Journal of Anhui Vocational College of Police Officers , 22(4), 112-116.

[10] Kokotsaki, D., Menzies, V., & Wiggins, A. (2016). Project-based learning: A review of the literature. Improving Schools , 1365480216659733.

[11] Osunbunmi, I., & Fang, N. (2024). Work in progress: An early look into the systematic review of project-based learning in engineering education.