Journal of Computer and Communications, 2025, 13(8), 205-221
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

AI Large Language Model-Driven Pedagogical Reform and Practice for the Python Programming Course: A Case Study in the Electronic Information Engineering Program

Zhengjie Wang

College of Electronic and Information Engineering, Shandong University of Science and Technology, Qingdao, China
Email: cieewangzj@163.com

How to cite this paper: Wang, Z.J. (2025) AI Large Language Model-Driven Pedagogical Reform and Practice for the Python Programming Course: A Case Study in the Electronic Information Engineering Program. *Journal of Computer and Communications*, **13**, 205-221.
https://doi.org/10.4236/jcc.2025.138010

Received: August 4, 2025
Accepted: August 19, 2025
Published: August 22, 2025

Abstract

This paper explores the integration of Artificial Intelligence (AI) large language models to empower the Python programming course for junior undergraduate students in the electronic information engineering program. We propose a reform framework driven by dual innovations in pedagogical models and curriculum content. In terms of pedagogy, AI large language models serve as intelligent teaching assistants for instructors and personalized tutors for students, significantly enhancing teaching efficiency and learning experiences through human-AI collaboration. Regarding curriculum, the course content is deeply integrated with AI technology, featuring a modular project cluster that progresses from collaborative learning of Python fundamentals to data analysis and web applications, culminating in a typical project on advanced AI applications such as semantic image segmentation and large language model fine-tuning. A semester-long implementation of this model demonstrated a significant improvement in students' learning interest, engineering practice capabilities, and depth of understanding of cutting-edge AI technologies. Furthermore, it fostered critical thinking, AI ethics awareness in the AI era. This study provides a systematic, feasible implementation plan and a valuable reference for reforming programming courses within the context of New Engineering Education.

Keywords

AI, Large Language Models, Pedagogical Reform, Python Programming, Project-Based Learning

1. Introduction

The wave of Artificial Intelligence (AI), represented by Large Language Models (LLMs), is profoundly reshaping global industries, technological innovation, and higher education systems [1]. For higher education, this technological revolution not only necessitates the cultivation of new talent adapted to future industrial demands but also requires a fundamental reassessment and reconstruction of traditional teaching content, methods, and philosophies [2]. In this context, programming courses, as a core component for developing students' computational thinking and engineering practice skills, are at the forefront of facing these unprecedented challenges [3].

The limitations of traditional teaching models for the Python programming course, particularly in practice-intensive majors like the electronic information engineering program, have become increasingly apparent [4]. First, the curriculum content often lags technological advancements, with topics confined to basic syntax, standard libraries, and conventional data processing, creating a significant disconnect from the rapidly evolving applications of AI. Second, a gap exists between academic projects and industry demands. Traditional course projects are typically small-scale, validation-oriented exercises that fail to simulate the complexity of real-world engineering problems, thus inadequately preparing students for cutting-edge technological challenges. Finally, the teacher-centric instructional model struggles to meet the diverse and individualized needs of students. A uniform teaching pace and fixed Q&A sessions cannot accommodate variations in students' foundational knowledge, interests, and learning rhythms, thereby limiting both learning efficiency and engagement.

In the field of programming education, scholars worldwide have begun to actively explore the integration of LLMs [5]. Existing research can be broadly categorized into two dominant models:

1) Model 1: AI as an Auxiliary Programming Tool. This model positions LLMs (e.g., GitHub Copilot) as productivity tools for code completion, debugging, syntax lookup, and code optimization [6] [7]. Research in this area primarily evaluates the impact of these tools on students' coding speed, code quality, and learning motivation. While these studies have validated the value of AI tools in enhancing programming efficiency, their limitations are also evident. The integration remains superficial, failing to deeply connect with the course's educational objectives and content structure. This approach may even foster cognitive laziness, leading to students knowing what works without understanding why, thereby weakening their grasp of underlying logic and design principles.

2) Model 2: AI as the Main Subject of Instruction. This model treats AI technologies, particularly machine learning and deep learning theories, as the core instructional content [8]. The curriculum focuses on topics such as neural networks, model training, and algorithmic principles. The value of this approach lies in imparting cutting-edge AI theory to students. However, its shortcoming is that the teaching methodology itself does not innovate in synergy with AI tools. Students

remain passive recipients of knowledge, lacking the experience of learning by doing and collaborating effectively with AI to tackle complex tasks.

In summary, existing research tends to treat AI either as an external tool [9] or as purely knowledge, failing to organically combine the two into a closed-loop pedagogical system. The innovation of this paper, therefore, lies in proposing a full, tripartite reform framework encompassing pedagogical model innovation, curriculum restructuring, and strategic implementation support. This framework not only focuses on leveraging AI to improve the efficiency of teaching and learning but also commits to restructuring the curriculum to keep up with the advancements in AI technology. The ultimate objective is to guide students, through carefully designed strategies, from being passive knowledge consumers to active, inquiry-based learners, cultivating well-rounded engineering problem-solvers who can adeptly "use AI" for efficient learning and innovation, but also thoughtfully "create AI" with a strong sense of critical thinking and ethical responsibility.

To solve these typical problems, we implement our reform and practice using a Python class. Python language, with its concise syntax, rich ecosystem of libraries, and seamless integration with mainstream AI frameworks like PyTorch and TensorFlow, has become the "first language" of the AI era. This positions the Python programming course as an ideal testbed for exploring AI-empowered pedagogical reforms. Integrating AI large language models into the Python curriculum is not merely a technological upgrade but a leap in educational philosophy [10].

Therefore, the core significance of this paper lies in exploring and implementing a new pedagogical paradigm that deeply integrates AI both as an auxiliary teaching tool (the means) and as core course content (the end). We aim to break the constraints of traditional teaching models by constructing a human-AI collaborative learning environment and a three-stage progressive, project-based curriculum. The goal of this paper is to cultivate versatile capable of both "using AI" to learn and solve problems efficiently and "creating AI" for technological innovation, thereby providing a concrete solution and practical reference for New Engineering Education to address the challenges of the AIGC era. The basic structure of this article is shown in **Figure 1**.

This paper is organized into five sections as follows. Section 1: Introduction. It outlines the research background and significance, analyzes the pain points of traditional Python instruction, reviews the relevant literature, and clarifies the innovative aspects and research framework of this study. Section 2: A New Human-AI Collaborative Pedagogical Paradigm. This section elaborates on the dual roles of AI large language models as an "intelligent teaching assistant" for instructors and a "personalized tutor" for students, and proposes implementation strategies and guiding mechanisms to ensure pedagogical effectiveness. Section 3: AI-Driven Three-Stage Progressive Curriculum Restructuring. This section details the core of the reform: restructuring the curriculum into three progressive stages—"Foundational Skills", "Applied Practice", and "Frontier Exploration"—and illustrates the AI empowerment points in each stage with specific project examples. Section

4: Implementation and Performance Evaluation. This section describes the implementation process of the pedagogical reform, establishes a multi-dimensional evaluation system, assesses the reform's effectiveness through quantitative and qualitative analysis, and reflects on the challenges encountered. Section 5: Conclusion and Future Work. This section summarizes the main findings of the study and provides an outlook for future work, including expansion, deepening, and extension of the research.

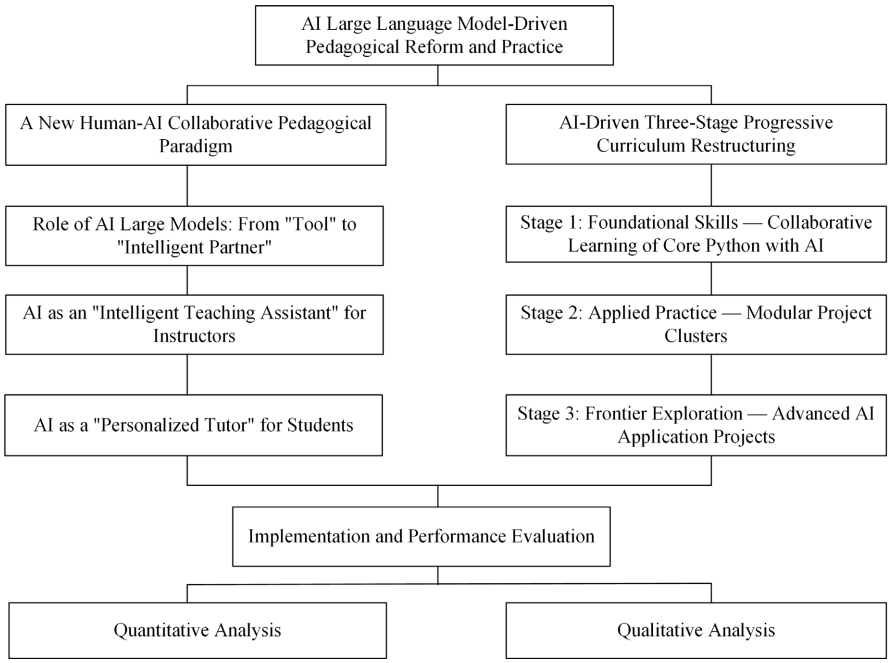

Figure 1. The basic structure of this article.

2. A New Human-AI Collaborative Pedagogical Paradigm

In traditional teaching models, knowledge transmission is primarily a one-way or limited two-way flow from instructor to student. The introduction of AI large language models aims to disrupt this linear structure, creating a learner-centered, dynamic learning ecosystem characterized by deep human-AI collaboration. This section details the composition, role-setting, and implementation strategies of this new pedagogical paradigm.

2.1. Role of AI Large Language Models: From "Tool" to "Intelligent Partner"

In our pedagogical reform, we elevate the role of AI large language models from a passive, command-responding "advanced tool" to an "intelligent partner" actively involved in the entire teaching process. It is no longer merely a code generator or error checker but a "smart hub" connecting instructors, students, and knowledge. This role is central to the four core parts of teaching: instruction, learning, practice, and assessment, creating a closed-loop empowerment cycle.

Instruction: AI assists instructors in efficient course preparation and the design

of innovative teaching activities.

Learning: AI provides students with 24/7 personalized learning support and inspiration.

Practice: AI generates a wide variety of exercises and project scenarios, offering real-time interaction.

Assessment: AI assists instructors with preliminary code evaluation and feedback, making assessment more efficient and formative. In this manner, the AI large language model becomes an integral, tireless, and knowledgeable partner for both instructors and students, collaboratively dedicated to improving educational quality and learning outcomes.

2.2. AI as an "Intelligent Teaching Assistant" for Instructors

For instructors, AI large language models significantly reduce the time spent on repetitive and administrative tasks, allowing them to focus on high-level curriculum design, heuristic guidance, and the development of students' higher-order thinking skills [11]. The "Intelligent TA" functions include:

1) Lesson Planning Support: Instructors can leverage LLMs to enhance the efficiency and quality of their preparation. For example, an instructor can input the prompt: "Please design an 8-week 'Python Programming' syllabus for junior undergraduate students in the electronic information engineering major. The syllabus should cover fundamental syntax, data analysis, and culminate in an application of PyTorch for image processing". The model can quickly generate a structured draft. For specific concepts (e.g., Python decorators), the instructor can ask the model to generate code examples of varying styles and difficulties, as well as vivid analogies for classroom explanation, thereby enriching teaching materials. Generating in-class quizzes is equally efficient, ensuring timely and interactive activities.

2) Assignment Design: Traditional assignment design is time-consuming and prone to repetition. LLMs can automatically generate diverse programming exercises based on specified knowledge points, including multiple-choice questions, fill-in-the-blanks, and code debugging tasks. Furthermore, instructors can ask the model to design small-scale projects based on specific scenarios, such as: "Design a small Python project assignment where students read a CSV file with simulated sensor data, calculate the mean and standard deviation for each sensor type, and visualize the results using Matplotlib". This makes assignments more relevant to real-world applications.

3) Preliminary Assessment: Grading a large volume of student programming assignments is a burdensome and highly repetitive task. AI can perform a preliminary analysis of submitted code, automatically checking for code style compliance (e.g., PEP 8 standards), identifying obvious syntax or logical errors, and providing initial optimization suggestions (e.g., "A list comprehension could make this code more concise"). This significantly reduces the instructor's grading workload, enabling them to concentrate on providing higher-level feedback on students' program architecture, algorithm design, and problem-solving approaches.

4) High-Quality Resource Recommendation: As a vast knowledge aggregator, LLMs can recommend high-quality learning resources to both instructors and students. The recommendations provided by the AI, such as the tables below, were consistent with established authoritative sources. The recommendations include websites in Table 1 and books in Table 2.

Table 1. Resource recommendation of websites.

No	Name	Website	Description
1	Python Official Documentation	https://docs.python.org/3/	The most authoritative reference for syntax and standard libraries.
2	PyTorch Official Tutorials	https://pytorch.org/tutorials/	The best starting point for learning the deep learning framework, with numerous examples.
3	Scikit-Learn Official Examples	https://scikit-learn.org/stable/auto_examples/index.html	A rich collection of machine learning algorithm application cases.

Table 2. Resource recommendation of classic books.

No	Name	Author	Description
1	The Quick Python Book, 3rd	Naomi Ceder	An excellent book for both beginners and intermediate learners.
2	Python Data Science Handbook: Essential Tools for Working with Data, 2nd	Jake VanderPlas	A classic in the data science field, systematically introducing core libraries like NumPy, Pandas, and Matplotlib
3	Machine Learning with PyTorch and Scikit-Learn	Sebastian Raschka, Yuxi (Hayden) Liu, *et al.*	A practical guide that bridges traditional machine learning and deep learning.
4	Build a Large Language Model (From Scratch)	Sebastian Raschka	A cutting-edge book that helps students understand the underlying principles of LLMs.
5	Understanding Deep Learning	Simon J.D. Prince	A book that explains the theoretical foundations of deep learning from first principles.

2.3. AI as a "Personalized Tutor" for Students

For students, an AI large model acts as an ever-present and infinitely patient "personalized tutor", effectively addressing learning gaps caused by individual differences in a traditional classroom and fostering an atmosphere of autonomous, inquiry-based learning [12] [13].

1) Instant Concept Clarification: Students can ask the AI about any concept they encounter and receive immediate, multi-faceted explanations. Compared to search engines, AI's responses are more conversational and targeted. For instance,

a student could ask, "Explain Python decorators using an analogy of ordering food at a restaurant". The AI can generate an intuitive and easy-to-understand comparison, helping the student build a solid mental model. This instant feedback mechanism greatly reduces learning friction and frustration.

2) Code Debugging and Heuristic Guidance: Encountering bugs is a normal and often discouraging part of learning to program. Students can provide their erroneous code and error messages to the AI and request heuristic guidance. The key is to shift the student's questioning style from "Give me the correct answer" to "Help me analyze the possible causes of this TypeError" or "Guide me step-by-step to locate this logical error instead of just fixing it". In this role, the AI acts not as an answer provider but as a knowledgeable and tireless mentor. Through iterative questioning and hints, the AI guides students to discover the solution themselves, thereby developing their independent problem-solving skills.

3) Project Ideation and Exploration: During the project-based learning phase, the AI can serve as a "brainstorming partner". When students are unsure about a project topic, they can describe their interests to the AI, for example: "I am interested in audio signal processing and machine learning. Can you suggest a few moderately difficult Python project ideas"? The AI might suggest topics like "Speech Emotion Recognition" and provide potential technology stacks (e.g., using Scikit-Learn or PyTorch), effectively broadening students' creative horizons.

2.4. Implementation Strategies and Guiding Mechanisms

To ensure the new AI-empowered pedagogical paradigm is effective and prevents its misuse as a tool for "laziness", a clear and robust set of implementation strategies and guiding mechanisms is essential.

1) AI Usage Policy and Academic Integrity Education: At the outset of the course, clear rules and boundaries for AI use must be established. For instance, our policy permitted the use of AI for brainstorming, concept clarification, debugging, and generating code snippets, but strictly prohibited the direct submission of entire projects or core functional modules generated by AI. AI-powered code similarity detection tools were used for auxiliary verification, with academic integrity education integrated throughout the course.

2) Prompt Engineering Training: We believe that the ability to effectively query an AI is a core skill in the AI era. Therefore, we dedicated one to two class hours at the beginning of the course to teach the fundamentals of prompt engineering. The instruction covered how to define the AI's role, provide enough context, make clear and specific requests, and iterate through follow-up questions. We design useful prompts, such as: "For the topic 'Explain Python's GIL (Global Interpreter Lock), design three prompts with different levels of complexity (one for a beginner, one for an experienced developer, and one requesting an analogy). Compare and analyze the quality of the AI's responses and the reasons for the differences".

3) Assessment Focused on Critical Thinking: To fundamentally prevent cognitive inertia, we overhauled the course assessment methods. The weight of rote mem-

orization of code was reduced, while the evaluation of higher-order thinking skills was increased. The new assessment system emphasized understanding. Students were required to detail the architectural design and technology choices in their project reports and were encouraged to explore multiple implementation paths and analyze the pros and cons of different solutions, including those suggested by AI.

3. AI-Driven Three-Stage Progressive Curriculum Restructuring

This section presents the core of our pedagogical reform. We systematically restructured the course content to guide students through a transformation from mastering basic programming skills to practicing cutting-edge AI applications. The entire curriculum follows a "Foundational Skills-Applied Practice-Frontier Exploration" three-stage progressive structure.

3.1. Core Philosophy: From "Learning Python Syntax" to "Applying Python to Solve AI Problems"

The core philosophy of this curriculum restructuring represents a fundamental shift: the goal of the course is no longer for students to simply memorize and master Python's syntax rules, but to cultivate their ability to use Python as a powerful tool to analyze, understand, and solve complex, real-world engineering problems, particularly those related to AI. This problem-oriented and application-driven philosophy decides the entire curriculum design. We no longer ask, "What is a class in Python"? but rather, "When building a complex system, why do we need classes to organize our code? How do they help us manage state and behavior"? This shift from "what" to "why" and "how" is key to fostering students' higher-order engineering thinking.

3.2. Stage 1: Foundational Skills-Collaborative Learning of Core Python with AI

In the first stage, the curriculum covers Python's core fundamentals, including data types, control flow, functions, object-oriented programming, and modular programming. The innovation in this stage lies not in the content itself, but in the transformation of the learning method. We encouraged students to treat the AI large model as a knowledgeable dialogue partner, building a solid and flexible knowledge base through deep and skillful questioning.

AI Empowerment Points: Students transition from passive listening and note-taking to active, inquiry-based dialogue with the AI.

Scenario 1: Understanding Abstract Concepts. For a concept like "decorators", which is often difficult for beginners, a student could ask the AI: "Please explain Python's 'decorators' using a real-life analogy (e.g., renovating a house). Provide a comparative code example with and without the decorator, and summarize the main problems decorators solve (e.g., code reuse, logic decoupling)".

Scenario 2: Differentiating Confusing Concepts. To understand the difference

between the list. append ( ) and list. extend ( ) methods, a student could ask the AI: "Provide two specific code examples showing the different results of append ( ) and extend ( ) when adding another list to a list, and explain the difference in their underlying memory operations". Through this collaborative learning approach with AI, students not only master the knowledge itself but, more importantly, learn how to clarify ambiguous concepts through precise questioning, thereby internalizing the core skills of autonomous learning and problem decomposition.

3.3. Stage 2: Applied Practice—Modular Project Clusters

After mastering the Python basics, students enter the second stage. The philosophy here is to offer a "modular project cluster" that is multiple parallel projects of moderate difficulty from different domains. Students can choose one or two projects based on their interests and future career paths. This not only stimulates their motivation but also allows them to experience Python's powerful capabilities in various engineering fields.

3.3.1. Data Analysis and Visualization

Project: Based on the Iris dataset, use Pandas for data cleaning and feature engineering, build a logistic regression or decision tree classification model with Scikit-Learn, and evaluate the model's performance.

AI Empowerment Point: After training a model and obtaining a confusion matrix, students often struggle to interpret its business implications. They can ask the AI: "My model's confusion matrix on the test set is [[100, 10], [25, 50]]. Please help me calculate and explain the Precision, Recall, and F1-score. Specifically, please analyze how the feature dimensions of the samples might affect the recall and precision".

3.3.2. Data Scraping

Project: Write a web scraper using the Requests and Beautiful Soup libraries to crawl the top 250 movies from Douban, extracting information such as title, rating, director, and synopsis, and save it to a CSV file.

AI Empowerment Point: Web scraping often encounters anti-scraping measures. When a student's IP is blocked, they can seek help from the AI: "My scraper is getting a 403 error after frequent requests; I suspect it's an anti-scraping measure. Please analyze the possible reasons and provide at least three common bypass strategies (e.g., changing User-Agent, using proxy IPs, setting request delays), along with a brief analysis of their pros, cons, and suitable scenarios".

3.3.3. Web Application Development

Project: Use the lightweight Flask framework to design a simple student information management web application, implementing CRUD (Create, Read, Update, Delete) functionality for student records and interacting with an SQLite database.

AI Empowerment Point: After implementing basic functionality, to enhance students' engineering practices, they can be guided to ask the AI: "I have implemented a function to query student information via a URL path (e.g., /student/query? id = 101). Please explain the design principles of web APIs and help me refactor this functionality into an API endpoint that better adheres to these principles (e.g., using a path like /api/students/101 and the GET method)".

3.4. Stage 3: Frontier Exploration—Advanced AI Application Projects

The highest level of the curriculum is the frontier exploration stage. This stage focuses on the most prominent areas of AI technology, requiring students to complete a comprehensive, high-level project to gain in-depth experience, from using pre-trained models to performing model fine-tuning. This stage aims to elevate students from being users of AI technology to becoming AI developers and micro-innovators.

3.4.1. Computer Vision: Semantic Image Segmentation

Project Description: Utilize PyTorch and the torchvision library to load a pre-trained semantic segmentation model (e.g., DeepLabV3 or U-Net) and perform pixel-level segmentation on domain-specific images (e.g., urban streetscapes, medical images). The project emphasizes understanding the model's API, data preprocessing pipelines, loading pre-trained weights for inference, and visualizing the segmentation results.

AI Empowerment Points:

1) Debugging Support: "My input image tensor has a shape of [3, 512, 512], but the model throws an error saying it expects an input of [1, 3, 512, 512]. Please explain why this extra batch dimension is needed and show me how to fix it in PyTorch using the unsqueeze method".

2) Result Interpretation: After obtaining a multi-colored segmentation map, a student can ask the AI: "What do the different colors in this semantic segmentation result represent? How can I generate a legend for this map that associates the colors with specific classes (e.g., 'road', 'building', 'pedestrian') based on the model's class definitions"?

3.4.2. Natural Language Processing: Understanding and Fine-Tuning Large Language Models

Project Description: This project is a meta-application of using AI knowledge to create a smaller, customized AI. Students first need to understand the Transformer architecture and the training process of LLMs. They then select a small-scale open-source LLM (e.g., Qwen-1.8B, ChatGLM3-6B) and use a parameter-efficient fine-tuning method (e.g., from the Hugging Face ecosystem) on a small, domain-specific dataset (e.g., Python Q&A pairs from this course, summaries of specialized literature) to create a "course-specific mini-assistant".

AI Empowerment Points:

1) Environment and Resource Optimization: "I'm encountering a CUDA 'out of memory' error while fine-tuning the model. Please analyze the common causes (e.g., oversized batch size, high model precision) and provide a series of solutions, such as enabling gradient accumulation or using QLoRA for 4-bit quantized fine-tuning".

2) Core Principle Explanation: "Please explain the principle of LoRA fine-tuning with an easy-to-understand analogy. Why can it achieve results close to full-parameter fine-tuning by only training a small number of 'bypass' weights? How does this differ from traditional transfer learning"?

3) Result Interpretation: After fine-tuning, students design prompts to compare the model's responses before and after fine-tuning to assess whether it has acquired domain-specific knowledge and style.

3.5. Integrating Ethical and Societal Considerations: AI Value Alignment and Responsibility

Technological development must be guided by humanistic values. We integrated AI ethics and societal considerations throughout the curriculum to cultivate a sense of responsibility in students.

Embedded in Teaching: In the data analysis project, we use random data from NumPy without potential biases (e.g., gender, class) in the dataset and their impact on model fairness. In the AIGC project, we organized classroom debates on the potential misuse of Deepfake technology and corresponding governance strategies. In the LLM fine-tuning project, we emphasized the importance of Value Alignment and discussed how to prevent the fine-tuned model from generating harmful or irresponsible content.

A practical assignment example is as follows. Students were required to compare the outputs of different LLMs on the same query. They analyzed the tone, tendencies, and particularly the implicit biases within the responses. Then, they developed a series of related prompts to elicit answers from various models and then conducted a comprehensive analysis to identify their underlying attitudes. More importantly, a crucial aspect of the assignment was to address the issue of LLM hallucinations. Students were instructed to independently verify any factual claims made by the models, thereby preventing fabricated statements or non-existent citations.

4. Implementation and Performance Evaluation

To validate the effectiveness of the proposed reform framework, we conducted a full 8-week implementation during the first semester of the 2024-2025 academic year with a class of 93 junior students (Class of 2022) in the electronic information engineering major. The results were compared with a control group from a class of 80 junior students (Class of 2021).

The two groups had a similar programming foundation. First, the students were assigned to groups based on their original class divisions upon admission, ensur-

ing a random distribution without specific selection. This established homogeneity in terms of regional background, admission scores, and initial academic aptitude. Second, the students shared a similar programming foundation, as they had previously taken fundamental programming courses (e.g., C Language) with the same syllabus, textbook, teaching schedule, and instructor. Third, the course under study was taught by the same instructor to both groups, using the identical textbook and the same credit hours. Consequently, the pre-existing differences between the student groups were minimal, providing a reliable basis for comparison. This section outlines the implementation process and evaluates the reform's effectiveness from multiple dimensions.

4.1. Implementation Overview

Credit Hours: A total of 32 credit hours, comprising 24 hours of theory and practice and 8 hours of lab sessions. The content strictly followed the "three-stage progressive" structure: 4 weeks for foundational skills, 2 weeks for modular projects, and 2 weeks for the advanced AI project.

Hardware and Software Environment: Instruction took place in a lab equipped with JupyterLab and PyCharm Professional. For the computationally intensive Stage 3 projects, students had access to university lab servers equipped with NVIDIA GPUs, managed by a time-sharing scheduling policy. Some students also utilized third-party cloud platforms (e.g., AutoDL) as a supplement.

Projects and Assessment: Students worked in groups of 4-6. The final grade was based on a combination of continuous assessment (50%) and final evaluation (50%), broken down as follows: regular assignments and attendance (20%), modular project (30%), and final project (50%). The final project assessment included a detailed report, reproducible source code, and a group presentation.

4.2. Multi-Dimensional Performance Evaluation

We employed a mixed-methods approach, combining quantitative and qualitative analysis, to comprehensively evaluate the reform's effectiveness.

Quantitative Analysis includes the following parts：

1) Academic Performance: To evaluate the effectiveness of the proposed scheme, the scores from the practical section were analyzed. The data for the two groups are presented in Table 3. An independent samples t-test was conducted to examine the statistical significance. The result was a t-statistic of 2.228 and a p-value of 0.027. Since the p-value is less than 0.05, this indicates a significant difference between the two groups, demonstrating the notable effectiveness of our scheme.

Table 3. Statistical results of the two groups.

No	Name	Mean	Standard Deviation
1	Class of 2022	76.99	20.26
2	Class of 2021	70.75	16.55

2) Project Mastery: A post-course survey indicated a high level of mastery in advanced topics, with 77.1% of students reporting an understanding of image segmentation, 66.1% of image classification, and 52.5% of object detection. This demonstrates that the majority grasped key concepts in image processing.

3) Application Scope Awareness: Students identified the top five practical applications of Python as: web scraping and data acquisition (79.7%), image processing and recognition (75.4%), data mining and analysis (72.9%), building deep learning models (57.6%), and creating conversational chatbots (49.2%). These results show that the course and its projects provided students with a deeper understanding of Python's real-world problem-solving capabilities.

4) Learning Efficiency: The primary AI models used by students were Doubao, DeepSeek, and Qwen, with many using multiple models concurrently to compare outputs. According to a post-course survey, 100% of students reported that using AI large language models significantly reduced the time they were stuck on debugging and concept comprehension, with an estimated time saving of 2-3 hours per week on average.

Qualitative Analysis includes the following parts:

1) Skill Enhancement: Through course questionnaires and in-depth student interviews, we assessed self-perceived skill improvements. All students believed their ability to "efficiently solve unfamiliar problems using AI tools" had greatly improved. A vast majority stated that the course reform, especially the project-based practice, prompted them to think more deeply about Python's practical applications while also encouraging them to use AI more critically by analyzing the efficiency and correctness of generated code.

2) Learning Experience: Students showed a high level of interest and satisfaction with the new teaching pattern. In interviews, over 90% expressed satisfaction with personalized learning through AI and were highly engaged by the "three-stage progressive curriculum". The most popular topics of interest were machine learning classification, web scraping, and deep learning applications. The top three models that students found most interesting were CNN (ResNet), Transformer, and LSTM (RNN). Students felt the practical content was challenging and comprehensive, providing positive feedback such as "cutting-edge", "the projects were fun", and "we truly learned something useful".

4.3. Challenges and Reflections

Although the pedagogical reform achieved significant success, the implementation process also revealed several challenges:

1) Mitigating Cognitive Laziness: Continuously designing challenging, higher-order tasks is key to shifting students from "asking AI for answers" to "collaborating with AI to explore". We observed that if a task was not sufficiently complex, some students still tended to copy complete solutions from AI. Future instructional design must place a greater emphasis on assessing the "process" rather than just the "outcome".

Several measures were implemented to discourage plagiarism. First, during the initial class session, students were informed that using LLMs for assistance was permitted, but direct copy-pasting of solutions was strictly prohibited. Any violation would result in a failing grade for the assignment. Second, it was emphasized that all submitted code would undergo a plagiarism check, and any code identified as non-original would be flagged. Only original work would pass this check. Third, the course concluded with a project presentation and oral defense, designed to assess each student's genuine understanding. A lack of comprehension resulting from plagiarism would become evident during this stage and negatively impact their final grade. In summary, these multifaceted measures aimed to encourage the effective use of large language models as a learning tool while minimizing the submission of thoughtless, unoriginal work.

2) Equity in Computational Resources: While we provided GPU servers, resource contention remained an issue during peak project submission periods. Ensuring that every student has adequate and equal access to practical opportunities through optimized cloud solutions, more efficient fine-tuning techniques, and refined scheduling policies is an ongoing engineering challenge.

3) Faculty Development: The rapid evolution of AI technology places high demands on instructors' knowledge base and continuous learning capacity. Instructors need to be proficient not only in Python but also in AI models, fine-tuning, and prompt engineering. We have come to realize that establishing a faculty learning community and organizing regular technical training and pedagogical workshops are crucial for the sustainability of such reforms.

4) Deep Integration of AI Ethics: Currently, ethics education is primarily a baseline requirement. Elevating it from a superficial mention to a deeply internalized value system for students is a greater challenge. Future iterations could incorporate more real-world case studies on ethical dilemmas, allowing students to experience the consequences of technology misuse.

5. Conclusion and Future Work

5.1. Main Conclusions

In response to the new demands on engineering talent in the AI era, this study explored and implemented a systematic, AI large language model-driven pedagogical reform for the Python programming course in the electronic information engineering program. The key conclusions are as follows:

The proposed system framework—encompassing pedagogical model innovation, curriculum restructuring, and strategic implementation support—is both effective and feasible. By positioning AI large language models as "intelligent teaching assistants" for instructors and "personalized tutors" for students, we established a "human-AI collaborative" pedagogical paradigm that significantly improved teaching efficiency and personalized learning experiences. By restructuring the curriculum into a "three-stage progressive" model—"Foundational Skills", "Applied Practice", and "Frontier Exploration"—we successfully elevated the

learning objective from traditional syntax mastery to a higher level of applying Python to solve cutting-edge AI engineering problems.

The practical results demonstrate that this reform model achieved remarkable success in stimulating students' interest, enhancing their engineering practice and innovation skills, and fostering higher-order thinking and AI ethical awareness. It provides a valuable and practical example for the modernization of programming courses within the context of New Engineering Education.

5.2. Future Work

Based on the successes and reflections from this study, we propose the following directions for future work:

Curricular Expansion of the Reform Model: The framework validated in this study is highly transferable. Future efforts could focus on extending it to other core courses such as Data Structures, Machine Learning, and Digital Signal Processing. For instance, in Data Structures, AI could help students visualize abstract algorithmic processes (e.g., B+ tree insertion) with code and dynamic diagrams. In Machine Learning, AI could analyze the characteristics of different algorithms to help students deeply understand their applicable scenarios and limitations.

Development of a Domain-Specific Instructional Tool: Looking ahead, we can leverage open-source large models and fine-tune them on the accumulated resources from this course—including syllabi, lecture materials, outstanding project code, and high-quality student-AI interaction logs. The goal is to develop a domain-specific LLM assistant deeply integrated with our course's knowledge base, student learning profiles, and best-practice Q&A examples. Such a specialized assistant would understand our curriculum and our students better than any general-purpose model. When developing the instructional tool, potential challenges may arise from the acquisition of fine-tuning data and associated server costs. Regarding the data, publicly available datasets or published textbook materials can be utilized to mitigate privacy concerns. The primary cost of fine-tuning stems from GPU server expenses. In the context of a teaching reform project, grant funding can cover these server costs. Alternatively, when the model is developed as an in-house tool, the department's GPU servers can be leveraged to implement development.

Building an Integrated Industry-Academia-Research Ecosystem: The development of AI technology is boundless. We are committed to establishing a dynamically updated course project library that continuously tracks technological frontiers and rapidly translates the latest research findings (e.g., multimodal large models, AI Agents) into teaching cases. Concurrently, we will actively explore collaborations with industry partners to introduce real-world challenges into the curriculum, allowing students to engage with frontline engineering problems on campus. This will help build an integrated industry-academia-research ecosystem for talent cultivation, preparing innovative engineers who are truly competitive for the future.

Acknowledgements

The work is funded by the foundation of the Exploration and Practice of Programming Course Teaching Models for Electronic Information Engineering Major in the Context of New Engineering Education (QX2022M39).

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Denny, P., Prather, J., Becker, B.A., Finnie-Ansley, J., Hellas, A., Leinonen, J., *et al.* (2024) Computing Education in the Era of Generative AI. *Communications of the ACM*, **67**, 56-67. https://doi.org/10.1145/3624720

[2] Pereira, A.F. and Ferreira Mello, R. (2025) A Systematic Literature Review on Large Language Models Applications in Computer Programming Teaching Evaluation Process. *IEEE Access*, **13**, 113449-113460. https://doi.org/10.1109/access.2025.3584060

[3] Hazzan, O. and Erez, Y. (2024) Generative AI in Computer Science Education. P*roceedings of the* 55*th ACM Technical Symposium on Computer Science Education V.* 2, Portland, 20-23 March 2024, 1899. https://doi.org/10.1145/3626253.3633409

[4] Zhang, W.J. (2024) Design and Application of Programming Learning Feedback Strategies Based on Generative Artificial Intelligence. Ph.D. Thesis, Guangzhou University. https://doi.org/10.27040/d.cnki.ggzdu.2024.000127

[5] Haldar, S., Pierce, M. and Fernando Capretz, L. (2025) Exploring the Integration of Generative AI Tools in Software Testing Education: A Case Study on ChatGPT and Copilot for Preparatory Testing Artifacts in Postgraduate Learning. *IEEE Access*, **13**, 46070-46090. https://doi.org/10.1109/access.2025.3545882

[6] Gan, J.H. and Li, N. (2025) Copilot: Development and Challenges of Large Model Assisted Programming. *Computer and Network*, **51**, 149-155.

[7] Haindl, P. and Weinberger, G. (2024) Students' Experiences of Using ChatGPT in an Undergraduate Programming Course. *IEEE Access*, **12**, 43519-43529. https://doi.org/10.1109/access.2024.3380909

[8] Mogan, S.C., Mustaffa, Z., Sulaiman, M.H. and Ernawan, F. (2023) Product Recommendation Using Deep Learning in Computer Vision. 2023 *IEEE* 8*th International Conference on Software Engineering and Computer Systems* (*ICSECS*), Penang, 25-27 August 2023, 263-267. https://doi.org/10.1109/icsecs58457.2023.10256332

[9] Palla, D. and Slaby, A. (2025) Evaluation of Generative AI Models in Python Code Generation: A Comparative Study. *IEEE Access*, **13**, 65334-65347. https://doi.org/10.1109/access.2025.3560244

[10] Vemula, S. (2024) Enriching Python Programming Education with Generative AI: Leveraging Large Language Models for Personalized Support and Interactive Learning. 2024 *IEEE Frontiers in Education Conference* (*FIE*), Washington, 13-16 October 2024, 1-8. https://doi.org/10.1109/fie61694.2024.10893561

[11] Prather, J., Leinonen, J., Kiesler, N., Benario, J.G., Lau, S., MacNeil, S., *et al.* (2024) How Instructors Incorporate Generative AI into Teaching Computing. *Proceedings of the* 2024 *on Innovation and Technology in Computer Science Education V.* 2, Milan, 8-10 July 2024, 771-772. https://doi.org/10.1145/3649405.3659534

[12] Newby, N. (2024) Askers, Answerers, Non-Answerers, and Lurkers: Investigating

Online Q&A Platforms Personas and Success in a CS1 Course. *Proceedings of the* 55*th ACM Technical Symposium on Computer Science Education V.* 2, Portland, 20-23 March 2024, 1886. https://doi.org/10.1145/3626253.3635413

[13] Snider, J.M. (2024) Edit, Run, Error, Repeat: Learning Analytics to Find the Most Improved Programming Student. *Proceedings of the* 55*th ACM Technical Symposium on Computer Science Education V.* 2, Portland, 20-23 March 2024, 1888. https://doi.org/10.1145/3626253.3635423