Journal of Education and Educational Research
ISSN: 2957-9465 | Vol. 19, No. 3, 2026

Research on the Teaching Reform Path of Python Courses for Mechanical Majors

Li Wang

School of Chengdu Jincheng University, Chengdu, Sichuan, 61000, China

Abstract: Driven by the rapid advancement of intelligent manufacturing technologies, Python has gained broader application scenarios within mechanical engineering disciplines. Nevertheless, existing Python courses tailored for mechanical majors still face multiple teaching drawbacks, including ambiguous curriculum positioning, disconnection between teaching materials and industry specialties, as well as monotonous teaching modes. Combined with project-based learning and integrated teaching concepts, this paper puts forward a targeted reform scheme for Python courses in mechanical majors. The reform covers four key dimensions: clarifying curriculum orientation, reorganizing teaching content, innovating teaching approaches and optimizing assessment systems. Practical teaching cases are also provided to verify the actual effect of the proposed reforms. After implementing the above adjustments, the updated teaching framework can effectively stimulate students' learning initiative and strengthen their practical programming capabilities. This research delivers solid support for cultivating mechanical engineering talents who can meet the talent demands under the intelligent manufacturing industry background.

Keywords: Intelligent Manufacturing; Mechanical Majors; Python Course; Teaching Reform; Project-Based Teaching.

1. Introduction

Intelligent manufacturing has become a core direction of transformation and upgrading in the modern manufacturing industry, covering intelligent design, intelligent production, intelligent management and other key technical fields. In this context, programming capability has evolved into an essential core competency for students majoring in mechanical engineering. Featuring concise syntax and abundant functional libraries with wide application prospects , Python has emerged as the preferred programming language for data analysis and automation tool development in mechanical engineering[1]. According to the TIOBE Programming Community Index released in May 2026, Python ranks first with a popularity rating of 19.98%, approximately 8.4 percentage points higher than C++ in second place, further consolidating its dominant position in engineering application scenarios.

2. Analysis of Current Problems in Python Course Teaching for Mechanical Majors

2.1. Unclear Course Positioning

Currently, there are two tendencies in the positioning of Python courses for mechanical majors. One is to regard it as a general course, focusing on the explanation of programming syntax and basic concepts, and has little connection with mechanical engineering professional knowledge. The other is to take it as a professional elective course with less class hours and more content, which should not only explain the basic content of grammar, but also explain the application combined with practice, ignoring the actual needs and knowledge base of mechanical engineering students [2]. These two positioning methods fail to accurately grasp the learning characteristics and career development needs of mechanical students .

2.2. Disconnection Between Teaching Content and Professional Requirements

Mechanical students usually have a strong engineering background, but relatively difficult to understand the abstract concepts of programming languages. Most of the current Python course teaching content uses computer professional textbooks, focusing on the core content of computer science such as data structure and algorithm design, while the application of data analysis, engineering calculation, automatic control and other applications related to mechanical engineering are less involved. This content setting makes it difficult for students to combine programming knowledge with professional learning, and the learning motivation is insufficient.

2.3. Singular Teaching Methods

Mechanical students are more inclined to intuitive and practical learning. However, the current Python course teaching is still mainly based on theoretical explanation and grammar demonstration, and lacks practical teaching links combined with practical problems in mechanical engineering. Students passively accept knowledge in class, lack opportunities to actively explore and practice, and the improvement of programming ability is limited.

2.4. Imperfect Assessment Mechanism

The traditional assessment method is based on theoretical examination, which mainly examines students' memory and understanding of programming grammar, but lacks of examination of students' actual programming ability [3]. This assessment method leads students to pay more attention to the memory of theoretical knowledge, but ignore the training of programming practice, and it is difficult to truly master the ability to solve practical problems.

3. Teaching Reform Path for Python Courses of Mechanical Majors

3.1. Clarify Course Positioning

The positioning of Python courses for mechanical majors should revolve around the core concept of "serving mechanical engineering." The course objective should be positioned to cultivate students' ability to use Python language to solve practical mechanical engineering problems, rather than training professional programmers. Specifically, the course should enable students to master the basic syntax and core concepts of Python programming, be able to use Python for engineering calculations, data analysis, automation script writing, etc., laying the foundation for subsequent professional courses such as intelligent design and intelligent manufacturing.

3.2. Reconstruct Teaching Content System

Based on the characteristics of the mechanical engineering major and the development needs of intelligent manufacturing, the teaching content of the Python course has been reorganized into three modules: the basic module, the application module, and the comprehensive module.

The basic module mainly covers the core concepts of Python, such as basic syntax, data types, control structures, and functions. It lays the foundation for subsequent learning. Unlike computer courses, the explanations in the basic module are more closely integrated with practical cases in mechanical engineering, such as using variables and expressions to describe the geometric parameters of mechanical parts, and using loop structures to calculate the motion parameters of mechanical systems, thereby connecting abstract programming concepts with specific engineering problems.

The application module focuses on Python libraries and tools related to mechanical engineering, such as NumPy for numerical calculations, Matplotlib for data visualization, Pandas for data processing, and PyAutoGUI for automated operations. The teaching content of this module should be closely integrated with typical application scenarios in mechanical engineering, such as using NumPy to calculate the stress of mechanical parts, using Matplotlib to draw displacement curves of mechanical parts, using Pandas to process experimental data, and using PyAutoGUI to automate the operation of CAD software.

The comprehensive module enables students to integrate the knowledge they have learned and solve complex mechanical engineering problems through project practice. The project topics should be based on actual problems in the field of mechanical engineering, such as motion simulation of mechanical systems, analysis and processing of experimental data, and development of automation tools. Through project exercises, students can deeply understand the practical value of programming in mechanical engineering and enhance their ability to solve practical problems.

3.3. Innovate Teaching Methods

Project-based teaching and integrated teaching are the core methods of this teaching reform. Project-based teaching uses actual projects as the carrier, integrating programming knowledge with professional practice, enabling students to master programming skills during the process of completing the projects. While integrated teaching focuses on the integration of interdisciplinary knowledge, organically combining programming knowledge with mechanical engineering professional knowledge to cultivate students' comprehensive abilities.

Specifically, the teaching methods can take the following forms:

(1) Case-based teaching: Select typical cases related to mechanical engineering, such as the analysis of mechanical systems, the processing and analysis of experimental data, and the development of automation tools. By analyzing the requirements in the cases, programming knowledge is introduced. Students can learn programming based on their understanding of the actual requirements, making the learning goals more clear and the learning motivation more sufficient.

(2) Flipped classroom model: Basic grammar and concepts are learned before class, and students can conduct self-study through micro-lessons and online resources [4]; classroom time is mainly used for case analysis and project practice, with the teacher mainly responsible for guidance and answering questions. This model can improve the efficiency of classroom time utilization and increase the opportunities for students to participate in practical activities.

(3) Group cooperative learning: Students are divided into groups to complete projects related to mechanical engineering. Each group is responsible for a specific task. Group members work collaboratively to complete tasks such as requirement analysis, design, coding, and testing of the project. Through group cooperation, students can learn teamwork skills, improve communication skills, and enhance project management abilities.

(4) Online and offline integration: Utilize online learning platforms and tools, such as programming practice platforms, code sharing tools, and online debugging tools, to support students' after-class learning and practice. Teachers can assign homework and provide feedback through online platforms, while students can submit code online and participate in discussions, achieving an effective integration of online and offline learning.

3.4. Improve Assessment Mechanism

The evaluation mechanism should change from a single theoretical assessment to a diversified comprehensive assessment. Specifically, the evaluation can be divided into two parts: process evaluation and outcome evaluation.

The process evaluation mainly examines the learning process and practical performance of students, including classroom participation, homework completion, and group project progress. Process assessment can urge students to continue learning and find and solve learning problems in time.

Performance assessment mainly examines students' academic performance and practical ability, including final projects, programming tests, and innovation results. The final project requires students to work independently or in groups on a comprehensive project related to mechanical engineering, the programming test examines students' programming fundamentals and problem-solving skills, and the innovation achievement encourages students to use their knowledge to develop innovative solutions.

The diversified assessment mechanism can comprehensively evaluate students 'learning effect, investigate theoretical knowledge and practical ability, and guide students 'all-round development.

4. Teaching Case Design

Taking the major of Mechanical Design, Manufacturing and Automation as an example, the course is offered in the first semester of the junior year with a total of 24 class hours.

4.1. Division of Teaching Content

Based on the aforementioned teaching reform path, the course content is organized according to three modules: foundation module, application module, and comprehensive module.

Foundation module (8 class hours) covers Python basic syntax, data types, control structures, functions, and other core concepts. Teaching uses cases related to mechanical engineering, such as calculating the volume and mass of mechanical parts, analyzing the motion state of simple mechanical systems, etc., making abstract programming concepts connected to engineering reality.

Application module (8 class hours) focuses on Python libraries and tools related to mechanical engineering. Teaching adopts a case-driven method, with each case originating from actual applications in mechanical engineering. For example, when explaining NumPy, use the case of stress calculation of mechanical parts; when explaining Matplotlib, use the case of plotting displacement curves of mechanical parts; when explaining Pandas, use the case of experimental data processing and analysis.

Comprehensive module (8 class hours) centers on project practice, where students need to complete a comprehensive mechanical engineering project. Project topics originate from actual problems in the field of mechanical engineering, such as motion simulation of mechanical systems, analysis and processing of experimental data, development of automation tools, etc. Students complete projects in groups, with 5-6 people per group, working collaboratively to complete requirements analysis, design, coding, and testing.

4.2. Teaching Implementation

Teaching implementation is divided into three stages: foundation learning stage, application expansion stage, and comprehensive practice stage.

Foundation learning stage: The main objective of this stage is to consolidate students' Python programming foundation, cultivate basic programming thinking, and master essential basic programming skills. The teaching content is mainly based on the Python foundation module, covering core knowledge points such as variables and data types, operators, conditional judgment, loop structures, common data structures, custom functions, and basic file operations. Teaching adopts a combination of online and offline methods. Teachers rely on resources such as micro-lectures and online learning materials to guide students to independently learn basic concepts and syntax knowledge before class[5]. Classroom sessions omit redundant repetitive elaboration of elementary theoretical knowledge. Instructional emphasis shifts to the dissection of fundamental programming cases, paired with in-session coding drills and on-site debugging exercises, aiming to pinpoint and resolve difficulties emerging from students' pre-class self-study. Layered, progressive coding training enables learners to acquire proficient capabilities including standardized code specification, elementary program logic construction, lightweight program development and error troubleshooting. Students can independently complete programming tasks such as numerical computation, text processing and basic statistical analysis, laying a solid foundation for follow-up application-oriented learning combined with mechanical engineering expertise.

The application expansion stage is carried out on the basis of mastered fundamental programming skills. This stage focuses on Python applications for mechanical engineering, so as to cultivate students' practical ability to address basic professional problems with programming tools. The teaching core lies in hands-on training of three widely adopted Python libraries for engineering, namely NumPy, Matplotlib and Pandas. Teaching contents are designed to match practical demands of mechanical engineering data processing, data analysis and data visualization. Teachers deliver case teaching centering on authentic mechanical engineering scenarios, including dimension statistics of mechanical components, data analysis of material mechanical properties, processing of equipment operating parameters, and plotting of engineering data curves. Through case interpretation, code demonstration and step-by-step decomposition of problem-solving approaches, students can master practical workflows including engineering data reading, data cleaning, numerical calculation and result visualization. By reproducing given cases, adjusting parameters, conducting independent debugging and finishing extended exercises, students gradually master practical skills such as batch engineering data computation, professional chart plotting and programmable processing of routine engineering problems. This breaks the pure theoretical learning mode and realizes the preliminary integration of Python programming techniques and mechanical engineering knowledge.

The comprehensive practice stage aims to integrate knowledge and skills acquired in previous stages, advance the in-depth combination of Python programming and mechanical engineering, and improve students' comprehensive capacity to tackle practical engineering problems. Group cooperative learning is adopted in this stage, where students are divided into teams of five to six members to finish comprehensive programming projects related to mechanical engineering. Two types of project topics are available: topics formulated by teachers according to key teaching points, and self-designed topics proposed by students based on their professional interests. All topics must be reviewed by instructors to verify feasibility before implementation. A typical comprehensive practice project arranged in this course is the calculation, data analysis and visualization of mechanical transmission parameters based on Python. Closely linked to core knowledge of mechanical design, this project takes common mechanical transmission structures such as gears and belt drives as research objects. Students are required to independently finish the full engineering workflow of the project: they utilize Pandas to classify and import raw operational data including transmission power, rotational speed and transmission ratio, then embed professional mechanical transmission formulas into Python scripts to automatically compute and validate key design indicators, such as gear modulus, tooth count, pulley diameter, center distance and transmission efficiency. Meanwhile, Matplotlib is deployed to generate speed-torque characteristic curves, efficiency fluctuation graphs and comparative charts of structural parameters, which render computational design outcomes in an intuitive visual format and realize seamless linkage of data collation, parametric calculation and result visualization.

Throughout project development, instructors schedule periodic group progress briefings and in-class sharing workshops, offering individualized technical consultations for encountered obstacles. Timely corrections are provided for flawed coding logic, misapplied mechanical formulas and inappropriate data analysis workflows, alongside targeted guidance to optimize program architecture and overall project schemes. Each learning team submits complete program source files, technical project reports and presentation slides for final course evaluation. Full-cycle engineering project training allows students to flexibly combine coding techniques and mechanical expertise, bridging the gap between theoretical learning and industrial practice. Meanwhile, learners cultivate teamwork awareness, fault diagnosis skills, engineering practice capabilities and innovative thinking, which collectively advance their overall disciplinary literacy[6].

5. Conclusion and Future Work

5.1. Conclusion

At present, Python courses for mechanical engineering undergraduates in Chinese universities generally suffer from multiple prevalent teaching deficiencies. Typical issues include ambiguous curriculum positioning, disconnection between teaching content and industrial practical demands, single and rigid classroom teaching modes, as well as simplistic and unitary assessment mechanisms. Based on the pedagogical concepts of project-driven learning and interdisciplinary integrated teaching, this study proposes a targeted curriculum reform scheme that adapts to the talent training orientation of mechanical engineering majors. Combined with the practical implementation process and classroom teaching feedback of the reform practice, four core research conclusions are summarized in this paper.

First, clarified and refined curriculum positioning is the prerequisite for effective Python teaching reform. Unlike Python teaching for computer majors, which aims to cultivate professional software development talents, the teaching orientation for mechanical engineering undergraduates possesses distinct disciplinary characteristics. The curriculum design should be closely oriented toward the professional training objectives and actual engineering application needs of mechanical engineering. The core teaching goal is to equip students with the capability of solving practical mechanical engineering problems by means of programming tools, so as to make up for the deficiency of traditional teaching that overemphasizes theoretical programming knowledge and deviates from the professional talent training demands of mechanical majors.

Second, the systematic optimization and hierarchical reorganization of teaching content constitute the core foundation for the implementation of curriculum reform. Traditional Python teaching usually adopts a single knowledge arrangement mode centered on grammar indoctrination, which fails to match the cognitive rules of mechanical students. In view of the professional knowledge reserve and learning characteristics of mechanical engineering undergraduates, this research constructs a progressive teaching content system from basic to advanced levels. The whole learning system covers three core stages: consolidation of basic programming syntax proficiency, skilled application of engineering-oriented Python tool libraries, and comprehensive solution of complex engineering problems. This hierarchical content framework conforms to the learning logic of mechanical students, effectively enhancing the professionalism, pertinence and practicality of the course teaching system.

Third, the diversified and flexible application of multi-teaching methods is the key to improving teaching quality and ensuring reform effectiveness. Relying on the integration of project-driven teaching and interdisciplinary teaching ideas, this reform breaks through the limitations of traditional cramming teaching modes. It realizes the deep integration of abstract programming theories and typical mechanical engineering scenarios, including mechanical component parameter calculation, experimental data mining and analysis, mechanical system kinematic simulation, and secondary development and automatic operation of engineering design software. Diversified teaching means such as case-based teaching, flipped classroom, team collaborative learning and online-offline hybrid teaching are comprehensively applied to stimulate students' autonomous learning initiative. In the whole-process practical project training, students can internalize professional knowledge, polish practical operation skills, and ultimately achieve comprehensive improvement in programming practice ability and engineering problem-solving thinking.

Fourth, the establishment of a multi-dimensional comprehensive evaluation system provides a strong guarantee for the long-term effectiveness of teaching reform. The traditional single final written examination is unable to objectively and comprehensively reflect students' real learning status and comprehensive ability. This study constructs a holistic evaluation system that organically combines formative process assessment and summative terminal assessment, realizing full-cycle and multi-angle evaluation of students' learning performance.The system takes daily learning attitude, in-class practical performance, group collaboration competence and final comprehensive operation ability into account, so as to assess students' learning performance objectively from multiple perspectives and promote the all-round improvement of their professional capacity and comprehensive literacy.

Practical teaching results prove that this revised teaching framework effectively addresses various deficiencies of existing Python courses for mechanical majors. It greatly stimulates students' willingness to learn independently and improves their practical coding ability and engineering problem-solving capacity. The proposed reform scheme bridges the gap between programming technology and mechanical engineering disciplines, complies with the training standards for compound practice-oriented mechanical talents required by modern manufacturing, and delivers practical references for universities to optimize Python courses and innovate talent cultivation modes for mechanical majors.

5.2. Limitations and Future Work

This study has several limitations. First, the sample size of teaching practice is limited, as the reform was carried out only for one grade of a single major in one university; hence the generalizability of research findings needs further verification. Second, the effectiveness evaluation of teaching reform relies mainly on short-term data without sufficient long-term follow-up investigation. In addition, the development of teaching resources and the training of teaching staff remain critical challenges for the reform.

Further research can be conducted in four directions. First, the teaching practice scale will be expanded by promoting the proposed reform scheme in more universities and mechanical-related majors, so as to validate the universal applicability and practical effectiveness of the research outcomes. Second, long-term follow-up investigations will be carried out on participating students to explore the sustained influence of this teaching reform on their professional growth and career development. Third, discipline-specific teaching resources will be further supplemented and optimized through constructing Python case bases and engineering project databases that adapt to the training characteristics of mechanical majors. Fourth, targeted faculty development activities will be organized to enhance teachers' interdisciplinary teaching ability and improve the overall quality of integrated teaching.

Against the background of rapid development in intelligent manufacturing, mechanical engineering students are required to possess solid programming competency. The specialized Python teaching reform for mechanical majors is therefore a sustained and progressive work. To satisfy the updated talent training demands of the manufacturing industry, continuous exploration and iterative optimization of teaching modes are essential. Such efforts can provide solid talent support for the intelligent transformation and high-quality upgrading of modern manufacturing industries.

References

[1] Xie, R. S. (2025). Teaching exploration of Python programming course based on generative artificial intelligence technology. *Computer Knowledge and Technology*, 21(30), 166–168.

[2] Li, D. H. (2025). Research on teaching reform path of Python courses in non-computer science majors in colleges and universities. *Henan Journal of Finance and Economics*, 39(6), 92–96.

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

[4] Guo, S., Li, G. H., & Wang, X. Y. (2023). Exploration and practice of diversified teaching models for Python language courses oriented toward non-computer science majors. *Modern Educational Technology*, 33(8), 78–82.

[5] Wang, J. F., Li, Y. Q., & Sun, Y. C. (2025). Teaching reform of Python programming for non-computer science majors. *Fujian Computer*, 41(10), 102–106.

[6] Sui, X. L. (2025). Exploration and practice of "Python Programming" course teaching model based on OBE concept. *Wireless Internet Technology*, 22(23), 87–91.