ISSN 2522-6398 Vol. 9, Issue 5: 222-230, DOI: 10.25236/FER.2026.090530

Exploration and Practice of Teaching Reform in Practical Courses for Big Data Majors Based on Problem-Based Learning—Taking the Course "Python Programming" as an Example

Zhou Ruan[1], Na Xie[1], Chengqiang Wang[1,a,*], Zhenfen Dong[1]

[1]*School of Mathematics and Physics, Suqian University, Suqian, 223800, Jiangsu Province, China*
[a]*chengqiangwang2022@foxmail.com*
**Corresponding author*

Abstract: *With the development of artificial intelligence, the market demand for big data talents with both theoretical and practical competence has surged. Therefore, students' practical competence is crucial for the success of their programming learning. Currently, students majoring in big data face problems such as disconnection between skills and application scenarios, insufficient self-driven ability, and rationality deficits in AI application awareness, which restrict the improvement of their programming practical competence. Based on the Problem-Based Learning (PBL) theory, this study proposes four core strategies: first, enrich the diversity of teaching scenarios to guide students in exploring the essence of code and architectural design ideas, thereby consolidating the foundation of programming; second, implement a research-based learning model with the role transformation of dominant participants, enabling students to understand the underlying logic of code from multiple perspectives; third, construct the rational application of AI to cultivate students' critical thinking and reconstruction abilities regarding AI-generated code; fourth, encourage the openness of teaching materials and knowledge integration to facilitate students' autonomous learning and knowledge combing. Meanwhile, curriculum improvement strategies are put forward, allowing students to solve programming problems through multiple methods and introducing challenging real questions from competitions such as the Blue Bridge Cup. Practice has shown that these strategies can effectively stimulate students' sense of exploration, promote their transformation from passive learning to active knowledge construction, and provide references for programming teaching and innovative talent cultivation in big data majors.*

Keywords: *Talent Cultivation of Big Data Majors, Practical Competence, Teaching Strategies, Programming Learning*

1. Introduction

In recent years, with the rapid development of artificial intelligence technology, the market demand for big data talents with both theoretical literacy and practical competence has continued to rise. Among them, Python programming, as a core tool, its teaching quality directly affects the overall level of talent cultivation in big data majors. Therefore, improving the practical Python programming competence of students majoring in big data has become an important direction of current higher education curriculum reform. Combining the current situation of domestic education, Wu, G. [1] proposed the reference significance of the PBL (Problem-Based Learning) model for China's higher education reform. This learning model oriented by real problems can significantly stimulate students' willingness to take the initiative in inquiry and make up for the limitations of traditional teaching dominated by theoretical lectures. Consequently, we will explore the reform of practical programming courses for big data majors based on the PBL concept.

The limitations encountered by college students majoring in big data when delving into programming problems are also universal, as Wang, Q. [2] noted regarding the difficulties faced by beginners in programming. These limitations prevent students from integrating basic knowledge coherently, making them prone to fragmented and superficial learning predicaments when confronting complex programming tasks. In response to these issues, relevant studies have proposed solutions. Lu et al. [3] conducting PBL teaching research with the Python course as the practical carrier, put forward

ISSN 2522-6398 Vol. 9, Issue 5: 222-230, DOI: 10.25236/FER.2026.090530

the core idea of "deeply binding project design with real scenarios" and the concept that "technical assistance should serve thinking cultivation." They emphasized that in PBL teaching, "the use of tools should center on the core of problem inquiry and avoid becoming mechanical operations." The "problem-oriented + action learning" model proposed by Liu, M., & Zeng, X. [4] pointed out the need to break the traditional "teacher lecture - student passive acceptance" model and verified the rationality of reconstructing the roles of teachers and students. Dai, B. [5] indicated in his research that in PBL teaching of the Python course, it is necessary to lay the foundation for students' independent inquiry by "publicizing courseware, case codes, and extended task lists in advance." He particularly emphasized the importance of "guiding students to systematically summarize scattered programming knowledge, such as organizing code modules under different scenarios and summarizing algorithm optimization ideas." Collectively, these studies suggest that teaching reform centered on the PBL concept is a key path to addressing the pain points in cultivating practical competence in big data professional courses, and it requires the coordinated implementation of multi-dimensional strategies including "scenario connection, role reconstruction, and resource support."

In response to these insights, Dai, M.Y. [6] argued that in PBL teaching of the Python course, it is necessary to break the traditional teaching limitation of "a single standard answer." Through the design of "guiding students to explore multiple implementation paths for the same programming problem," students' divergent thinking and code innovation capabilities can be stimulated. Building on this, this study focuses on exploring the core carrier for realizing the transformation from "passive learning" to "active creation." Specifically, taking the course "Python Programming" as an example, it aims to enhance college students' subjective initiative and improve their programming competence.

The rest of this paper is organized as follows. In Section 2, we elaborate on the limitations in improving college students' practical competence. In Section 3, we propose several strategies based on the PBL theory to enhance the practical competence of college students majoring in big data, aiming to foster their creative thinking. In Section 4, with the help of real programming questions from the Blue Bridge Cup, we propose curriculum improvement strategies through case analysis and decomposition. In Section 5, we present the conclusions.

2. Limitations in Enhancing College Students' Practical Competence

In this section, our primary purpose is to illustrate that the limitations in improving college students' practical competence stem from mechanized practical opportunities, weak willingness to practice, and unbalanced AI application. We further explain that these three issues are fundamentally rooted in the disconnection between skills and application scenarios, insufficient self-driven learning ability, and deficiencies in the rationality of AI application awareness. Most students can only master grammatical rules, basic algorithms, and problem-solving methods through courses, but they lack considerations for code readability and scalability in real-world contexts. When confronted with practical project architectures, they are prone to falling into fragmented coding dilemmas. Meanwhile, college students often focus on the code implementation of individual problems, lacking the decomposition and exploration of the essence of basic programming. Especially when debugging complex code, they are easily distracted by superficial errors and struggle to identify flaws in the underlying logical design.

2.1 Disconnection between Skills and Application Scenarios

One of the core limitations hindering the improvement of college students' programming practical competence lies in mechanized practical opportunities, which manifests as a disconnection between skill acquisition and real-world application scenarios. A direct reflection of this disconnection is the low frequency of using programming tools in daily life. Current programming teaching mostly centers on theoretical lectures, with course cases taking basic code as the core carrier. However, these cases only focus on solving single, isolated basic problems, confining teaching objectives to superficial and simple tasks that lack relevance to the actual needs of students' daily study and life.

Most students neither develop the awareness of actively using programming to solve problems nor have the practical opportunity to transform classroom learning into practical tools in their daily scenarios outside class. This mismatched state prevents students from consolidating programming skills through specific contextualized practice and hinders their in-depth understanding of the mapping relationship between code logic and real-world problems. Consequently, it forms a competence dilemma, ultimately restricting the transformation of programming practical competence toward the core goal of addressing complex problems.

ISSN 2522-6398 Vol. 9, Issue 5: 222-230, DOI: 10.25236/FER.2026.090530

2.2 Insufficient Self-Driven Learning Ability of College Students

At the level of learning subjects, a key limitation affecting the improvement of college students' programming practical competence lies in their weak willingness to practice during programming learning. This is mainly manifested as a superficial understanding of basic code and a tendency toward mechanical application. In the process of programming learning, most students fail to explore the essential logic of basic code with an inquiry-based mindset. They neither take the initiative to analyze the adaptation principles of basic code in solving complex problems nor systematically sort out the methodology for the combined application of basic code modules.

This passive learning state prevents students from establishing logical connections between basic code and complex problems. As a result, they can neither conduct connective combinations of a single basic code module with other code logics nor flexibly call basic code to construct solutions in complex programming tasks. Ultimately, this restricts their ability to conduct in-depth deconstruction and breakthroughs in the face of complex programming problems.

2.3 Deficiencies in the Rationality of AI Application Awareness

From the perspective of the alienation of technical tool application, a limitation hindering the improvement of college students' programming practical competence is also reflected in unbalanced AI practice. This is manifested as an imbalance in the awareness of using AI programming tools and excessive dependence on them, which in turn leads to a serious lack of in-depth understanding of the structural functions of complex code.

In current programming learning scenarios, most students simplify AI tools into code generators rather than taking the initiative to use them as auxiliary carriers for inquiry, lacking a rational positioning of tool application. When faced with the demand for complex code involving multi-layered architectures, multi-module coupling, and cross-scenario adaptation, students generally directly input task descriptions to obtain AI-generated results. They do not participate in the core thinking processes such as requirement decomposition, module division, and logical connection, nor do they conduct reverse tracing of the generated complex code.

This forms a competence paradox: students can rely on AI to generate code but struggle to deconstruct code without AI support. It greatly weakens the reconstructive value of complex code learning for programming thinking and seriously restricts their ability to conduct in-depth exploration and independent breakthroughs in the face of complex programming problems.

3. Strategies for Enhancing Practical Competence of Big Data Majors Based on PBL Theory

In this section, we will discuss strategies for improving the practical competence of college students majoring in big data, based on Problem-Based Learning (PBL)—a core concept of the PBL theory. These strategies are specifically designed to equip students with the abilities of independent problem-solving and innovative thinking.

The specific methods include enriching the diversity of teaching scenarios to expand the scope of practical application, implementing a research-based learning model with the transformation of dominant roles to stimulate practical inquiry, rationally utilizing AI tools while fostering critical thinking and reconstruction capabilities, and encouraging the openness of teaching materials and students' independent knowledge integration.

These strategies help bridge the gap between programming theory and engineering practice. By promoting active learning, critical thinking, and practical application, they effectively stimulate students' internal motivation for learning and lay a solid competence foundation for them to address complex big data challenges.

3.1 Enriching the Diversity of Teaching Scenarios

Enriching the diversity of teaching scenarios is one of the core strategies for enhancing the practical competence of students majoring in big data.

The core value of this strategy lies in realizing the starting point of "real or complex problems" as advocated in the PBL concept. It extends programming learning from closed classrooms to open

ISSN 2522-6398 Vol. 9, Issue 5: 222-230, DOI: 10.25236/FER.2026.090530

real-world problems, thereby improving the flexibility of programming practice. Specifically, by introducing diverse cases closely related to students' daily study, life, and even future careers, the diversity of teaching scenarios connects abstract code logic with vivid practical needs. Taking the Python Programming course as an example, in terms of data acquisition, teachers can guide students to write crawler scripts to automatically retrieve campus notices, academic forum hot topics, or commodity prices on e-commerce platforms. Linking classroom knowledge with frequently used real-world software not only stimulates students' internal motivation to "explore and solve problems through programming" but also provides high-frequency practical opportunities to transform classroom learning into practical tools.

Its significance lies in breaking the barrier between basic code and complex real-world problems in traditional teaching. In the process of solving real, comprehensive problems close to their own lives, students can gain a deeper understanding of the value of technology and systematically consolidate the skills they have learned, thereby effectively promoting the transformation of programming practical competence toward the core goal of solving complex problems.

In summary, promoting the diversity of teaching scenarios is not a simple formal reform but a return to the essence of programming education. By constructing a highly relevant and high-frequency practical environment, it builds a bridge for students from theoretical cognition to practical exploration, and is an indispensable path for cultivating the core competitiveness of big data majors.

3.2 Research-Based Learning with Role Transformation of Dominant Participants

Research-based learning with role transformation of dominant participants is an innovative strategy for enhancing the practical competence of students majoring in big data.

The core value of this approach lies in implementing the key steps of the Problem-Based Learning (PBL) concept. After teachers raises questions, the dominant role shifts from teachers to students. Students explore and collaborate to develop solutions to problems, with guidance from teachers, and present their approaches—this process enhances college students' willingness to practice programming. Specifically, in classrooms, teachers are responsible for decomposing complex programming problems into key simple modules, posing questions, and implementing incentive mechanisms. Students, on the other hand, identify basic code corresponding to these problem modules, elaborate on their understanding of the code's application, and outline preliminary solutions, while teachers refine the remaining parts.

Taking the Python Programming course as an example, teachers decompose the problem of processing web data into sub-questions such as "How to send HTTP requests?", "How to store data?", and "How to count word frequencies?". Based on these key questions, students search for basic code like requests.get(), csv.writer(), and collections. Counter(), explain the reasons for their selection, and link them to propose a solution framework. This role transformation converts students from passive knowledge recipients into active knowledge constructors and expressers, compelling them to deepen their understanding of complex concepts through organizing logically coherent language. Additionally, with students at the center of the classroom and driven by incentive mechanisms, their internal motivation and sense of responsibility are fully stimulated. To successfully solve problems, they will take more initiative in consulting materials, testing code, and engaging in critical thinking.

In summary, implementing the PBL concept through the strategy of role transformation of dominant participants reconstructs teacher-student interaction patterns. It transforms the challenge of learning complex code into opportunities for students to demonstrate their abilities and exercise their thinking, effectively fostering their practical competence in programming research-based learning.

3.3 Construction of Rational AI Application

Rational utilization of AI tools serves as an auxiliary approach to enhancing the practical competence of big data majors in programming learning.

The rapid advancement of AI compels us to focus on whether AI can help improve programming competence with half the effort. The core value of this method lies in strengthening the problem-based learning approach in PBL theory, enabling students to acquire knowledge points more quickly and expand their knowledge scope more extensively during inquiry and collaboration. Meanwhile, this requires teachers not only to impart AI application skills to students in class but also to cultivate their critical thinking and reconstruction abilities regarding AI-generated code.

ISSN 2522-6398 Vol. 9, Issue 5: 222-230, DOI: 10.25236/FER.2026.090530

For example, in Python Programming teaching, when students handle list-related problems, they may obtain AI-generated code for quickly filtering even numbers using a list comprehension: [x for x in range(10) if x%2==0]. Teachers guide students to first analyze the AI-generated code—identifying that a list comprehension consists of three components (expression, loop, and condition)—then compare it with the traditional approach of for loop + append(), and further require students to implement the same function using the filter() function. This teaching method, starting from optimized AI-generated code and returning to basic grammar, not only leverages AI's demonstration role but also strengthens students' programming thinking through the process of "deconstruction-reconstruction".

This learning mode transforms students from passive code copiers into active code deconstructors. It utilizes AI to improve learning efficiency while deepening students' understanding of basic principles through code reconstruction. More importantly, for programming problems not covered in class or new topics, students can independently learn and understand with the assistance of AI, which significantly expands their reading scope and practical application range.

In summary, this model of critical AI application allows students to enjoy the convenience of programming while avoiding mental laziness, ultimately achieving dual improvement in programming practical competence and innovative thinking.

3.4 Encouraging the Openness of Teaching Materials and Knowledge Integration

Encouraging the openness of teaching materials and the integration of knowledge points is a crucial strategy for enhancing the practical competence of students majoring in big data.

In terms of the openness of teaching materials, when teachers release courseware, case codes, extended literature, and other resources on public platforms in advance, students can conduct pre-class preview at their own pace. By browsing these materials, they can clarify basic knowledge points, key contents, and difficult points, and participate in classes with targeted questions, thereby improving listening efficiency. After class, they can review knowledge points with the help of these resources and deepen their understanding by combining class notes. Especially for abstract content involved in the big data field, such as complex algorithm principles and distributed system architectures, open materials provide opportunities for repeated study and research, helping students break through the time and space limitations of classrooms to consolidate knowledge. Meanwhile, the open materials include programming codes related to classroom learning from competitions over the years, enabling students to be exposed to the difficulty of competitive programming in advance, recognize the practical value of programming knowledge, and thus enhance their learning motivation and competitive experience.

Regarding the integration of knowledge points by students, this process compels them to actively sort out classroom content and systematize scattered programming concepts and basic code snippets. For example, summarizing the implementation codes of different sorting algorithms and their application structures in big data processing frameworks not only strengthens the mastery of basic grammar but also helps understand the operating logic of code in complex systems. During the integration process, students need to reconstruct the knowledge summarized by teachers and establish their own integration systems in a familiar and understandable way. This helps cultivate logical thinking and inductive abilities. When the integrated content forms a system, it not only facilitates their own review but also allows communication and sharing with classmates, deepening understanding through mutual inspection and supplementation and identifying their own knowledge blind spots. In addition, the act of integration itself is a manifestation of active learning, which can promote students to transform from passive knowledge recipients to active knowledge explorers and gradually develop the habit of practice. This is particularly important for the big data field that requires continuous follow-up of technological development, laying a foundation for students' autonomous learning ability in their future career development.

4. Strategies for curriculum improvement

In the previous two sections, we have found that Problem-Based Learning (PBL)—a core concept of the PBL philosophy—plays a significant role in enhancing the practical competence of college students majoring in big data. In a broad sense, problem-oriented learning serves as the core carrier for big data majors to improve their practical competence and achieve the transformation from "passive learning" to "active creation". Therefore, we propose two curriculum improvement strategies based on

ISSN 2522-6398 Vol. 9, Issue 5: 222-230, DOI: 10.25236/FER.2026.090530

problem-oriented learning.

4.1 Allowing Big Data Majors to Adopt Multiple Methods for Solving Common Programming Problems

To enhance the practical competence of students majoring in big data, solutions to a specific programming problem should not be limited to a single approach—this can effectively foster their divergent thinking and innovative capabilities. For example, after learning three types of basic code: list comprehension, set initialization, and the dictionary get() method, students can gain an understanding of their functions.

Basic code examples:

```
# List comprehension

new_list = [expression for item in iterable if condition]

# Set initialization

my_set = {1, 2, 3} or my_set = set(iterable))

# Dictionary get() method

value = my_dict.get(key, default_value))
```

Subsequently, real programming questions from the National Competition of the 11th Blue Bridge Cup Youth Group (2020) are incorporated into the Python Programming course.

Example 1: Problem Description: Given a positive integer , calculate the total number of times the digit (where $0 \leq k \leq 9$) appears in all integers from 1 to n.

Students solve this problem using the three code constructs they have learned, following these steps: First, in terms of data preparation: compared with the traditional iterative approach for generating a string list, the list comprehension enables faster generation.

```
str_list = [str(i) for i in range(1, n + 1)]
```

Second, for the counting process: the get() method of dictionaries allows traversing all digits only once while simultaneously counting the occurrence frequency of each digit.

```
count_dict[char] = count_dict.get(char, 0) + 1
```

Additionally, if the problem were modified to “find the digit characters that never appear in all integers from 1 to n”, a set-based approach could be employed to optimize the solution logic.

```
all_digits_set = {char for num in range(1, n+1) for char in str(num)}
```

Finally, the teacher summarizes the solutions and integrates students’ code. More concise and innovative approaches are also provided, such as using ".join() and count(str(k)). This method simplifies the solution to a single line of code as supplementary content.

```
total = ''.join(str(i) for i in range(1, n+1)).count(str(k))
```

In Python Programming courses, some undergraduates have responded positively to the ongoing curriculum improvement strategies. Therefore, both theoretical and practical evidence indicates that solving common programming problems using multiple approaches is an effective strategy for enhancing the practical competence of students majoring in Big Data.

4.2. Provide Challenging Programming Problems for Undergraduates Majoring in Big Data

Beyond basic code constructs and fundamental programming problems, exposing students to more challenging programming problems in class can push them out of their comfort zones. This encourages students to proactively learn multiple coding approaches to solve complex problems, rather than mechanically applying basic code. To enhance the practical ability of Big Data majors in applying the PBL theory, it is essential to allocate part of the class time for students to engage with more challenging programming problems. For instance, the "Maze and Trap" problem from the National Final of the 9th Blue Bridge Cup Competition (Python Group, 2018) can be integrated into the "Python Programming" course.

ISSN 2522-6398 Vol. 9, Issue 5: 222-230, DOI: 10.25236/FER.2026.090530

Example 2: Problem Description: Xiaoming is in an N×N-sized maze, where there are some traps. When Xiaoming encounters a trap: If he has an invincibility state, he can pass through the trap; Otherwise, he cannot pass through the trap.

The list comprehension learned in Example 1 demonstrates significant advantages in Example 2. It enables efficient data processing, such as quickly locating the start position and trap positions:

start_pos = [(i, j) for i in range(n) for j in range(n) if maze[i][j] == 'S']

trap_positions = [(i, j) for i in range(n) for j in range(n) if maze[i][j] == 'X'].

Additionally, it enhances data readability—for instance, generating all possible movement positions:

next_positions = [(x + dx, y + dy) for dx, dy in directions]

Filtering out invalid moves:

valid_moves = [(nx, ny) for nx, ny in next_positions if 0 <= nx < n and 0 <= ny < n and maze[nx][ny] != '#'].

Furthermore, it offers fast access speed and high state integrity, as seen in initializing the visited marker array:

visited = [[[False] * (k + 1) for _ in range(n)] for _ in range(n)].

By refining the inherent functions of list comprehension to align with the specific requirements of the problem, we can explore the intrinsic value of list comprehension code in depth, transforming it from basic code into key code, thereby solving this problem.

In-depth learning of basic code through challenging programming problems enables students to understand the code's versatility and wide applicability. Shifting the focus from the code itself to the problem itself is a key strategy for undergraduates majoring in Big Data to enhance their practical competence.

5. Empirical Research Based on Mediating Effect

This section presents an empirical study based on the mediating effect. The mediating effect is a widely recognized statistical analysis method for exploring the indirect influence pathways between variables. This method can reveal the internal mechanism through which an independent variable exerts an impact on a dependent variable—specifically, whether the effect is achieved via one or more mediating variables—thereby enabling a more accurate understanding of the essence of the relationships between variables. The following analysis applies this technique to data from an educational research context, aiming to clarify the specific functional pathways through which teaching strategies contribute to the improvement of students' competencies. The research findings provide a solid theoretical and empirical foundation for interpreting the effect mechanism of innovative strategy interventions and supporting teaching decision-making.

To construct a dataset for analyzing inter-variable relationships, we collected data pertaining to PBL-oriented teaching engagement, learning mechanisms, and programming practical competence from Big Data majors at a university through a structured questionnaire survey. Concurrently, objective materials—including students' programming assignment scores and project practice reports—were incorporated to supplement and validate the survey data, thereby ensuring the reliability, validity, and comprehensiveness of the dataset for subsequent analytical processes.

In the variable definition, we adopted a 5-point Likert scale for measurement. Specifically, the independent variable (X) was defined as the implementation degree of the PBL concept (1 = "Not implemented at all", 5 = "Fully implemented"); the dependent variable (Y) was students' programming practical competence (1 = "Extremely poor competence", 5 = "Extremely strong competence"); and the mediating variables included: M1 = diversification of teaching scenarios (1 = "No relevance", 5 = "High relevance"), M2 = research-oriented learning with leader role transformation (1 = "Never", 5 = "Always"), M3 = rational construction of AI application (1 = "No ability", 5 = "Strong ability"), and M4 = openness of teaching materials and knowledge integration (1 = "Incomplete", 5 = "Extremely complete"). Subsequently, we constructed a parallel mediating effect regression model and calculated the direct effect regression model of X on Y:

ISSN 2522-6398 Vol. 9, Issue 5: 222-230, DOI: 10.25236/FER.2026.090530

$$Y = cX + e_1 \quad (1)$$

Regression model for X on each mediator variable:

$$M = aX + e_2 \quad (2)$$

Regression model of X on Y with a mediator variable:

$$Y = c'X + bM + e_3 \quad (3)$$

Based on the calculation of the mediating effect regression equations, we obtained the following results:

Table 1: Summary of Regression Analysis Results

Model	Regression Coefficient	Standard Error	*t-Value*	*p-Value*	R^2
Model1(Y with respect to X)	c =0.780	0.051	15.250	0.000**	0.704
Model2 (M_1 with respect to X)	a_1 =0.728	0.065	11.204	0.000**	0.562
Model2 (M_2 with respect to X)	a_2 =0.763	0.081	9.429	0.000**	0.476
Model2 (M_3 with respect to X)	a_3 =0.800	0.068	11.777	0.000**	0.586
Model2 (M_4 with respect to X)	a_5 =0.566	0.078	7.251	0.000**	0.349
Model3 (Y with Respect to X + M_1 □ M_2)	c' =0.159	0.064	2.509	0.014*	0.889
	b_1 =0.226	0.054	4.210	0.000**	
	b_2 =0.251	0.044	5.640	0.000**	
	b_3 =0.165	0.048	3.396	0.001**	
	b_4 =0.235	0.043	5.407	0.000**	

Table 2: Summary Results of Mediating Effect Test

Term	a*b Mediation Effect Value	a*b(Boot SE)	a*b (P-Value)	a*b (95%BootCI)	Test Conclusion
$X \geq M_1 \geq Y$	0.165	0.047	0.001***	0.07-0.259	Partial Mediation Effect
$X \geq M_2 \geq Y$	0.191	0.042	0.000***	0.122-0.292	Partial Mediation Effect
$X \geq M_3 \geq Y$	0.132	0.042	0.002***	0.054-0.227	Partial Mediation Effect
$X \geq M_4 \geq Y$	0.133	0.027	0.000***	0.084-0.189	Partial Mediation Effect

As indicated in Tables 1 and 2, enriching the diversity of teaching scenarios, research-based learning with role transformation of dominant participants, construction of rational AI application, and encouraging the openness of teaching materials and knowledge integration all play significant partial mediating roles between the PBL concept and students' programming practical competence. Among these, the M2 path contributes the most: it activates the learning transformation from "passive to active" and strengthens students' independent inquiry and collaborative abilities. Next is the M1 pathway, which serves as a bridge between theory and practice and mitigates the mismatch between acquired skills and practical application contexts. The M4 path consolidates the knowledge foundation from "fragmented to systematic," helping students construct a programming knowledge system. The M3 path resolves the competence paradox in AI usage, guiding students to rationally leverage AI rather

ISSN 2522-6398 Vol. 9, Issue 5: 222-230, DOI: 10.25236/FER.2026.090530

than over-rely on it. Tests via the Bootstrap method show that all path coefficients pass the significance test at , and none of the Bootstrap confidence intervals include 0, indicating that the results are robust and reliable.

This confirms that the PBL concept exerts its influence through a dual-path mechanism of "direct effect + four mediating indirect effects." On one hand, the core characteristics of the PBL concept—"problem-driven and independent inquiry"—can directly stimulate students' motivation for programming learning and cultivate their ability to think critically about solving complex problems. On the other hand, the PBL concept takes root through the four mediating variables, forming specific and operable improvement paths.

6. Conclusions

This study explores how to enhance the practical competence of Big Data majors in the "Python Programming" course through Problem-Based Learning (PBL). It addresses current teaching challenges, including the disconnect between skills and real-world application, insufficient student self-initiative, and the inefficient use of AI tools. Four core PBL strategies are proposed: Diversifying teaching scenarios; Implementing a research-oriented learning model through role-shifting; Leveraging AI tools to cultivate critical thinking and reconstruction skills; Promoting open teaching materials and student-led knowledge integration.

Practice demonstrates that these strategies stimulate students' practical exploration, facilitate a shift from passive knowledge acceptance to active knowledge construction, and improve their ability to analyze and solve complex problems. By incorporating multi-method problem-solving and challenging tasks such as real questions from the Blue Bridge Cup competition, students further consolidate their programming skills and practical competence. This study offers theoretical and practical insights for programming instruction in Big Data majors, providing a valuable reference for future curriculum reform and the cultivation of innovative talent.

Acknowledgement

This work is partially supported by College Students' Innovation and Entrepreneurship Projects for Suqian University (X2025141600360); Suqian Social Science Research Project "Research on the Construction of a Livelihood Expenditure Performance Evaluation System in the Context of Big Data-Enabled Equalization of Basic Public Services" (25SYS-36); Startup Foundation for Newly Recruited Employees and the Xichu Talents Foundation of Suqian University (2022XRC033) ; Qing Lan Project of Jiangsu; The interdisciplinary Integration and Innovation Project of Suqian university (2025XKTD02); Suqian Sci & Tech Program (M202206); Planning Project of the China Commercial Statistics Society (CCSS) (#2025STY26).

References

[1] Wu, G. (2012). A review of problem-based learning (PBL) model. Shaanxi Education (Higher Education Edition), (4), 3-7.
[2] Wang, Q. (2020). Why should children learn programming? Insights from the book Mindstorms: Children, Computers, and Powerful Ideas. Modern Educational Technology, 30(2), 122-126.
[3] Lu, H. L., Wang, B. B., & Gao, X. P. (2022). Research on the design and practice of project-based learning in the intelligent era: A case study of PBL teaching in Python course. Journal of Henan Institute of Education (Natural Science Edition), 31(2), 67-75.
[4] Liu, M., & Zheng, X. (2022). Practice of the "problem-oriented + action learning" teaching model. Journal of Electrical & Electronic Education, 44(6), 17-20.
[5] Dai, B. (2024). Application of project-based learning (PBL) in Python programming teaching. Integrated Circuit Applications, 41(12), 78-79.
[6] Dai, M. Y. (2025). Teaching reform and practice of Python programming course based on PBL concept. Computer Knowledge and Technology, 21(8), 171-173.