International Education Forum, 2025, Volume 3, Issue 12
http://ojs.bbwpublisher.com/index.php/IEF
Online ISSN: 2981-8605
Print ISSN 3083-4902

Exploration of a Dual-Helix-Driven Practical Teaching Model Integrating PDCA Cycle and Aerospace Cases

Jinli Li*

School of Space Information, Space Engineering University, Beijing 101416, China

**Author to whom correspondence should be addressed.*

Abstract: Aiming at the problems existing in traditional programming practice courses, such as fragmented cases, insufficient integration of aerospace scenarios, and a lack of process-oriented assessment, an innovative dual-helix-driven teaching model is proposed. Taking the Beidou Satellite Navigation System as a cross-cutting case, this model deeply integrates the PDCA (Plan-Do-Check-Act) quality management theory to construct a dual-helix teaching framework of "aerospace case traction + PDCA cycle control". By embedding core knowledge points such as Python syntax, data structures, and algorithm design into aerospace application scenarios, including Beidou satellite orbit calculation, positioning solution, and data visualization, the synergistic improvement of theoretical knowledge and application capabilities is realized.

Keywords: PDCA Cycle; Aerospace cases; PBL; Programming course; Process-oriented assessment; Engineering literacy

Online publication: December 31, 2025

1. Introduction

With its concise syntax, efficient numerical computing capabilities, and rich scientific computing libraries, Python has become a key tool for core aerospace tasks such as satellite orbit prediction, remote sensing data processing, and spacecraft control algorithm verification [1]. The "Basic Requirements for College Computer Foundation Courses Teaching (2021 Edition)" issued by the Ministry of Education clearly points out that programming courses should strengthen practical attributes, promote in-depth integration with professional scenarios, and cultivate students' computational thinking and engineering application capabilities [2,3]. For science and engineering universities with aerospace characteristics as the core, Python programming is a core basic course, and its teaching quality directly affects students' learning effects in subsequent professional courses and the formation of engineering literacy.

However, the current practical teaching of Python programming in our university has three major pain points that seriously restrict the improvement of talent training quality. Firstly, fragmented cases: the adoption of a teaching method of "grammar points + isolated cases" lacks a large cross-cutting case throughout the course, making it difficult for students to connect scattered knowledge points to form a complete knowledge system. Secondly, lack of aerospace scenarios: Python is not closely combined with aerospace application scenarios such as Beidou navigation and satellite communication, making it difficult for students to understand the application value of Python technology in actual engineering and resulting in insufficient learning motivation and goals. Thirdly, insufficient process-oriented assessment: over-reliance on final programming exams, lack of continuous evaluation of processes such as code standardization and performance optimization, which can neither fully reflect students' true abilities nor timely identify problems in the teaching process. This leads to students' insufficient engineering-oriented programming thinking and rigor, failing to meet the high standards for talents in the aerospace field. Therefore, there is an urgent need to construct a new Python teaching model adapted to aerospace professional characteristics through systematic teaching method research, so as to comprehensively improve students' computational thinking abilities.

2. Research status

Universities at home and abroad attach importance to the combination of case-driven teaching and Project-Based Learning (PBL) in practical teaching [4]. The Massachusetts Institute of Technology (MIT) provides a large number of course resources and project examples integrating Python with artificial intelligence and simulation computing through its OpenCourseWare (OCW) [5]; Stanford University's projects such as "Code in Place" have developed systematic Python teaching cases, emphasizing the connection of knowledge points with professional applications through community-based learning and real projects [6]. Focusing on the application of online platforms and blended teaching, Wang Qiang et al. constructed a Python programming teaching plan based on the EduCoder platform, improving students' practical abilities through automatic evaluation and online training [7]. However, existing research still has shortcomings: most cases lack professional pertinence and fail to form a systematic characteristic case throughout the course [8]; insufficient attention is paid to the dynamic control of the teaching process, making it difficult to achieve closed-loop optimization of "teaching - learning – assessment."

As a quality management tool for continuous improvement, the PDCA (Plan-Do-Check-Act) cycle has been widely applied in the field of education. Wu Jiazhou et al. applied PDCA to blended teaching, realizing the precise implementation of teaching objectives through the closed-loop control of "Plan-Do-Check-Act" [9]; Cui Wensheng et al. introduced PDCA into the practical teaching reform of programming courses, improving the standardization of experimental teaching and data-driven continuous improvement [10]. These studies have verified the effectiveness of PDCA in teaching quality control, but research on constructing a dual-helix-driven teaching method for programming courses by deeply integrating it with aerospace characteristic cases is still at a blank stage.

3. Design of the PDCA-aerospace case dual-helix-driven teaching model

This paper constructs a "PDCA-aerospace case dual-helix-driven" teaching model, designs a Beidou satellite navigation case system throughout the course to realize the organic integration of core Python knowledge

points and aerospace application scenarios; builds a teaching control process based on the PDCA cycle to form a closed loop of "Plan-Do-Check-Act," establishes a full-process practical platform based on the EduCoder platform, and conducts whole-process evaluation to ensure the continuous improvement and optimization of the teaching process.

3.1. Overall framework of the teaching model

Drawing on the synergistic mechanism of the DNA double-helix structure, the curriculum team constructs a dual-helix teaching system of "Beidou task main line + PDCA control line", with "knowledge transfer and ability training" as the core. Through the mutual interweaving and dynamic coordination of the two spirals, the systematization and standardization of the teaching process are realized.

3.1.1. Core components of the dual helix

(1) Main Spiral (Beidou Task Line)

Taking the full-process work tasks of the Beidou Satellite Navigation System as the logical main line, it runs through all teaching content of the Python course. According to the engineering process of "satellite orbit calculation → ground station data reception → user positioning solution → trajectory visualization", a four-level progressive task chain is constructed. Core Python knowledge points such as basic syntax, functions and modules, file operations, exception handling, object-oriented programming, and data visualization are embedded into corresponding task links one by one, forming a progressive learning path of "scenario cognition → grammar learning → algorithm design → task implementation".

(2) PDCA Spiral (Quality Control Line)

The four links of the PDCA cycle are embedded into each teaching module to implement quantitative control over the entire process of "teaching" and "learning". Each module is promoted in accordance with the process of "clarifying goals → practical operation → evaluation and diagnosis → optimization and improvement", ensuring the precise implementation of teaching objectives and timely solving problems in students' learning and teachers' teaching.

3.1.2. Synergy mechanism of the dual helix

The synergistic operation of the two spirals is realized through the three-dimensional coupling of "task decomposition - knowledge mapping - quality control". Task decomposition: decompose the full-process Beidou navigation tasks into subtasks corresponding to teaching chapters to ensure the adaptation of tasks to teaching progress; Knowledge mapping: establish a "subtask - knowledge point" correspondence table to clarify the Python syntax, library functions and other contents required for each task, avoiding knowledge omissions; Quality control: through the PDCA cycle, conduct dual evaluation of the "learning process" and "task results" of each subtask, and adjust teaching strategies and task difficulty according to the evaluation results to realize the synchronization of "task advancement" and "quality control".

The core innovation of this model lies in breaking the closed nature of traditional "knowledge transmission - mechanical practice". On the one hand, aerospace cases endow abstract programming knowledge with engineering significance, stimulating students' exploration desire through the rigor and innovation of the aerospace field; on the other hand, the PDCA cycle endows the teaching process with

dynamic optimization capabilities, adjusting the teaching rhythm through continuous feedback to adapt to students' cognitive laws.

3.2. Design of the cross-cutting Beidou satellite navigation case library

Based on teaching objectives, following the principles of "full coverage of knowledge points, gradient difficulty, and realistic scenarios", a case library running through the entire course is designed. Each teaching chapter corresponds to a Beidou navigation-related case module, clarifying case objectives, required knowledge points, and PDCA control requirements to ensure the close combination of knowledge learning and aerospace applications. The specific design is shown in **Table 1**.

Table 1. Design table of Beidou navigation cases for each chapter

Teaching Chapters	Beidou Case Modules	Python Knowledge Points	PDCA Control Objectives
Basic Syntax	Satellite Orbit Parameter Parsing	Variables, data types, conditional statements	Code standardization rate ≥ 85%, TLE format parsing accuracy 100%
Functions and Modules	Orbit Calculation Function Encapsulation	Function definition, parameter passing, module import	Function reuse rate ≥ 90%, calculation error ≤ 100m
File Operations	Batch Processing of Ephemeris Data	File reading and writing, CSV/JSON format parsing	Data processing efficiency increased by 30%, exception handling coverage 100%
Exception Handling	Communication Link Fault Simulation	try-except statements, logging	Program robustness score ≥ 90 points
Object-Oriented Programming	Beidou User Terminal Simulation	Classes and objects, inheritance and polymorphism	Class design rationality ≥ 85%, code scalability score ≥ 90
Data Visualization	Satellite Trajectory and Positioning Result Display	Matplotlib/Pyecharts plotting	Visualization effect score ≥ 85 points, chart interactivity up to standard
Comprehensive Module	Simple Simulation of Beidou Navigation System	Comprehensive application of knowledge points, system integration, document writing	System function realization rate ≥ 90%, code operation efficiency up to standard, complete and standardized project documents

The following principles are adhered to in case design: first, authenticity, such as ephemeris data referring to the public service performance specifications of Beidou, close to engineering practice; second, relevance, realizing vertical connection of knowledge; third, gradient, with difficulty gradually increasing from "basic operation" to "comprehensive application". Cases in basic chapters focus on "knowledge point mastery," while cases in comprehensive modules focus on "systematic thinking training," adapting to students' cognitive laws. At the same time, the PDCA cycle is used to continuously improve and optimize cases to ensure that cases are always consistent with teaching objectives.

3.3. Design of PDCA-based teaching implementation process

Taking each teaching module (corresponding to a single chapter case in **Table 1**) as a unit, a closed-loop implementation process of "Plan-Do-Check-Act" is designed, clarifying the operation points, participating subjects, and output results of each link to ensure that the teaching process is controllable, evaluable, and optimizable.

3.3.1. Plan phase: Clarify goals and decompose tasks

The Plan phase is the starting point of the PDCA cycle. Its core is to clarify module tasks and implementation requirements based on teaching objectives and students' foundations. The specific process is as follows:

(1) Goal Setting: Combine curriculum standards and aerospace talent needs to formulate "knowledge goals", "ability goals", and "literacy goals" for the module. For example, the knowledge goal of the "batch processing of ephemeris data" module is "master CSV/JSON format parsing", the ability goal is "possess multi-source data batch processing capabilities", and the literacy goal is "cultivate rigorous engineering data processing thinking";

(2) Task Decomposition: Decompose the module case into multiple operable subtasks, clarifying the requirements and time nodes of each subtask. Taking "batch processing of ephemeris data" as an example, it is decomposed into subtasks of "data reading → data cleaning → data calculation → result output → exception handling", and class hours are specified for each subtask;

(3) Resource Preparation: Teachers prepare teaching resources such as case documents, reference code snippets, and aerospace background materials and upload them to the EduCoder platform; at the same time, identify knowledge weaknesses based on students' learning data from previous modules to provide a basis for subsequent tutoring;

(4) Standard Formulation: Clarify the evaluation standards for subtasks, form a "Module Evaluation Standard Table," and announce it to students in advance.

3.3.2. Do phase: Practical exploration and collaborative guidance

The Do phase focuses on students' independent practice and teachers' precise guidance. The specific process is as follows:

(1) Scenario Introduction: Introduce the case scenario through videos of Beidou navigation system applications such as satellite positioning demonstrations and aerospace task simulations, and explain module objectives combined with the significance of aerospace engineering to stimulate students' interest.

(2) Knowledge Explanation: Teachers explain core knowledge points combined with case needs, focusing on explaining the application methods of knowledge points in aerospace scenarios. For example, when explaining loop statements, the application scenario of for loops is illustrated in combination with the need for "line-by-line parsing of ephemeris data".

(3) Independent Practice: Students complete programming practice on the EduCoder platform. The platform provides an online programming environment and real-time syntax checking functions, allowing students to submit code at any time to obtain feedback; group collaboration is adopted for complex subtasks with clear division of labor;

(4) Guidance and Support: Teachers real-time monitor students' progress through the platform, conduct centralized explanations for common problems, and one-on-one tutoring for individual problems; at the same time, encourage students to solve problems through group discussions and online forum exchanges.

3.3.3. Check phase: Multi-dimensional evaluation and problem diagnosis

The core of the Check phase is to comprehensively grasp teaching effects through multi-dimensional

evaluation and identify problems in “teaching” and “learning”. The specific process is as follows:

(1) Result Evaluation: Based on the automatic evaluation function of the EduCoder platform, quantitatively score the “correctness”, “efficiency”, and “standardization” of students’ code; evaluate their understanding of knowledge points in combination with the experimental reports submitted by students (reports should include “problem analysis - implementation ideas - result summary”);

(2) Process Evaluation: Evaluate students’ learning attitudes and independent problem-solving abilities through process data recorded by the platform, such as the number of code submissions, debugging time, and number of help requests. For example, “excessive number of code submissions” may reflect insufficient logical understanding, and “long debugging time” may reflect weak algorithm design capabilities;

(3) Multi-Source Evaluation: Supplement teacher evaluation with “student self-evaluation + group peer evaluation”. Students’ self-evaluation needs to reflect on “learning gains - existing problems”, and group peer evaluation assesses members’ contribution to collaborative tasks;

(4) Problem Summary: Based on the evaluation results, form a “Module Problem List”, sort out common problems and individual problems by category, and clarify the causes of problems, such as “insufficient explanation of knowledge points” or “excessively high case difficulty”.

3.3.4. Act phase: Optimization and improvement, experience solidification

The Act phase is the key to the PDCA cycle. Its core is to formulate improvement measures for the problems found in the Check phase, providing an optimization direction for the teaching of subsequent modules. The specific process is as follows:

(1) Problem Rectification: For common problems, solve them through “supplementary teaching” (such as adding special lectures on “time system conversion”) or “task optimization” (such as adjusting case data difficulty); for individual problems, help students improve through “additional exercises” (such as providing basic supplementary practice cases) or “one-on-one tutoring”;

(2) Teaching Optimization: Adjust teaching strategies according to the causes of problems. For example, “insufficient explanation of knowledge points” can be improved by means of “animation demonstration + example decomposition”, and “excessively high case difficulty” can be solved by adding “step-by-step guidance prompts”;

(3) Experience Solidification: Sort out and file successful experiences in module teaching (such as “scenario introduction methods” and “collaborative grouping models”) and optimized resources to form a “Module Teaching Resource Package” for application in subsequent teaching;

(4) Continuous Cycle: Incorporate unresolved problems (such as “insufficient algorithm optimization capabilities of some students”) into the Plan phase of the next module to realize the continuous operation of the PDCA cycle.

3.4. Design of teaching support conditions

To ensure the smooth implementation of the dual-helix-driven teaching method, a trinity support system of “platform + resources + teachers” is constructed, clarifying the construction requirements and functional positioning of each support condition.

Firstly, select the EduCoder platform as the core online support tool, matching teaching needs based

on its functional characteristics: an online programming environment supporting Python 3.x, with built-in required libraries such as NumPy and Matplotlib, eliminating the need for students to install local environments; an automatic evaluation function allowing teachers to preset evaluation cases, with the platform providing real-time feedback on scores after students submit code; process data recording that automatically records students' programming process data, providing a quantitative basis for process evaluation; a resource management function supporting the upload of courseware, videos, and case documents for students to access at any time; and experimental teaching assistant Q&A services.

Secondly, in terms of teaching resources, compile a characteristic lecture note "Python Programming (Aerospace Application Edition)", integrating case operation steps, aerospace background knowledge, and knowledge point analysis; refer to the public service performance specifications of the Beidou Satellite Navigation System, sort out real ephemeris data and TLE orbital element data, and construct a "Beidou Case Database" to ensure data authenticity and availability.

Thirdly, build a double-qualified teaching team of "Python teaching + aerospace professionals", inviting professional teachers with aerospace engineering practice experience to answer students' professional questions about the Beidou navigation system; regularly carry out teaching seminars to jointly optimize case design and teaching processes, ensuring the in-depth integration of programming knowledge and aerospace scenarios.

4. Curriculum implementation plan and expected effects

The curriculum implementation is promoted in three phases to ensure the stable landing and continuous optimization of the teaching model. The first phase is the pilot verification phase: select 2 classes as pilots, initially deploy the case library based on the Educoder platform, and carry out teaching in accordance with the designed teaching model; focus on recording teaching data of each module during the teaching process, including students' code submission status, evaluation results, and problem feedback, to form a teaching report for the first phase. The second phase is the promotion and application phase: promote the teaching model to the entire grade, optimize the case library and teaching process according to feedback from the pilot phase, and improve the quantitative standards of the multi-modal evaluation system. The third phase is the iterative optimization phase: continuously collect teaching data based on the PDCA cycle, regularly review the teaching model, update case content in combination with needs; establish a communication mechanism to promote teaching achievements.

5. Conclusion

Through the in-depth integration of Beidou cases and PDCA, a new paradigm of Python practical teaching featuring "demand-driven - process control - continuous improvement" is constructed, breaking the traditional teaching closed loop of "knowledge transmission - mechanical practice"; taking the Beidou Satellite Navigation System as a cross-cutting case, the organic combination of Python knowledge points and aerospace applications is realized; at the same time, the PDCA cycle endows the teaching process with dynamic optimization capabilities, effectively solving the three major teaching pain points of traditional courses. Through phased curriculum implementation and improved guarantee measures, it is expected to significantly improve students' code quality, engineering practice capabilities, and systematic thinking, while

forming high-quality teaching resources and replicable teaching experience.

In the future, further innovation and optimization of the teaching model will be carried out: first, explore the in-depth combination of PDCA and AI teaching assistants, using AI technology to achieve precise learning situation diagnosis, automatic generation of personalized learning paths, and intelligent real-time Q&A, improving the pertinence of teaching guidance; second, expand the coverage of the case library, enrich teaching content by combining new technologies and applications in the aerospace field, and add complex cases to meet the needs of advanced learning; third, strengthen interdisciplinary integration, deeply combining Python programming with aerospace engineering, artificial intelligence, data science and other disciplines to cultivate compound innovative talents, providing stronger talent support for the development of China's aerospace industry.

Disclosure statement

The author declares no conflict of interest.

References

[1] Song T, Huang T, Li X, 2019, Python Language: An Ideal Choice for the Teaching Reform of Programming Courses. Computer Education, (2): 1–5.

[2] Ministry of Education of the People's Republic of China, 2021, Basic Requirements for College Computer Foundation Courses Teaching. Ministry of Education of the People's Republic of China, Beijing.

[3] Ren L, Han X, Luo C, 2025, Reform and Exploration of "Programming" Courses Oriented to Cultivating College Students' Computational Thinking. Industry and Information Technology Education, (10): 50–53.

[4] Mandal S, Das S, 2018, Application of PDCA Cycle for the Quality Enhancement of Engineering Education. International Journal of Quality & Reliability Management, 35(9): 1866–1884.

[5] Massachusetts Institute of Technology, 2016, MIT OpenCourseWare: Introduction to Computer Science and Programming in Python. MIT OpenCourseWare, visited on November 7, 2025, https://ocw.mit.edu/courses/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/

[6] Stanford University, 2020, Code in Place. Stanford University, visited on November 10, 2025, https://codeinplace.stanford.edu/

[7] Chen Y, Li J, Liu J, 2022, Online Training Teaching Reform of Python Language Courses Based on EduCoder Platform. Computer Education, (10): 178–182.

[8] Zhao L, Wang D, 2022, Research on the Construction of Python Programming Case Library for Aerospace Majors. Experimental Technology and Management, 39(4): 179–182.

[9] Wu J, Zhang S, Liu J, 2023, Teaching Reform of C Language Course Based on OBE + PDCA. Computer Education, (2): 192–197.

[10] Cui W, Li X, 2025, Research and Practice on the Assessment Scheme of Practical Teaching in Programming Courses—Exploration of Curriculum Reform in Applied Universities Guided by OBE Concept. Journal of Innovative Education Research, 13(10): 116–124.