International Journal of Education and Social Development
ISSN: 3078-9931 (Print), ISSN: 3078-2287 (Online) | Vol. 5, No. 1, 2025

Analysis of Python Applications in Operations Research Teaching

Xinyuan Guo, Yunna Liu *

School of Management, Tianjin University of Technology, Tianjin, China

* **Corresponding author**: (Email: liu.yunna@foxmail.com)

Abstract: As a cross-disciplinary field focused on optimal decision-making, operations research teaching has long relied heavily on theoretical derivations and specialized commercial software (e.g., Lingo, CPLEX), leading to issues such as disconnection between theory and practice, weak hands-on skills among students, and high learning barriers. In recent years, Python, with its concise syntax, robust open-source ecosystem, and widespread application in big data and artificial intelligence, has emerged as an ideal tool for reforming operations research teaching. This paper systematically analyzes the advantages of Python in operations research teaching and elaborates on its implementation through typical teaching cases, including linear programming, integer programming, network optimization, and simulation. Analysis suggests that Python can transform abstract mathematical models into intuitive and executable code, effectively enhancing students' modeling capabilities, programming skills, and learning motivation. It serves as a key driver in advancing the transition of operations research teaching toward an integrated "theory-modeling-solving-analysis" paradigm. Finally, this paper also discusses the challenges and corresponding strategies of incorporating Python into teaching.

Keywords: Python, Operations Research Teaching, Mathematical Modeling, Educational Reform.

1. Introduction

Operations Research (OR) provides scientific support for optimal decision-making in complex systems through the application of mathematical methods, statistical tools, and computer technology. Its core teaching content includes linear programming, integer programming, dynamic programming, graph and network theory, queuing theory, inventory theory, simulation, and more [1]. The traditional teaching model of operations research faces two main challenges: firstly, it overemphasizes the derivation of mathematical theories and manual computation, making it difficult for students to deeply understand the application scenarios of complex models; secondly, the experimental components often rely on tools such as Lingo, MATLAB, or commercial solvers [2]. While these tools offer high computational efficiency, they often operate as "black-box" solutions with proprietary syntax, which hinders students' ability to grasp the complete process from problem formulation to modeling.

In this context, the Python language stands out due to its unique advantages. Firstly, its syntax resembles natural language, resulting in a gentle learning curve that is particularly suitable for students from non-computing backgrounds to quickly get started [3]. Secondly, Python boasts an exceptionally rich and vibrant ecosystem for scientific computing. Libraries such as NumPy, SciPy, and Pandas are used for numerical computation and data processing; Matplotlib and Seaborn for data visualization; while specialized libraries like PuLP, CVXPY, OR-Tools, and Scikit-learn cover various operations research problems, spanning from linear programming to combinatorial optimization [4,5]. Finally, as a general-purpose programming language, Python enables the seamless integration of the entire workflow—from data acquisition and preprocessing to model building, solution, result analysis, and visualization—empowering students to build end-to-end solutions.

This paper aims to provide an in-depth analysis of Python's utility in operations research teaching, demonstrate its implementation pathways through concrete teaching cases, and discuss the pedagogical transformations it enables as well as potential challenges.

2. Advantages of Python Applications in Operations Research Teaching

2.1. Lower the Learning Curve and Focus on Model Essence

Python's concise and clear syntax allows students to focus less on complex programming constructs and more on understanding the structure and logic of operations research models. For instance, when using the PuLP library to build a linear programming model, the code closely mirrors the mathematical formulation—directly translating decision variables, objective functions, and constraints—thereby significantly reducing the abstraction inherent in modeling.

2.2. Establish a Unified Modeling-Solving-Analysis Platform

In traditional teaching, students often need to switch between different software tools. Python, however, provides a unified environment [6]. Students can use Pandas to read and manage data, employ PuLP or CVXPY to formulate optimization models, call interfaces of solvers such as CBC, GLPK, or commercial ones like Gurobi and CPLEX to solve the models, and finally visualize the results using Matplotlib. This integrated workflow cultivates students' systems thinking in solving practical problems.

2.3. Robust Open-Source Ecosystem and Vibrant Community Support

The open-source nature of Python enables free access to all tools, facilitating installation on personal computers and

supporting after-class practice. Its vast developer community offers extensive resources such as tutorials, comprehensive documentation, and Q&A platforms (e.g., Stack Overflow), allowing students to promptly resolve issues and thereby cultivating self-directed learning and problem-solving skills.

2.4. Bridge Academic and Industrial Applications

Python has become the de facto standard in contemporary data science and artificial intelligence. Its adoption in operations research teaching ensures that the skills students acquire are directly aligned with industry demands, significantly enhancing their employability. Furthermore, leading companies such as Google (with its OR-Tools) actively develop and release Python-based optimization toolkits, creating seamless integration between academic instruction and industrial practice.

3. Typical Teaching Cases and Application Practices

Taking a production planning problem as an example, suppose a factory plans to produce two products, A and B. Assume that producing each unit of product A requires 2 hours of labor and 4 kg of raw materials, while producing each unit of product B requires 3 hours of labor and 2 kg of raw materials. The factory has 100 hours of labor and 120 kg of raw materials available per day. The profit from producing each unit of product A is 30 yuan, and each unit of product B is 20 yuan. What production plan should the factory adopt to maximize its profit?

This problem can be represented by the following mathematical model:

$$\max Z = 30x_1 + 20x_2 \quad s.t. \begin{cases} 2x_1 + 3x_2 \le 100 \\ 4x_1 + 2x_2 \le 120 \\ x_1 \ge 0, x_2 \ge 0 \end{cases} \tag{1}$$

Fig. 1 presents the sample code for solving the aforementioned production planning problem using Python and the PuLP library. This code demonstrates how to transform a mathematical model into executable code, while Fig. 2 displays the optimal production plan obtained through the solution process:

```python
import pulp
model=pulp.LpProblem( name: "Production_Planning_Optimization", pulp.LpMaximize)
x1=pulp.LpVariable( name: 'Product_A',lowBound=0,cat='Continuous')
x2=pulp.LpVariable( name: 'Product_B',lowBound=0,cat='Continuous')
model+=30*x1+20*x2,"Total_Profit"
model+=2*x1+3*x2<=100,"Labor_Constraint"
model+=4*x1+2*x2<=120,"Material_Constraint"
model.solve()
print("Solution status:",pulp.LpStatus[model.status])
print("Optimal Production Plan:")
print(f"Product A Quantity (x1): {x1.varValue:.2f} units")
print(f"Product B Quantity (x2): {x2.varValue:.2f} units")
print(f"Maximum Profit: {pulp.value(model.objective):.2f} dollars")
```

Figure 1. Integer programming code

```
Solution status: Optimal
Optimal Production Plan:
Product A Quantity (x1): 20.00 units
Product B Quantity (x2): 20.00 units
Maximum Profit: 1000.00 dollars
```

Figure 2. Output results

This case clearly demonstrates the complete workflow from problem definition and mathematical modeling to code implementation. Through the code, students can directly observe the correspondence between variables, objectives, and constraints, significantly deepening their understanding of linear programming models. By modifying parameters and re-executing the solution, they can immediately observe how the solution changes, achieving interactive learning.

4. Teaching Challenges and Strategies

While the advantages of introducing Python into operations research teaching are significant, educators may encounter a series of challenges during practical implementation. Clearly recognizing these challenges and formulating effective countermeasures is a prerequisite for the successful implementation of this educational reform.

4.1. Significant Disparities in Students' Programming Backgrounds

The student cohort in the course may come from various majors such as Management Science, Industrial Engineering, Logistics Management, etc., with significantly varying levels of computer programming foundation [7]. Some students may already have experience with Python or other programming languages, while others may be complete beginners with no prior coding experience. This considerable disparity can make it difficult to maintain a uniform teaching pace: beginners may struggle to keep up, leading to frustration and anxiety, while students with prior experience may find the content too basic and become disengaged.

To address this situation, an intensive Python foundation short course can be organized. This course should be scheduled 1-2 weeks before the official semester begins, utilizing weekends or evenings, with a total duration of 8-12 instructional hours [8]. The content should be highly focused on core concepts relevant to the course, such as: variables and data types, lists/dictionaries, loops and conditional statements, function definition and calls, as well as how to import and use third-party libraries (e.g., NumPy, Pandas).

4.2. Instructional Hours Conflict between Theory and Practice

The operations research curriculum inherently encompasses substantial rigorous mathematical theories (e.g., simplex method, duality theory, optimality conditions, etc.). Introducing programming practice will inevitably compress the time originally allocated for theoretical instruction, potentially resulting in students who can implement computational solutions without understanding the underlying algorithmic principles.

To address this issue, a blended online-offline teaching model can be adopted [9]. Specifically, explanations of certain theoretical content can be recorded as short videos requiring pre-class viewing. These videos may include animated demonstrations of algorithmic concepts and theoretical derivations of mathematical models. Subsequently, in-person classroom sessions can focus on clarifying difficulties regarding core theories, explaining programming and modeling methodologies, and facilitating group collaboration on practical case studies, thereby achieving "theory outside the classroom, and practice and deepening inside the classroom [10]."

Furthermore, a "project-driven" approach can be implemented for post-class practice. The main component of programming practice can be structured as a series of assignments or a semester-long project. Classroom time would then be dedicated to demonstrating key code segments and explaining core methodologies [11]. This approach ensures the central role of theoretical instruction while deepening understanding through practical application.

4.3. Faculty Team's Knowledge Base and Teaching Inertia

Faculty members experienced in traditional operations research instruction may be more familiar with tools like MATLAB and Lingo, while relatively unfamiliar with the Python ecosystem. Mastering a new set of tools and preparing corresponding teaching materials requires substantial time and effort, creating certain transition barriers.

Therefore, systematic faculty development programs should be implemented. Academic departments should organize regular training sessions conducted by experienced internal and external experts, covering content ranging from Python fundamentals to advanced applications of optimization libraries [12]. Establishing course teaching teams to implement collaborative lesson preparation and teaching resource sharing can effectively reduce the preparation burden on individual instructors. Furthermore, open-source community and industrial resources should be fully utilized, encouraging faculty to engage with education-related topics at conferences such as PyCon and SciPy. Notably, the official documentation of many open-source libraries (e.g., PuLP, CVXPY) provides excellent teaching exemplars that can be directly utilized.

4.4. Cognitive Misconceptions Regarding "Black-Box" Operations

Students (and even some instructors) may harbor the misconception that Python's computational speed is insufficient for large-scale problems. Furthermore, overreliance on high-level interfaces like model. solve() can lead students to perceive solvers as "magical black boxes," focusing solely on obtaining solutions while neglecting the underlying algorithmic mechanisms. This approach fundamentally contradicts the essential purpose of operations research teaching, which is to cultivate systematic optimization thinking.

In pedagogical practice, it must be clearly communicated to students that Python libraries like PuLP and CVXPY function as modeling languages, whose fundamental purpose is to provide user-friendly human-computer interfaces for problem formulation. The actual computational solving is executed by underlying high-performance solvers (such as CBC, GLPK, Gurobi, and CPLEX) implemented in C/C++/Fortran. Python's distinctive value resides in its modeling convenience and integrative capabilities, rather than in the computational efficiency of the algorithms per se. This clarification helps students accurately comprehend the respective roles of modeling tools and solvers within the optimization workflow, while properly recognizing Python's unique advantages in constructing comprehensive end-to-end "modeling-solving-analysis" pipelines.

For complex models, students should be encouraged to utilize high-level interfaces for rapid solution generation, enabling model validation and solution acquisition while focusing on modeling concepts and result analysis. For fundamental algorithms (such as the simplex method, Dijkstra's shortest path algorithm, genetic algorithms, etc.), students should be required to implement core computational steps from scratch or using NumPy in Python. This process enables students to deeply comprehend the algorithm's iterative process, convergence properties, and computational complexity, thereby demystifying the "black box" phenomenon. For instance, implementing a basic simplex tableau iteration procedure provides significantly deeper theoretical understanding than merely invoking a solver.

5. Summary

With its simplicity, unified workflow, and powerful ecosystem, Python has breathed new life into operations research teaching. It transforms abstract mathematical theories into executable, verifiable, and visualizable computational processes, significantly enhancing the intuitiveness and interactivity of teaching. By integrating Python into the curriculum, the research not only cultivate students' core competency in "using mathematics and programming to solve real-world problems" but also equip them with a toolchain aligned with contemporary technological advancements. This lays a solid foundation for their future career development in fields such as data analysis, supply chain management, and fintech. Therefore, actively promoting Python-based reform in operations research teaching is an essential step to meet the talent development demands of the digital and intelligent era.

References

[1] Yu Peitai, Chen Huijie, Wang Yinghua, Feng Xue. Research on Teaching Reform of Operations Research Course in Management Science and Engineering Majors under the Context of Application-Oriented Universities [J]. Modern Business Trade Industry, 2024, 45(08): 255-258.

[2] Alexej Moskovka, Talal Rahman, Jan Valdman, Jon Eivind Vatne. Frameworking the vectorized basic linear algebra for prototyping codes in MATLAB [J]. Applied Mathematics and Computation, 2026, 513.

[3] Ma Huiyu, Pan Haizhu, Yang Xinyu, et al. Research and Practice of Ideological and Political Education in Curriculum Based on Python Experimental Teaching [J]. Journal of Science of Teachers' College and University, 2024, 44(11): 85-90.

[4] Kai Nylund, Jennifer Mankoff, Venkatesh Potluri. MatplotAlt: A Python Library for Adding Alt Text to Matplotlib Figures in Computational Notebooks [J]. Computer Graphics Forum, 2025, 44(3): e70119-e70119.

[5] Nurimangu·Yasen, Rozi Memet·Mahemuti. Research on the Application of Python in Transportation Problems [J]. Logistics Technology, 2024, 47(22): 82-85.

[6] Gu Jing, Li Yuqi, Li Siyu. Construction of a Value-Guided, Knowledge-Practice Integration, and Digital Intelligence-Supported Teaching System for Operations Research Courses [J]. Journal of Neijiang Normal University, 2025, 40(08): 121-128.

[7] He Caixia, Lyu Yanan, Cen Weifu. Practice and Exploration of PBL Teaching Model in "Python Programming" under the Background of Emerging Engineering Education [J]. Journal of Zunyi Normal University, 2025, 27(05): 144-149.

[8] Wang Yongmei, Zheng Yongai, Wang Yingying. Construction and Research of a Student-Centered "Golden Course" Teaching Model for Python Programming [J]. Information and Computer, 2025, 37(03): 176-179.

[9] Lu Yonglai. Construction and Application of Teaching Case Library for "Python Data Analysis" Course [J]. Office Informatization, 2024, 29(19): 41-44.

[10] Bai Xuejie, Zhang Yaqian. Practice and Exploration of Operations Research Teaching Reform Based on PBL Teaching Model [J]. Education and Teaching Forum, 2020, (03): 131-133.

[11] Lei Bing. Integrating Professional Characteristics with Emphasis on Practical Application: A Summary of Teaching Reform in Operations Research Course [J]. Journal of Zhengzhou Industrial College, 2000, (04): 51-52.

[12] Jiang Lin. Exploration of Teaching Reform in Operations Research at Newly-Upgraded Undergraduate Universities [J]. Examination and Evaluation, 2016, (02): 117-119.