Int. j. adv. multidisc. res. stud. 2025; 5(1):295-298

Received: 25-11-2024
Accepted: 05-01-2025

International Journal of Advanced Multidisciplinary Research and Studies

ISSN: 2583-049X

Connecting Discrete Structures and Python: Transforming Discrete Structures Education

Bui Trong Hieu
Institute of Information Technology and Electrical–Electronic Engineering, Ho Chi Minh City University of Transport, Ho Chi Minh City, Vietnam

DOI: https://doi.org/10.62225/2583049X.2025.5.1.3653
Corresponding Author: **Bui Trong Hieu**

Abstract
This paper presents a Python-integrated framework for teaching Discrete Structures, focusing on bridging the gap between theoretical concepts and practical applications. The primary goals of this framework are to enhance student engagement, improve conceptual clarity, develop computational thinking, and provide real-world relevance to abstract topics. By leveraging Python's intuitive syntax and versatile libraries, such as itertools for combinatorics and networkx for graph theory, the framework transforms traditional teaching methods into a hands-on, interactive learning experience. The framework aims to help students connect core topics like propositional logic, set theory, and graph traversal with their computational implementations. For instance, students can write Python programs to generate truth tables, explore set operations, or visualize graph algorithms. These activities foster active learning by providing immediate feedback and enabling experimentation, which promotes critical thinking and problem-solving skills. Another key goal is to demonstrate the real-world relevance of Discrete Structures. Students learn to apply modular arithmetic in cryptography, graph algorithms in network analysis, and recursion in algorithmic problem-solving. This interdisciplinary approach prepares students for modern technological challenges while aligning theoretical knowledge with practical skills. Initial evaluations show significant improvements in student engagement, comprehension, and performance. By integrating Python into the teaching of Discrete Structures, this framework not only bridges the gap between theory and practice but also equips students with the tools and mindset needed to excel in computational fields.

Keywords: Active Learning, Computational Thinking, Discrete Structures, Educational Framework, Problem-Solving Skills, Python Programming

1. Introduction

Discrete Structures form a foundational pillar in computer science, encompassing topics such as logic, set theory, combinatorics, graph theory, and Boolean algebra. Despite its significance, the abstract nature of Discrete Structures often poses challenges for students. Many struggle to grasp the underlying concepts or to connect them with practical applications, leading to disengagement and a lack of appreciation for the subject's relevance. Traditional teaching methods, which rely heavily on theoretical exposition and rote problem-solving, frequently fail to address these challenges, leaving students unprepared for real-world computational tasks [1].

The integration of programming into Discrete Structures education offers an effective solution to these challenges. Python, with its intuitive syntax and extensive libraries, serves as an ideal tool for bridging the gap between abstract theory and practical application [2]. Python, with its intuitive syntax and extensive libraries, serves as an ideal tool for bridging the gap between abstract theory and practical application [3]. For example, students can use Python to generate truth tables for logical expressions [4], visualize graph algorithms [5], and explore combinatorial problems [6].

This paper introduces a Python-integrated framework for teaching Discrete Structures, designed to enhance student engagement, improve conceptual clarity, and develop computational thinking skills. The framework focuses on bridging theory and practice by embedding hands-on programming tasks into the learning process. Key topics such as propositional logic [7], graph traversal [8], set operations [9], and modular arithmetic are taught alongside their Python implementations, allowing

students to explore concepts interactively and in context. Additionally, the framework emphasizes the real-world relevance of Discrete Structures, highlighting applications in cryptography [10], data science [11], and network analysis [12] to inspire students and prepare them for interdisciplinary challenges [13].

The primary goals of this framework are:

1. To enhance student engagement by integrating interactive Python-based exercises into the curriculum.
2. To improve learning outcomes by providing immediate feedback through computational implementations.
3. To develop problem-solving and critical thinking skills through active experimentation with algorithms and data structures.
4. To demonstrate the real-world applications of Discrete Structures, fostering an appreciation of its relevance in modern technology.

Initial evaluations of the framework have demonstrated its effectiveness in improving student comprehension, engagement, and performance [14]. By transforming traditional teaching methods into a dynamic and interactive experience, this Python-integrated approach not only bridges the gap between theory and practice but also equips students with the computational skills needed to succeed in the field of computer [15].

2. Background

Discrete Structures, a foundational area of computer science and mathematics, includes topics such as logic, set theory, combinatorics, graph theory, and Boolean algebra. These concepts underpin much of the theoretical and practical knowledge required for fields like data science, cryptography, artificial intelligence, and software development. However, the abstract nature of Discrete Structures often makes it challenging for students to grasp and apply its concepts effectively [16]. Discrete Structures play a critical role in developing the logical and analytical thinking skills essential for problem-solving in computer science. Topics like propositional and predicate logic are fundamental to understanding algorithms and programming languages. Similarly, combinatorics and graph theory are integral to optimization problems, network design, and data analysis. Mastery of these topics provides the foundation for advanced studies in algorithms, machine learning, and systems design [17].

Despite their importance, students often perceive Discrete Structures as abstract and disconnected from practical applications. Traditional teaching methods focus heavily on theoretical derivations and manual problem-solving, which can lead to disengagement and a lack of motivation to explore deeper connections between theory and real-world problems [18]. The use of computational tools, particularly programming languages like Python, has emerged as an effective strategy to address the challenges associated with teaching Discrete Structures. Python's intuitive syntax and robust library ecosystem make it an ideal choice for introducing computational thinking and bridging the gap between theoretical concepts and practical applications [19]. Integrating Python into Discrete Structures education aligns with the broader trend of computational pedagogy, which emphasizes hands-on, experiential learning. By implementing algorithms, generating visualizations, and solving real-world problems through code, students develop a deeper understanding of abstract concepts while acquiring practical skills relevant to modern computational areas [20]. Traditional approaches to teaching Discrete Structures often emphasize rote memorization and manual derivation of proofs, which can obscure the relevance of these concepts to contemporary problems in computer science. Students frequently struggle to connect theoretical topics like modular arithmetic or graph traversal with their applications in cryptography, network analysis, or database management [21]. Moreover, the lack of immediate feedback in traditional methods limits students' ability to experiment and iterate on their understanding. Without practical tools to visualize and test their ideas, students may find it difficult to identify and correct misconceptions or apply their knowledge effectively to solve problems [22]. Python's integration into Discrete Structures education offers a powerful means of overcoming these challenges. Its versatility allows educators to design interactive lessons and exercises that not only reinforce theoretical knowledge but also demonstrate its relevance to real-world scenarios. By leveraging Python's computational power and accessibility, educators can bridge the gap between theory and practice, enhancing student engagement, comprehension, and problem-solving skills. This paper builds on this foundation to propose a comprehensive Python-integrated framework that addresses the challenges of traditional teaching methods while emphasizing the real-world applications of Discrete Structures.

3. Proposed framework

The proposed framework for this paper aims to transform the teaching and learning of Discrete Structures (DS) by fully integrating Python as a computational tool. This approach is designed to bridge the gap between abstract theoretical concepts and practical applications while fostering active learning and critical thinking. Below are the core components of the framework:

Table 1: Proposed framework

Discrete Structures Topic	Key DS Concepts	Python Tools	Activities
Logic and Boolean Algebra	Truth tables, logical equivalences, digital circuits	Python conditionals, sympy	Generate truth tables, simulate circuits
Set Theory	Union, intersection, Cartesian products, power sets	set, itertools	Perform set operations, compute power sets
Functions and Relations	One-to-one, onto, reflexivity, symmetry, transitivity	Dictionaries, numpy matrices	Analyze relations, compute matrix closures
Combi-natorics	Counting principles, permutations, combinations	itertools, random	Generate combi-nations, simulate probabi-lities
Graph Theory	Graph representations, BFS, DFS, shortest paths	networkx, matplotlib	Visualize graphs, implement BFS and DFS

3.1 Generate truth tables

```
def truth_table():
    print("P\tQ\tP and Q\tP or Q\tnot P")
    for P in [True, False]:
        for Q in [True, False]:
            print(f"{P}\t{Q}\t{P and Q}\t{P or Q}\t{not P}")

truth_table()
```

3.2 Set operations

```
A = {1, 2, 3}
B = {3, 4, 5}

print("Union:", A | B)
print("Intersection:", A & B)
print("Difference (A - B):", A - B)
```

3.3 Functions

```
def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n - 1)

print(factorial(5))  # Output: 120
```

3.4 Relations

```
import numpy as np

# Matrix representation of a relation
relation = np.array([[1, 0, 1], [0, 1, 0], [1, 0, 1]])

def is_reflexive(matrix):
    for i in range(len(matrix)):
        if matrix[i][i] != 1:
            return False
    return True

print("Is Reflexive?", is_reflexive(relation))
```

3.5 Combinatorics

```
from itertools import permutations, combinations

items = [1, 2, 3]
print("Permutations:", list(permutations(items)))
print("Combinations:", list(combinations(items, 2)))
```

3.6 Graph Theory

```
import networkx as nx

G = nx.Graph()
G.add_edges_from([(1, 2), (2, 3), (3, 4), (4, 1)])
print("Nodes:", G.nodes())
print("Edges:", G.edges())

# Perform BFS
from networkx.algorithms.traversal.breadth_first_search import bfs_edges
print("BFS Traversal:", list(bfs_edges(G, 1)))
```

3.7 Boolean Algebra

```
def simplify_boolean(a, b):
    return not (a and b) == (not a or not b)  # De Morgan's law

print(simplify_boolean(True, False))  # Output: True
```

Note that this mapping is not exhaustive and there may be some variations in the specific topics covered in different courses or curricula. Additionally, some topics in discrete mathematics, such as proof techniques and set theory, may not have direct counterparts in Python but can still be applied in the design and analysis of algorithms and data structures.

4. Analysis and discussion

Python has a rich library of functions that can be used to implement various concepts from Discrete Mathematics. For example, the Python library NetworkX can be used to create, manipulate and study graphs. Python's set data structure and functions can be used to implement set operations such as union, intersection, and difference. Python's itertools library can be used to generate all possible combinations and permutations of a given set. Python's ability to handle large data sets and matrices makes it a useful tool for linear algebra, a branch of Discrete Mathematics that deals with vectors, matrices, and linear transformations. The NumPy library in Python provides functions for manipulating large arrays and matrices, and the SciPy library provides functions for numerical optimization and linear algebra. Python also provides a rich set of functions for probability and statistics, which are essential components of Discrete Mathematics. Discrete Mathematics and Python also share common algorithms, such as sorting algorithms, searching algorithms, and graph algorithms. Python's built-in sorting functions, such as sorted() and sort(), can be used to implement various sorting algorithms, such as bubble sort, insertion sort, and quicksort. Python's built-in searching functions, such as in and index(), can be used to implement various searching algorithms, such as binary search and linear search. Python's NetworkX library provides functions for implementing various graph algorithms, such as Dijkstra's algorithm, Kruskal's algorithm, and Prim's algorithm.

In conclusion, the relationship between Discrete Mathematics and Python is evident, and Python can be used as a powerful tool for implementing concepts and algorithms from Discrete Mathematics. Python's rich set of functions and libraries makes it an excellent choice for implementing various algorithms and structures from Discrete Mathematics. On the other hand, Discrete Mathematics can enhance the understanding of Python by providing a foundation for understanding the mathematical structures that Python can manipulate. As both fields continue to grow and evolve, their relationship will continue to be important for future advancements in computer science and mathematics.

5. Conclusion and future works

The integration of Python into Discrete Structures (DS) education represents a transformative approach to teaching abstract mathematical concepts. By leveraging Python's computational capabilities, educators can bridge the gap between theoretical principles and practical applications, fostering a deeper understanding of DS topics such as logic, set theory, graph theory, and combinatorics. The proposed framework emphasizes interactive learning, real-world applications, and collaborative activities to enhance student engagement and critical thinking skills. Through carefully

designed Python-based exercises, visualizations, and projects, this framework equips students with the computational tools necessary for tackling complex problems in fields such as cryptography, data science, and optimization. The hands-on experience gained through Python not only solidifies theoretical knowledge but also prepares students for interdisciplinary challenges and real-world applications.

To further enhance the effectiveness and scalability of this framework, the following areas will be explored:

- Advanced Topics Integration:

Incorporate advanced DS topics such as automata theory, formal languages, and advanced graph algorithms with Python implementations.

- Adaptive Learning Systems:

Develop an adaptive platform that customizes Python exercises and assessments based on individual student performance and learning pace.

- Assessment Automation:

Integrate AI-driven tools to provide instant feedback on coding assignments, helping students identify and correct errors efficiently.

- Collaborative Learning Environments:

Enhance group-based learning by implementing collaborative platforms like shared coding environments and version-controlled project repositories.

- Longitudinal Studies:

Conduct longitudinal research to measure the long-term impact of Python integration on student success in advanced courses and professional domains.

- Interdisciplinary Applications:

Explore new applications of DS concepts in emerging fields like quantum computing, bioinformatics, and blockchain technology, using Python as a teaching tool.

- Open Educational Resources (OER):

Develop a repository of Python scripts, datasets, and tutorials tailored to DS topics, freely available for educators and students worldwide.

By addressing these areas, the framework can evolve into a comprehensive and adaptable model for teaching Discrete Structures, ensuring its relevance and effectiveness in the ever-changing landscape of computational education.

6. References

1. Stavely Allan M. Programming and Mathematical Thinking: A Gentle Introduction to Discrete Math Featuring Python. New Mexico Tech Press, 2014.
2. Liu Yanhong A, Matthew Castellana. Discrete math with programming: A principled approach. Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, 2021.
3. Moller Faron, Liam O'Reilly. Teaching discrete mathematics to computer science students. Formal Methods Teaching Workshop. Cham: Springer International Publishing, 2019.
4. Farahani Ali. Use of Python in Teaching Discrete Mathematics. 2009 Annual Conference & Exposition, 2009.
5. Zelle John M. Python programming: an introduction to computer science. Franklin, Beedle & Associates, Inc., 2004.
6. Leiva Dante, *et al*. A Novel Approach to Combinatorial Problems: Binary Growth Optimizer Algorithm. Biomimetics. 2024; 9(5):283.
7. Bell Nathan, Anil N Hirani. PyDEC: Software and algorithms for discretization of exterior calculus. ACM Transactions on Mathematical Software (TOMS). 2012; 39(1):1-41.
8. Valiente Gabriel. Algorithms on trees and graphs. Vol. 112. Heidelberg: Springer, 2002.
9. Gopalakrishnan Ganesh, *et al*. Discrete Structures via Circuits, BDD, SAT, SMT, and Functional Programming in Python, 2013.
10. Böhm Stanislav, Jakub Beránek, Martin Šurkovský. Haydi: Rapid Prototyping and Combinatorial Objects. International Symposium on Foundations of Information and Knowledge Systems. Cham: Springer International Publishing, 2018.
11. Çetinkaya-Rundel Mine, Victoria Ellison. A fresh look at introductory data science. Journal of Statistics and Data Science Education. 2021; 29(sup1):S16-S26.
12. Baptista Theuerkauf, Diego Alejandro. Optimal Transport: The Dynamical Monge-Kantorovich Model as a Bridge between Theory and Applications. Diss. Universität Tübingen, 2024.
13. Gervasi Osvaldo, *et al*. Computational Science and Its Applications–ICCSA 2023 Workshops.
14. Aiyenitaju Olaniyi, Sarita Majhi. Math with Python: A Context of Algebra, 2024.
15. Iqbal Muhammad Faisal, *et al*. Investigating Aptitude in Learning Programming Language Using Machine Learning and Natural Language Processing. International Journal of Data Informatics and Intelligent Computing. 2024; 3(4):40-61.
16. Rosen Kenneth H. Discrete Mathematics Applications and Its, 2019.
17. Block Philippe, *et al*. AI-guided generation of reticular structures by integrating reinforcement learning with shape grammars, 2024.
18. Grimaldi Ralph P. Discrete and Combinatorial Mathematics, 5/e. Pearson Education India, 2006.
19. Gu Jiqing, *et al*. Teaching Case Study in Discrete Mathematics: Application of Clustering Algorithms in Brucella Traceability Research. Medinformatics, 2025.
20. Liben-Nowell David. Connecting discrete mathematics and computer science. Cambridge University Press, 2022.
21. Sandefur James, *et al*. Teaching and learning discrete mathematics. ZDM–Mathematics Education. 2022; 54(4):753-775.
22. Duan Zhu, *et al*. The application of information technology in discrete mathematics teaching. 2022 International Conference on Education, Network and Information Technology (ICENIT). IEEE, 2022.