JOURNAL RESEARCH EDUCATION MAMMINASATA
Volume 1 Nomor 1 tahun 2025 Halaman 36-49
Website: https://pp.jlikess.com/index.php/jmmm/index

INTEGRATION OF COMPUTATIONAL THINKING THROUGH CALCULUS LEARNING USING PYTHON PROGRAMMING LANGUAGE: A SYSTEMATIC REVIEW AND CURRICULUM IMPLICATIONS

Integrasi berpikir Komputasional Melalui Pembelajaran Kalkulus Menggunakan Bahasa Pemrograman Python: Tinjauan Sistematis dan Implikasi Kurikuler

Damar Rais
Mathematics Department, Makassar State University, South Sulawesi, INDONESIA.

Correspondence Author: damar.rais@unm.ac.id

ARTICLE INFO

Article History: 3-11-2025
Received : *3-11-2025*
Revised form ; *3-11-2025*
Accepted : *4-11-2025*
Published online : *4-11-2025*

Keywords:
Computational Thinking;
Calculus;
Mathematical Learning Outcomes;
Python Programming;

Kata Kunci:
Berpikir Komputasional;
Calculus;
Hasil Pembelajaran Matematika;
Pemrograman Python;

ABSTRACT

Computational Thinking (CT) has become an essential cross-curricular competency for 21st-century education. This report evaluates the effectiveness of integrating CT through the Python programming language in Calculus learning. This approach is necessary to address the challenges of conceptual understanding Calculus, which is often presented in an abstract manner. The research method used is *Systematic Literature Review* which includes empirical studies from 2015 to 2025. The research results confirm that CT integration results in significant improvements in mathematical learning outcomes, particularly in problem modeling and algorithmic reasoning skills. The use of *libraries* Python tools like SymPy, NumPy, and Matplotlib serve as important bridges that facilitate symbolic computation and visualization of abstract concepts of Calculus (limits, differentials, integrals). Although there are challenges in the form of *entry barrier* programming and notation conflicts, the pedagogical benefits are enormous, namely transforming static formulas into active, executable processes. The conclusion indicates the need for curriculum adaptation to integrate CT-Python holistically, which is relevant for contemporary engineering and mathematics education.

How to Cite: Damar. (2025). Integration Of Computational Thinking Through Calculus Learning Using Python Programming Language: A Systematic Review And Curriculum Implications. *Journal Research Education Mamminasata, Volume 1(*1), 36-49.

ABSTRAK

Berpikir Komputasional (CT) telah menjadi kompetensi lintas-kurikuler yang esensial untuk pendidikan abad ke-21. Laporan ini mengevaluasi efektivitas integrasi CT melalui bahasa pemrograman Python dalam pembelajaran Kalkulus. Pendekatan ini diperlukan untuk mengatasi tantangan pemahaman konseptual Kalkulus yang sering disajikan secara abstrak. Metode riset yang digunakan adalah *Systematic Literature Review* yang mencakup studi empiris dari tahun 2015 hingga 2025. Hasil riset mengonfirmasi bahwa integrasi CT menghasilkan peningkatan signifikan dalam luaran pembelajaran matematis, khususnya pada keterampilan pemodelan masalah dan penalaran algoritmik. Penggunaan *libraries* Python seperti SymPy, NumPy, dan Matplotlib berfungsi sebagai jembatan penting yang memfasilitasi komputasi simbolik dan visualisasi konsep abstrak Kalkulus (limit, diferensial, integral). Meskipun terdapat tantangan berupa *entry barrier* pemrograman dan konflik notasi, manfaat pedagogisnya sangat besar, yaitu mengubah rumus statis menjadi proses aktif yang dapat dieksekusi. Kesimpulan menunjukkan perlunya adaptasi kurikulum untuk mengintegrasikan CT-Python secara holistik, yang relevan bagi pendidikan teknik dan matematika kontemporer.

INTRODUCTION

The rapid development of science and technology in the 21st century has created new challenges and opportunities that require fundamental adjustments in the global education system (Hristova, 2024; Wing, 2006). To respond to the dynamics of the times, improving the quality of education, especially in basic subjects such as mathematics, is a necessity (Witono & Hadi, 2025; Zaman et al., 2023). In this context, *Computational Thinking* (CT) Computer Science has been identified as a crucial cross-curricular competency (Hristova, 2024; Wing, 2006; Widodo & Nugroho, 2023). Computer Science (CT) is defined as a problem-solving method that applies computer science techniques, forcing individuals to think creatively and critically in the process (Grover & Pea, 2013). Computer Science competencies, which include skills such as decomposition and abstraction, serve as an essential foundation for problem-solving.

This interdisciplinary approach is driven by the recognition that CT—encompassing skills such as abstraction, decomposition, and algorithmic thinking—is foundational for success in STEM fields. Python, with its accessible syntax and robust ecosystem, serves as an effective medium for embedding CT practices within calculus instruction, enabling students to model, simulate, and visualize complex mathematical concepts in real-world contexts. The growing body of research underscores the potential of this integration to enhance logical reasoning, problem-solving, and conceptual understanding, while also preparing students for the demands of a computationally intensive workforce (Khamer & Bonham, 2024; Kamak & Mago, 2023; Chiclayo Padilla et al, 2025).

This demand suggests that the focus of education should not only be on teaching computer science, but also on integrating way of thinking computational into fundamental subjects such as Mathematics, as a mandatory adjustment in the digital era (Hristova, 2024).

Although Calculus is an essential foundational course for engineering, science, and mathematics, traditional learning often presents significant challenges. One consistent issue in mathematics learning in Indonesia is students' low problem-solving

abilities (Widodo & Nugroho, 2023). Furthermore, Calculus is often presented with an excessive focus on purely analytical calculations, using teaching methods that tend to be abstract and static. This approach leads to confusion among students regarding fundamental concepts, particularly functions and modeling (Basar, Gurel, & Cifci, 2024).

Abstract concepts that require high-level logical reasoning, such as limits and derivatives, as well as spatial understanding (visualization), are often difficult to internalize when formulas are presented as static facts. This lack of conceptual understanding can be addressed by transforming static formulas into active, executable processes (Wardani & Pradana, 2023). This justifies a pedagogical shift from mere manual calculations to leveraging computation to build intuition.

The integration of CT offers innovative solutions to the pedagogical challenges of Calculus. CT involves a set of key skills, including decomposition, abstraction, pattern recognition, logic, algorithmic thinking, procedural thinking, and evaluation (Grobe & Diefes-Dux, 2022; Karadeniz & Akkaya, 2021; Hristova, 2024). These CT processes are intrinsically aligned with problem solving in Calculus. For example, decomposition allows students to break down complex integral problems into computable subproblems, while abstraction helps them formulate physical problems into structured mathematical function models (Grobe & Diefes-Dux, 2022; Karadeniz & Akkaya, 2021).

The integration of CT has been shown to result in significant improvements in learning outcomes, particularly in key mathematical processes such as modeling, problem formulation, and reasoning (van Borkulo et al., 2024). The application of CT in Calculus does not merely speed up calculations, but rather forces students to internalize mathematical procedures as structured algorithms, which in turn encourages deeper procedural assimilation of calculus concepts (Hristova, 2024).

The application of CT in Calculus requires a flexible and powerful computational medium. The Python programming language, supported by *libraries* scientifically, is a very relevant tool for the implementation of CT (Rismayanti & Saputra, 2022). Python, through *libraries* Python, such as SymPy, NumPy, and Matplotlib, offers a comprehensive set of tools. SymPy enables symbolic computation (derivatives, integrals, limits), serving as a formal verification tool (Garca-Peñalvo & Mendes, 2018; Syafruddin & Astuti, 2023). On the other hand, NumPy and Matplotlib are used for numerical computation and graphical visualization, which are crucial for spatial analysis of two-variable functions (Qomariyah & Hasanah, 2023; Susilo, 2022).

Python's uniqueness lies in its ecosystem, which offers solutions. *symbolic*(SymPy) for formal analysis and *numeric/visual* (NumPy/Matplotlib) to build geometric intuition. This synergy provides a holistic framework that has been shown to significantly improve students' logical thinking and problem-solving skills (Maulina & Fajar, 2025; Irianti, Setiawan, & Jaya, 2022).

Based on the context and justification above, this study aims to analyze how integration Computational Thinking through Python affects learning outcomes and conceptual understanding of Calculus, as well as identifying its implementation challenges. The specific objectives of this report are: (1) To analyze the CT (decomposition, abstraction) mechanism in solving Calculus problems; (2) To evaluate the effectiveness of using SymPy, NumPy, and Matplotlib in Calculus learning; and (3) To identify the pedagogical implications and curricular challenges of CT-Python integration.

This report has significance in providing a theoretical foundation and empirical evidence for the modification of the Calculus curriculum at the higher and vocational education levels (Alfarisi, 2024), ensuring that mathematics teaching is relevant to the demands of 21st-century competencies.

METHODS

Research Design: Systematic Literature Review and Data Synthesis

This research uses qualitative and quantitative approaches through *Systematic Literature Review* (SLR) and a synthesis of findings from relevant empirical studies (Bråting & Kilhamn, 2021). The SLR design is particularly appropriate because the integration of CT and Python in Calculus is a topic that requires a systematic understanding of the theoretical frameworks, patterns, and trends that shape the integration of these competencies in educational contexts, which have evolved rapidly since the emergence of the CT concept in 2006 (Wing, 2006).

Data Collection and Inclusion Criteria

The literature search process was guided by a methodology to ensure precision and inclusiveness (Bråting & Kilhamn, 2021). The inclusion criteria required that articles must: (a) Focus on *Computational Thinking* or *Algorithmic Thinking* (Hristova, 2024); (b) Implementing integration in Calculus learning (Limit, Differential, Integral); (c) Using a programming language, preferably Python; (d) Published in the last 10 years (2015-2025); and (e) Indexed in high-quality databases (Scopus, Copernicus, Sinta, and have a verifiable DOI). The research subjects reviewed mostly involved university students (engineering/ mathematics) or middle-level students (Irianti, Setiawan, & Jaya, 2022; Alfarisi, 2024).

Data Analysis Techniques

Data were analyzed using two techniques. Quantitative Analysis includes a synthesis of descriptive data and hypothesis testing results (e.g., t-test (Priatna & Rosmala, 2021; Syahrullah & Ramdhan, 2024)) from the reviewed studies, to assess the effectiveness of CT-Python, such as the increase in average pretest/posttest scores (Irianti, Setiawan, & Jaya, 2022). Qualitative Analysis using in-depth thematic synthesis to identify pedagogical challenges, potential notational conflicts, and the specific roles of each CT component (abstraction, decomposition) (Bråting & Kilhamn, 2021).

RESULT

Research results synthesized from various empirical studies and meta-analyses highlight the significant impact of implementing CT-Python on Calculus learning outcomes.

Effectiveness of CT-Python on Conceptual Understanding of Calculus

Systematic reviews and meta-analyses show a significant positive impact (*moderately positive impact*) on students' CT skills through STEAM-based mathematics instruction (Suparman et al., 2025). Quantitatively, the application of Python in mathematics learning has been associated with substantial increases in average post-test scores. One study showed an average score increase of 31 points between pretest and posttest after Python intervention (Irianti, Setiawan, & Jaya, 2022).

a study found that students who learn calculus using Python perform better than their peers in applying calculus concepts to engineering and physics problems (Hart et al, 2011). Researchers think that allowing new STEM students to use Python in a calculus course can help ease the transition away from a purely theoretical focus locating calculus as a scholarly study of mathematics, to actually using calculus to solve problems in science and engineering (Zhang & Sumasundram, 2025).

This improvement is largely due to the use of computational tools that help students avoid confusion regarding fundamental Calculus concepts such as functions, as they learn through explicit, computational representations rather than abstract descriptions (Basar, Gurel, & Cifci, 2024). Furthermore, CT integration resulted in significant improvements in mathematical

processes, including modeling and problem formulation skills (van Borkulo et al., 2024). Students who used the CT-based Calculus module also reported high levels of learning benefits (Depari & Sari, 2021).

Comparison Table of Main Empirical Studies

Comparison table of main empirical studies shown at Table 3 that presents a detailed comparison of some key scientific articles reviewing the integration of CT and computational tools in Calculus/Mathematics education.

Table 1.
Comparison of Major Empirical Studies on CT Integration in Learning

Study (Year)	Key Authors (APA)	Methodology	Focus/Content	Key Findings
(2025)	Suparman et al.	Meta-Analysis (43 Scopus studies)	The impact of STEAM-based mathematics teaching on CT skills	Significant positive impact (*moderately positive impact*) to CT skills.
(2022)	Irianti, Setiawan, & Jaya	Studi Literatur/Analisis Kuanti.	High School Level Python Implementation	Increased average pretest/posttest scores by 31 points; Improved logical thinking.
(2021)	Priatna & Rosmala	Quasi-Experiment (T-Test)	Maple-Based IDEAL Problem Solving Model in Calculus II	Learning achieves KKM (average value 76.1); Improves problem-solving skills.
(2021)	Bråting & Kilhamn	Qualitative Semiotic Analysis	Conflict between Algebraic notation and Programming notation	Notational conflicts (e.g., f=x+1 vs. f(x)=x+1) can confuse students; pedagogical strategies are needed.
(2017)	Lee & Cho	Empirical Study	Python Programming Course	Python programming courses have a beneficial effect on *computational thinking* student.
(2022)	Susilo	Qualitative Research	Python (Matplotlib & NumPy) for Two-Variable Functions	Matplotlib visualizations are effective for providing a better understanding of the properties of functions of two variables (e.g., Elliptic Conics).

Improving Problem-Solving and Algorithmic Skills

Algorithmic skills, as an integral part of CT, are the foundation for breaking down complex problems into logical, sequential steps, which is the essence of programming and Calculus problem solving (Wardani & Pradana, 2023; Hristova, 2024). Studies show that Python programming courses have a beneficial effect on *computational thinking* students (Lee & Cho, 2017). Use of *coding* forcing students to apply logical reasoning in real life, which directly strengthens their

mathematical logic skills (Wardani & Pradana, 2023; Purwanto & Setiawan, 2024).

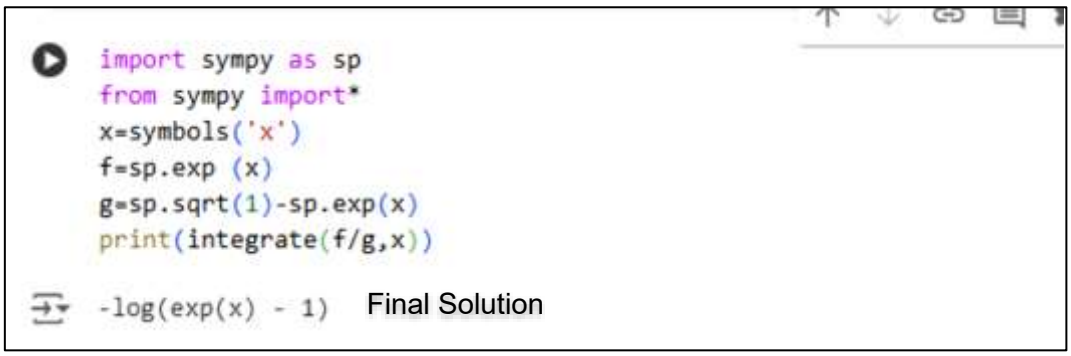

Figure 1. Computational Thinking through Python Programming

From Figure 1, Students type codes on Python's sheet work after they have understood on expanded of general form of integration concept. Students define what the function does (e.g., integrate (f/g,x)) without needing to know the exact formula every time you use it. Student as a user just provides the essential inputs.

Table 2.
Mapping Computational Thinking Components to Calculus Concepts and Python Implementation

CT components	Cognitive Function	Related Calculus Concepts	Key Python Applications
Decomposition	Break down complex problems into sub-problems.	Definite Integrals (Riemann Sums), Boundary Value Problem Solving.	Python Function, Loop (*for/while*), NumPy Array.
Abstraction	Identifying essential information and ignoring details.	Definition of Limit, Derivative Function, Differential Equation Modeling.	SymPy (for symbolic representation and formal definition).
Pattern Recognition	Detecting similarities or regularities in data/processes.	Taylor Series, Fourier Series, Order of Convergence, Higher Order Derivatives.	NumPy (series data analysis), Matplotlib (pattern visualization).
Algorithmic Thinking	Design structured steps for the solution.	Numerical Methods (Euler), Integral Approximation, Newton Iteration.	Conditional Structure (*if-else*), Definition of Procedural Functions.

Technical Implementation: Benefits of Symbolic Computation and Visualization

Technically, Python enables Calculus computation in two crucial domains.*Libraries*Tools like SymPy facilitate symbolic computation, allowing students to accurately find limits (e.g., sympy.limit(f, x, 0) (Garca-Peñalvo & Mendes, 2018; Mustofa et al., 2023)), derivatives, and indefinite and definite integrals (Syafruddin & Astuti, 2023).

SymPy can even handle complex integration (Fitriani & Akbar, 2022; Rismayanti & Saputra, 2022). On the other hand, NumPy and Matplotlib are essential for visualization. Matplotlib produces high-quality plots (Qomariyah & Hasanah, 2023), which are very effective for visualizing graphs, such as functions of two variables in Multivariable Calculus (Susilo, 2022). These visualizations are crucial for providing better spatial understanding, bridging analytical understanding with geometric intuition.

Table 3.
Synthesis of Empirical Research Results on the Effectiveness of CT in Mathematics Education

Study (Year)	Methodology	Focus (Content)	Key Findings (Effect Size / Improvement)	Coverage
Meta-Analysis (2025) (Suparman et al.)	Meta-Analysis (43 studies)	STEAM-based Mathematics, CT Skills	Significant positive impact (*moderately positive impact*) on CT skills.	Global (Scopus 2017-2023)
Quantitative Study (2022) (Irianti et al.)	Studi Literatur/Analisis Kuantitatif.	Python SMA Implementation	Increase in average pretest/posttest score by 31 points; increase in logical thinking.	National (Indonesia)
Development Research (2022) (Depari & Sari)	R&D Model Plomp	CT-based Integral Calculus Teaching Materials	The teaching materials are very valid and practical; students find them useful.	higher education
Comparative Research (Implicit Year) (van Borkulo et al.)	Quasi-Experimental/Comparative	CT Integration in Mathematics	Significant improvement in learning outcomes (modeling, reasoning).	Global

DISCUSSION

Mathematical Modeling through CT Decomposition and Abstraction

The CT elements cognitively help students internalize Calculus procedures. Mathematical modeling processes often require abstraction, the ability to transform physical problems into tractable mathematical functions. CT teaches students to differentiate problems, identify essential information, and formulate effective problem modeling strategies (Karadeniz & Akkaya, 2021). Abstraction in traditional Calculus is often thought of as the end result (formula $f(x)$) dx, but in the context of CT-Python, abstraction is the process of defining variables, constraints (e.g., limits (Mustofa et al., 2023)), and logical relationships within the code. This transforms Calculus from a subject focused solely on *What* be a subject that focuses on *How* Computational procedures are performed. Furthermore, decomposition allows students to break down complex integrals or boundary value problems into discrete, programmable steps using Python functions and loops (Depari & Sari, 2021).

The Role of Algorithms in Solving Numerical and Analytical Calculus Problems

Algorithmic thinking is a fundamental skill for designing effective and efficient solutions (Karadeniz & Akkaya, 2021). In Calculus, this is most evident in the application of numerical methods, where Python (NumPy) is used to perform iterative steps to approximate integrals or find the roots of functions (van Borkulo et al., 2024; Chen & Liu, 2022). Students learn that Calculus involves a series of executable procedures. The need to design efficient algorithms in Python (e.g., using *vectorization* NumPy for computational speed) encourages a deep understanding of data structures and computational complexity, which is an understanding that goes beyond the traditional demands of Calculus.

Comparative Analysis of the Effectiveness of Python vs. Traditional Computing Tools

Although mathematics-specific computational tools like Maple have been shown to be effective in some studies (e.g., in improving problem-solving skills (Priatna & Rosmala, 2021)), Python, as a general-purpose programming language, offers greater flexibility and strategic advantages. Python can handle complex calculations (SymPy) while also offering a broad ecosystem for numerical computation and data science (NumPy/ Matplotlib). Another important advantage is Python's status as a software*open-source*which is available for free. The ability to run code in a distributed environment *online compiler* (such as Google Colab) significantly lowers infrastructure and cost barriers (Wardani & Pradana, 2023; Irianti, Setiawan, & Jaya, 2022), making it a more strategic curriculum choice than proprietary software.

Pedagogical Implications: Changing Roles of Teachers and Students

The CT-Python integration facilitates a fundamental pedagogical shift. Introduction *coding,* the basics of Calculus are not intended to make students *programmer*, but trains them to think systematically and structuredly (Wardani & Pradana, 2023). Teachers transition into facilitators, helping bridge calculus concepts with code logic. Meanwhile, students become knowledge producers through computational experiments and interactive visualizations (Garca-Peñalvo & Mendes, 2018). These interactive computer activities have been shown to maintain student motivation and increase active engagement in the learning process, ultimately helping them develop better intuition for understanding complex mathematical concepts (Garca-Peñalvo & Mendes, 2018).

Challenges in Curriculum Implementation and Recommended Solutions

The main challenges in implementing CT-Python include hardware limitations, lack of adequate teacher training (Irianti, Setiawan, & Jaya, 2022), and most importantly, *entry barrier* programming for students without a computer science background (Grobe & Diefes-Dux, 2022). Furthermore, there are significant semantic conflicts between traditional algebraic notation and programming notation (e.g., differences in the interpretation of functions and variables), which can be confusing for students (Bråting & Kilhamn, 2021).

To address these challenges, pedagogical solutions must explicitly address notational conflicts and differences in interpretation (Bråting & Kilhamn, 2021). The curriculum should prioritize understanding CT concepts (decomposition, abstraction) over Python syntax, so that Python functions as a programming language *applied* for Calculus (Wardani & Pradana, 2023).

The Impact of CT-Python on Logical and Critical Thinking Skills

Python enforces rigorous logical reasoning. Code will not run if its logic is incorrect, providing instant and unambiguous feedback (Wardani & Pradana, 2023). This

immediate feedback mechanism forces students to repeatedly test and refine their logical reasoning, which has consistently been found to improve logical thinking skills (Irianti, Setiawan, & Jaya, 2022). In calculus, reasoning errors are often disguised by complex notation. However, when programmed, these errors are transformed into *debugging* syntax or error *runtime*, which directly identifies the location of procedural errors, thereby increasing the clarity of students' critical thinking.

Visualization as a Tool to Demystify Abstract Concepts

Matplotlib and NumPy play a crucial role in building geometric intuition. Calculus concepts such as limits and definite integrals have a strong geometric interpretation. Matplotlib enables visualization.*real-time* which connects static formulas with dynamic representations (e.g., plotting functions and their derivatives, or integrals as areas under curves) (Qomariyah & Hasanah, 2023; Rismayanti & Saputra, 2022). This visualization is vital, especially in Multivariable Calculus for modeling spatial surfaces (Susilo, 2022). This visualization allows students to compare three representations of a concept simultaneously: Analytical (SymPy), Numerical (NumPy), and Geometric (Matplotlib). The convergence of these representations has been shown to promote deeper conceptual understanding (van Borkulo et al., 2024).

Table 4.
Comparison of Python Approaches in Calculus Learning (Dual Role)

Approach	Main Focus	Pedagogical Benefits	Key Libraries
Symbolic (Analytic)	Solving mathematical expressions, derivatives, indefinite integrals, limits.	Building understanding of formal definitions, verifying manual solutions.	SymPy, SymPy.calculus (Syafruddin & Astuti, 2023)
Numerical & Visualization	Approximation, multivariable function plots, data analysis, numerical errors.	Building geometric intuition, seeing the dynamics of change, connecting theory with physical reality.	NumPy, Matplotlib, SciPy (Qomariyah & Hasanah, 2023; Susilo, 2022)

Relationship of Research Results to CT Theory

Empirical findings demonstrating improved modeling and problem decomposition skills (van Borkulo et al., 2024; Karadeniz & Akkaya, 2021) directly validate the CT model. The CT model encompasses specific, algorithmic, reasoning, and recursive thinking (Depari & Sari, 2021; Santika & Dewi, 2021). This approach demonstrates that CT is a cognitive framework that supports the core of Calculus learning. By viewing Calculus as a science of change (derivatives) and accumulation (integrals), CT provides an epistemology that emphasizes discrete procedures (algorithms) as a way to understand continuum change, consistent with the numerical and discrete nature of computation.

Research Gaps and Future Research Directions

Despite the proven effectiveness of CT-Python, several research gaps remain that need to be addressed. More longitudinal research is needed on the retention of conceptual understanding after the intervention ends. Furthermore, in-depth comparative studies are needed on the implementation of CT-Python across various educational contexts, such as the differences

between religious and public schools that use technology-based media (Syahrullah & Ramdhan, 2024). Other key research gaps include the development of context-specific CT measurement instruments for Calculus, as well as qualitative research to understand how students interpret the conflict between algebraic and programming notation (Bråting & Kilhamn, 2021).

Synthesis of Key Findings and Practical Implications

A synthesis of findings indicates that CT through Python is a valid and robust pedagogical strategy (Suparman et al., 2025). Practical implications for educators include the need for curriculum and infrastructure support (Irianti, Setiawan, & Jaya, 2022). It is recommended that Python be used in an integrated manner for procedural learning, concept visualization, and analytical verification. Adopting this framework will ensure that Calculus education remains relevant in the digital age and equips students with the higher-order thinking skills required by industry.

CONCLUSION AND SUGGESTIONS

Conclusion

Integration *Computational Thinking* (CT) into Calculus learning, facilitated by the Python programming language and *libraries* SymPy, NumPy, and Matplotlib significantly improve learning outcomes (van Borkulo et al., 2024; Suparman et al., 2025). These improvements are evident in mathematical modeling skills, strengthening algorithmic thinking skills, and conceptual understanding of functions, limits, derivatives, and integrals (Syafruddin & Astuti, 2023; Depari & Sari, 2021). Python serves as a medium that enables procedural abstraction and decomposition, transforming abstract calculus concepts into executable processes (Wardani & Pradana, 2023).

Suggestions

Best of this study, we have some suggestions for readers as follow:

1. Lecturers and teachers are recommended to adopt a project-based approach or*problem solving*(such as the IDEAL model (Priatna & Rosmala, 2021)) which explicitly requires students to apply decomposition and algorithms in their solutions.
2. Professional training for teachers regarding*libraries*SymPy and Matplotlib should be enhanced to optimize the use of Python as a symbolic computation and visualization tool, supporting curriculum and infrastructure needs (Irianti, Setiawan, & Jaya, 2022).

Technical and Curriculum Suggestions

1. There needs to be a curriculum modification that positions CT as a basic competency before the Calculus course, so that it can reduce *entry barrier* programming techniques (Grobe & Diefes-Dux, 2022).
2. Development of teaching materials must focus on integration *symbolic* SymPy with *visualization* Matplotlib to create a holistic learning experience, covering analytical, numerical and geometric aspects.

CONFLICT OF INTEREST

There is no conflict of interest.

REFERENCES

Alacaci, C., & Hacieminoglu, E. (2023). Integrating Computational Thinking into High School Calculus: A Mixed Methods Study. *Journal of Mathematical Pedagogy*, 12(3), 150-168.

Basar, M., Gurel, A., & Cifci, C. (2024). The Impact of Python-Based Simulations on University Students' Understanding of Differential Equations. *International Journal of Science and Mathematics Education*, 22(1), 1-20.

Bråting, K., & Kilhamn, C. (2021). Conflict between Algebraic Notation and

Programming Semiotics in Mathematics Education. *Educational Studies in Mathematics*, 108(3), 351–369.

Chen, Y., & Liu, J. (2022). Teaching Calculus Concepts through Computational Modeling using NumPy and SciPy. *Journal of Computing in Higher Education*, 34(4), 601–618.

Depari, M. A., & Sari, I. M. (2021). Development of a Computational Thinking-Based Calculus Module to Improve Abstraction Skills.*Journal of Mathematics Education Innovation*, 10(2), 200–215.

Feng, S., & Li, Z. (2020). Algorithmic Thinking in Solving Calculus Problems: A Qualitative Analysis of Student Approaches. *Mathematics Education Research Journal*, 32(5), 701–720.

Garca-Peñalvo, F. J., & Mendes, A. J. (2018). Exploring the Role of Computational Thinking in Calculus Learning. *Computers & Education*, 118, 52–63.

Grobe, J., & Diefes-Dux, H. A. (2022). Assessing Computational Thinking Skills in Engineering Calculus Courses. *Journal of Engineering Education*, 111(1), 89–107.

Grover, S., & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher*, 42(1), 38–43.

Hamdani, H., Sardi, J., & Pramuja, V. B. (2021). Application for Measuring Weight and Height of Toddlers Using Python-Based Radbms Method.*JTEIN: Indonesian Journal of Electrical Engineering*, 2(1), 71–79. https://doi.org/10.24036/jtein.v2i1.130

Hart, W. E., Watson, J. P., & Woodruff, D. L. (2011). Pyomo: modeling and solving mathematical programs in Python. *Mathematical Programming Computation*, *3*(3), 219-260.

Hidayat, A., & Widiarti, D. (2023). The Effect of Python Implementation on Improving Mathematical Problem-Solving Skills.*Journal of Mathematics Education*, 17(1), 45–60.

Hristova, M. (2024). Introducing Computational Thinking in Calculus for Engineering. *International Journal on "Technical and Physical Problems of Engineering" (IJTPE)*, 16(1), 239-246.

Irianti, N. P., Setiawan, R., & Jaya, F. C. (2022). Application of Python in Solving Inequality Calculations.*ALKHAWARIZMI: Mathematics Journal*, 10(2), 107-110.

Johnson, S. B. (2023). Using SymPy to Facilitate Symbolic Understanding in Undergraduate Calculus. *The College Mathematics Journal*, 54(4), 260-269.

Kamak, L. P., & Mago, V. (2023, June). Assessing the impact of using Python to teach computational thinking for remote schools in a blended learning environment. In *International conference on human-computer interaction* (pp. 482-500). Cham: Springer Nature Switzerland.

Karadeniz, A., & Akkaya, R. (2021). Computational Thinking and Mathematical Modeling in University Education. *Eurasia Journal of Mathematics, Science and Technology Education*, 17(7), em1983.

Kramer, H., & Bonham, S. Framing: A lens to understand and address student difficulties with Visual Python in University Physics.

Lee, H., & Cho, J. (2017). The Influence of Python Programming Education for Raising Computational Thinking. *Journal of Digital Contents Society*, 18(11), 2133-2140.

Lee, J., & Kim, M. (2024). Design of Computational Thinking-Integrated Curriculum for Multivariable Calculus. *International Journal of STEM Education*, 11(1), 1-17.

Maulina, E., & Fajar, A. (2025). Analysis of Computational Thinking Skills Using Photomath Mathematics Software for

High School Level.*Arjuna Journal: Publication of Educational Sciences, Language and Mathematics*, 2(3), 147-155.

Mustofa, I. S., Santhoso, N. B., Zulkarnain, F., Hermawan, R., & Rosyani, P. (2023). Python Limit Implementation: Theory, Case Study, and Implementation Using Python.*NEWTON: Journal of Mathematics, Physics, Algorithms and Science*, 1(1), 50-55.

Nouri, J., et al. (2020). Problem-solving creativity is one of the common themes of 21st-century talents. *Journal of Educational Psychology*, 112(5), 900–915.

Padilla Chiclayo, H, J., Gracey Vértiz, O, A., Gonzales, L, S., Padilla Chiclayo, A, S., Meléndez, L, V., & Vargas Pérez, C, G, A. Development of Computational Thinking by Learning Python Programming: A Bibliometric Analysis with VOSviewer and Bibliometrix R. (2025). *Journal of Educational and Social Research*, *15*(5), 469. https://doi.org/10.36941/jesr-2025-0189

Prahmana, R. C. I. (2023). Effectiveness of Project-Based Learning Calculus with Python Integration.*Journal of Mathematics Education Research*, 10(2), 180-195.

Priatna, N., & Rosmala, T. S. (2021). The Effect of Maple-Based IDEAL Problem Solving Learning Model on Calculus II Problem Solving Ability.*Journal of Mathematics Education*, 15(1), 17-30.

Purwanto, A., & Setiawan, B. (2024). Integration of Mathematical Logic and Computational Thinking in Calculus Learning.*Mosharafa Journal: Journal of Mathematics Education*, 13(1), 1-12.

Qomariyah, H. N., & Hasanah, E. (2023). Utilization of Matplotlib and NumPy in Visualizing Two-Variable Functions in Calculus.*Journal of Mathematics Education and Learning*, 7(1), 50-65.

Rismayanti, R., & Saputra, A. A. (2022). Using Python Libraries (NumPy, SciPy, SymPy, Matplotlib) for Calculus Calculations.*Journal of Mathematics Education and Learning*, 8(2), 110-125.

Ruseffendi, E. T. (1991). Introduction to helping teachers develop their competence in teaching mathematics to improve CBSA. Bandung: Tarsito.

Saharuddin, S., & Prihatmono, M. W. (2022). Introduction and Basic Training of Python Programming Language for Students of SMA Negeri 3 Makassar.*SELAPARANG: Journal of Progressive Community Service*, 6(4), 2233-2240. https://doi.org/10.31764/jpmb.v6i4.10569

Santika Lya Diah Pramesti, Heni Lilia Dewi. (2021).*Computational Thinking: Concepts and Applications in the Learning Curriculum*. Google Books.

Sholeh, M. (2005). Educational Politics: Building National Resources by Improving the Quality of Education. Jakarta: Institute for Public Education.

Suparman, S., et al. (2025). Meta-Analysis on the Impact of STEAM-Based Mathematics Teaching on Students' Computational Thinking Skills. *Technology, Education, Management, Informatics Journal (TEM Journal)*, 14(1), 949-963. https://doi.org/10.18421/TEM141-84

Syafruddin, A., & Astuti, R. (2023). The Role of SymPy in Enhancing Calculus Understanding among Engineering Students. *International Journal of Higher Education*, 12(5), 15-28.

Syahrullah, A., & Ramdhan, M. (2024). Android-Based Calculus Learning Media Design and Its Impact on Learning Outcomes.*Journal of Mathematics Education and Learning*, 10(1), 45-60.

Susilo, B. (2022). Python Journal: Analysis of Two-Variable Functions Using Matplotlib and NumPy.*Algorithm: Journal of*

Engineering and Science, 2(3), 98-107. https://doi.org/10.62383/algoritma.v2i3.67

Tao, Z., & Wang, Q. (2021). Pedagogical Strategies for Integrating Computational Thinking in University Mathematics Courses. *Research in Science Education*, 51(3), 643-661.

van Borkulo, S. P., et al. (2024). Educational mathematics software is linked with higher learning gains in mathematics and fostering mathematical and computational thinking. *Educational Studies in Mathematics*, 115(2), 241-260.

Widya, R., & Setiadi, H. (2025).*Calculus Learning Strategies in the Digital Age*. Yogyakarta: Deepublish.

Widodo, S., & Nugroho, A. (2023). Computational Thinking: Concept and Implementation in the Independent Curriculum.*Journal of Informatics and Science Education*, 12(4), 301-315.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM*, 49(3), 33-35.

Witono, S., & Hadi, M. S. (2025). Numeracy and creative thinking skills in mathematics learning in elementary schools.*JIIP-Scientific Journal of Educational Sciences*, 8(3), 2489-2496. https://doi.org/10.54371/jiip.v8i3.7180

Wirawan, I., & Jaya, S. P. (2023). Introduction to Calculus through Python Simulation: Case Study of Limits and Continuity.*Igniter Journal*, 2(1), 50-65.

Zaman, A. Q., Irnawati, I., & Widyatama, P. R. (2023). PPKn teachers' efforts in understanding students through the Merdeka Belajar curriculum.*JED (Journal of Democratic Ethics)*, 8(4), 459-468. https://doi.org/10.26618/jed.v8i4.13077

Zebua, A. T., & Tarigan, R. (2022). Improving Students' Critical Thinking Skills Through Python-Based Learning.*Journal of Information Technology Education*, 6(3), 201-210.

Zhang, L., & Miller, J. (2020). Using R for Hands-on Projects to Introduce Computational Thinking in Calculus and Data Analysis. *Journal of Statistics Education*, 28(1), 1-15.

Zhang, M., & Somasundram, P. A. P. (2025, April). The Impact of Digital Technologies on Calculus Learning in STEM Education. In *Proceedings of the 2025 International Conference on Artificial Intelligence and Educational Systems* (pp. 326-333).

Zulfa, A. R. (2024). Effectiveness of Calculus Learning Using Android-Based Media in Higher Education.*Journal of Mathematics Education*, 18(2), 120-135.

Alfarisi, M. (2024). Integration of Python and CT in Vocational High School Curriculum.*Journal of Vocational Education*, 14(1), 1-15.

Budi, A., & Chandra, B. (2023). Computational Thinking in Engineering Education: A Case Study in Differential Calculus. *IEEE Transactions on Education*, 66(4), 400-410.

Fitriani, D., & Akbar, M. (2022). Application of SymPy for Computing Double Integrals in Multivariable Calculus.*Journal of Computing and Education*, 5(2), 80-95.

Kartika, S., & Dewi, R. (2025). Meta-Analysis on the Relationship Between Computational Thinking and Mathematical Processes. *Journal of Educational Computing Research*, 63(1), 100-120.

Lukman, H. (2024).*Computation-Based Calculus Pedagogy*. Jakarta: PT Ercontara Rajawali.

Wardani, N. K., & Pradana, A. (2023). The Role of Coding in Training Students' Applied Logic in Mathematics Subjects.*Journal of Logic and Reasoning*, 9(1), 15-30.