Sociedade Brasileira de Educação Matemática

Between Facilities and Learning: Connecting Mathematical Skills Through Python Programming

Jailson França dos Santos
Universidade Federal do Oeste da Bahia
Barreiras, BA — Brasil
jailson.santos@ufob.edu.br
0000-0002-9847-4081

Ilton Ferreira de Menezes
Universidade Federal do Oeste da Bahia
Barreiras, BA — Brasil
Ilton.menezes@ufob.edu.br
0000-0002-9590-6731

2238-0345

10.37001/ripem.v15i3.4528

Received • 24/03/2025
Approved • 25/06/2025
Published • 01/09/2025

Editor • Gilberto Januario

Abstract: The use of digital technologies has transformed mathematics education, modernizing teaching practices and aligning them with active methodologies. In Brazil, the Ministry of Education has encouraged the introduction of these innovations, including the teaching of programming languages as a component of the curriculum. Among the languages available, *Python* stands out, offering several libraries focused on solving mathematical problems. This theoretical paper provides a critical analysis of the use of these libraries, discussing the pedagogical benefits and challenges involved. The methodology is based on a theoretical analysis supported by illustrative examples, which show how the practicality of *Python* can both favor and limit the development of mathematical and computational thinking. It is hoped that the research will contribute to the discussion on the inclusion of programming in mathematics education, promoting a more interactive environment aligned with active methodologies.

Keywords: Digital Technologies. Programming Language. Computational and Mathematical Thinking. Active Methodologies.

Entre las instalaciones y el aprendizaje: conectando habilidades matemáticas mediante la programación en Python

Resumen: El uso de tecnologías digitales ha transformado la enseñanza de las matemáticas, modernizando las prácticas pedagógicas y alineándolas con metodologías activas. En Brasil, el Ministerio de Educación ha incentivado la inclusión de estas innovaciones, incluyendo la enseñanza de lenguajes de programación como un componente curricular. Entre los lenguajes disponibles, destaca *Python*, que ofrece varias bibliotecas orientadas a la resolución de problemas matemáticos. Este artículo teórico realiza un análisis crítico del uso de estas bibliotecas, discutiendo los beneficios y los desafíos pedagógicos involucrados. La metodología se basa en un análisis teórico respaldado por ejemplos ilustrativos, que muestran cómo la practicidad de *Python* puede tanto favorecer como limitar el desarrollo del pensamiento matemático y computacional. Se espera que la investigación contribuya a la discusión sobre la inclusión de la programación en la enseñanza de las matemáticas, promoviendo un entorno más interactivo alineado con las metodologías activas.

Palabras clave: Tecnologías Digitales. Lenguaje de Programación. Pensamiento Computacional y Matemático. Metodologías Activas.

Entre Facilidades e Aprendizagens: Conectando Habilidades Matemáticas Através da Programação em Python

Resumo: O uso de tecnologias digitais tem transformado o ensino de Matemática, modernizando práticas pedagógicas e alinhando-se às metodologias ativas. No Brasil, o Ministério da Educação tem incentivado a inserção dessas inovações, incluindo como componente curricular o ensino de linguagem de Programação. Entre as linguagens disponíveis, destaca-se o *Python*, que oferece diversas bibliotecas voltadas à resolução de problemas matemáticos. Este artigo, de caráter teórico, realiza uma análise crítica sobre o uso dessas bibliotecas, discutindo os benefícios e desafios pedagógicos envolvidos. A metodologia baseia-se em uma análise teórica com apoio de exemplos ilustrativos, que evidenciam como a praticidade do *Python* pode tanto favorecer quanto limitar o desenvolvimento do pensamento matemático e computacional. Espera-se que a pesquisa contribua para a discussão sobre a inserção da Programação no ensino de Matemática, promovendo um ambiente mais interativo e alinhado às metodologias ativas.

Palavras-chave: Tecnologias Digitais. Linguagem de Programação. Pensamento Computacional e Matemático. Metodologias Ativas.

1 Introduction

The advancement of Information and Communication Technologies (ICT) has driven the development of digital tools that, when integrated with Digital Information and Communication Technologies (DICT), have positively transformed the educational environment, especially when associated with active methodologies (Morán, 2015). In mathematics education, historically seen as a challenging and often feared subject, these technologies have played a crucial role in modernizing teaching practices.

The appropriate use of TDIC has enabled the adaptation of pedagogical practices to educational demands, in line with the normative document of the Brazilian Ministry of Education, the National Common Curriculum Base [BNCC] (Brasil, 2018). This integration enables more dynamic and interactive teaching that is connected to the technological realities of students, promoting more meaningful and engaging learning, according to Papert (1980) and Geldreich and Hubwieser (2020). In addition, the strategic use of TDIC encourages the application of mathematical concepts in real contexts, strengthening the development of skills such as critical thinking, problem solving, and creativity (Siqueira, 2022; Burton *et al.*, 1991). Thus, digital technologies not only broaden educational horizons but also favor a more contextualized and inclusive education.

Given the contributions that TDICs have made to teaching and learning (Bacich & Moran, 2018; Selwyn, 2021), consciously incorporating them into the classroom can be a natural step, especially with regard to computer programming content to aid in the teaching of mathematics. Papert (1980), the creator of Constructionism, a theory developed from Jean Piaget's concepts of Constructivism, proposed the use of computers as a learning tool, allowing students to construct their own knowledge. This approach encourages students to take an active role in the learning process, highlighting the importance of creative and meaningful experiences that promote more qualified learning.

Some programming languages, such as *Python,* are widely recognized for their simplicity, flexibility, practicality, and efficiency (Lutz, 2013). In addition, they have a rich framework of libraries with predefined functions that facilitate the implementation of computational solutions. These characteristics and the support offered by ready-made libraries make *Python* an excellent choice from a computational point of view. However, from a pedagogical perspective, especially for students who are just beginning to learn programming languages, this approach may not be an effective methodology. This is because the use of ready-made functions can limit the exploration of the implicit processes used in the development of an algorithm, hindering the development of computational thinking and mathematical thinking, which are closely intertwined (Paper, 1980).

Therefore, *Python* reveals great potential as a tool to support mathematics teaching, especially through libraries such as NumPy, SymPy, and SciPy, which simplify the resolution of more complex mathematical problems. Based on this assumption, an important reflection arises, particularly in the context of introducing programming languages to students in basic education. The question that emerges is: *"Although Python's mathematical libraries offer powerful tools to facilitate calculations and analysis, to what extent can over-reliance on these ready-made libraries compromise the development of fundamental skills such as logical, mathematical, and computational thinking? Furthermore, can this dependence convey a false sense of mastery of mathematical skills?"*

Given this question, it is essential to discuss this issue within the scientific community. Through this discussion, it is possible to generate hypotheses such as: i) excessive use of ready-made *Python* functions may limit understanding of fundamental programming and mathematical concepts, hindering the development of essential skills; ii) relying solely on ready-made functions leads to difficulties in advancing in concepts in the field, in addition to creating a false sense of competence, hindering the acquisition of the necessary skills in computational mathematical thinking.

To suggest a path to address the problem and corroborate the hypotheses mentioned, this paper advocates a teaching methodology that focuses on developing computational algorithms in detail, step by step, establishing connections between programming and mathematics. This encourages the improvement of essential skills, such as computational mathematical logic, algorithmic reasoning, and the ability to solve complex problems, as discussed by Stephens (2018). This methodology not only reinforces logical and analytical thinking but also facilitates knowledge transfer and learning in new contexts.

Thus, this paper contributes to Mathematics Education by critically discussing two approaches to the use of the *Python* language: on the one hand, the excessive use of ready-made libraries that promote computational facilities, and on the other, step-by-step programming as a strategy for developing computational and mathematical skills. By analyzing how each approach impacts logical-computational reasoning and the learning of mathematical concepts, the study offers theoretical and practical support for more conscious pedagogical practices. To illustrate the effects of these methodologies, two mathematical examples are presented, with comments that highlight the positive and negative impacts of both strategies. Thus, the paper strengthens the debate on pedagogical practices that actually develop computational and mathematical thinking in an integrated and critical way, offering theoretical and practical support for teachers and trainers working in Basic Education and teacher training.

It is hoped that this paper will make a significant contribution to the field and collaborate with the academic and scientific community, in the sense of enabling gaps for increasingly broader discussions on the need to include programming language as a support tool in mathematics teaching, creating a more engaging and motivating environment that fosters characteristics such as student participation and assistance, in line with active teaching methodologies.

2 Theoretical Framework

The inclusion of programming language in basic education is still, in many countries, a reality limited to extracurricular contexts (Geldreich & Hubwieser, 2020). However, this situation has gradually changed over the years, as discussions in the academic community have revealed the potential of computing as an auxiliary tool for the development of analytical, critical, and creative thinking (Papert, 1980). In addition to fostering problem-solving and collaboration skills, computing also promotes the construction of the digital world, computational thinking, and digital culture (Siqueira, 2022).

Given its contribution to building skills and competencies in this area, countries such as the United Kingdom (Brown *et al.* 2014), Australia (Falkner *et al.* 2014), Finland (Kwon & Schroderus, 2017), and Japan (Lv, Yang & Zhang, 2022) have taken significant steps in this direction, including computer science content in their school curricula. In Brazil, this movement has been accompanied by initiatives from the Ministry of Education (MEC), which seeks to incorporate these innovations into the school curriculum from the educational base. The recently approved CNE/CEB opinion No. 2/2022 represents a milestone in this process, proposing a resolution for the teaching of computing in basic education with the enactment of Law No. 14.533/2023, which updates the Law on Guidelines and Bases for National Education (LDB) by including mandatory digital education in the basic curriculum.

This inclusion is still in its early stages, and therefore it is not very clear to what extent and in what form the teaching of computing and programming can and should be introduced into basic education in Brazilian schools. Even at this uncertain stage, there has been extensive discussion in favor of including programming, as argued by Lovatti *et al*. (2017), Wing (2006), and Geldreich and Hubwieser (2020). To date, a reference curriculum proposal for technology and computing to complement the BNCC for early childhood and elementary education has been presented by Raabe, Brackmann, and Campos (2018). Similarly, for secondary education, a proposal was presented by the Center for Innovation in Brazilian Education (Campos & Dias, 2020). Meanwhile, in 2022, a proposal for standards on computing in basic education was presented as a complement to the BNCC (Siqueira, 2022).

These changes enable students to move from being mere passive users of technology to becoming creators in the digital world (Garneli, Giannakos & Chorianopoulos, 2015). By taking on an active role, students can develop, innovate, and contribute significantly to digital culture (Resnick *et al*., 2009), becoming protagonists in the process of knowledge construction. This focus on student protagonism reflects the principles of active methodologies, which shift the focus from the teacher as the main transmitter of knowledge to the role of facilitator (Weisz, 2004) and (Bacich & Moran, 2018). In this context, students are encouraged to actively participate in the teaching and learning process, developing autonomy, critical thinking, and reasoning skills to solve proposed problems (Perim et al., 2023) and (Pinto et al., 2020).

Based on the authors' experience, it is believed that active methodologies offer significant support for the development of computational thinking, aligning with the skills promoted by mathematical thinking. Skills such as problem solving, algebraic reasoning, and algorithmic thinking are common to the approaches presented, reinforcing the intersection between Programming and Mathematics. Thus, the mandatory inclusion of digital education in the Brazilian basic curriculum strengthens this connection by establishing competencies that interconnect these areas of knowledge. Computational thinking contributes to the construction of analytical skills essential to Mathematics, as pointed out by Wing (2006). This relationship promotes the idea that learning to program can enhance the learning of mathematical concepts, as discussed by Morais, Basso, and Fagundes (2017). In addition, according to Papert (1980), the use of computers and the practice of programming allow students to understand mathematical concepts more intuitively.

The relationship between programming and mathematics has increasingly fueled discussions in the academic community through proposals for postgraduate work, such as Junior (2021), Navarro (2021), and Azevedo (2022). Several studies have explored methodologies that integrate mathematics teaching with skills development through programming. In Giraldo, Mattos, and Caetano (2012), the authors propose the design of teaching situations as an approach to incorporating programming into the classroom, facilitating the learning of mathematical concepts. Similarly, Souza (2016) presents teaching sequences focused on teaching specific programming topics, highlighting mathematical skills, while Morais, Basso, and Fagundes (2017) discuss the application of programming to teach basic mathematical operations, increasing student engagement and understanding.

Some of the papers cited used the *Python* programming language. For teaching mathematics, some *Python* libraries stand out for their focus on mathematical calculations and operations, such as NumPy, SymPy, and SciPy. Some works, such as Harris et al. (2020), explore the potential of NumPy to facilitate the understanding of mathematical concepts, highlighting its efficiency mainly in performing matrix operations. Similarly, Meurer et al. (2017) highlighted the importance of using SymPy in symbolic algebra, solving problems involving derivatives, integrals, and differential equations. Virtanen et al. (2020) discussed applications of SciPy, demonstrating how its advanced tools can be used in teaching applied mathematics, including solving optimization problems and data analysis.

From a computational and even mathematical point of view, these libraries are highly efficient, offering simple and straightforward commands for complex calculations and data manipulation. However, from a pedagogical perspective, especially for beginners in programming, excessive use of predefined functions can limit understanding of the fundamentals of both programming and mathematics. This excess can restrict the development of essential skills, such as logical reasoning and the ability to build algorithms from scratch.

Relying solely on ready-made functions can lead to difficulties in advancing in concepts in the field, as well as giving a false sense of mastery, since the student does not need to understand the fundamentals behind the operations performed. This becomes evident in practical situations, such as when using the NumPy library to solve mathematical problems. For example, when implementing an algorithm to multiply two matrices (A) and (B), students can simply use the command numpy.dot(A, B) to obtain the immediate result, without necessarily understanding the logic of matrix multiplication. Another example is the calculation of polynomial roots. In the case of a second-degree polynomial, simply use numpy.roots([a, b, c]), providing a list with the coefficients a, b, and c. This demonstrates that the student would not necessarily need to understand how to calculate the value of delta or the three cases that determine the existence and nature of the roots according to the Bhaskara formula.

A relevant analogy, although not common in the literature, can be drawn between the excessive use of ready-made computational functions by novice students in programming languages and the resolution of multiple-choice questions in mathematics. In both situations, students have two possible approaches: the superficial approach, in which, faced with a lack of understanding or mastery of the content, they may resort to random guesses, which, although they may eventually lead to the correct answer, do not promote real understanding of the concepts involved. This problem is widely discussed in the literature, including studies such as those by Burton et al. (1991) and others that address the limitations of multiple-choice questions and associated pedagogical practices, such as Nicol (2007).

In programming, the frequent use of ready-made functions can hinder understanding of the internal processes and logic required to create algorithms (Linge & Langtangen, 2020). The second possibility, similarly, if an analogy is made with mathematics, would correspond to answering multiple-choice questions without an exploratory analysis of the skills involved, relying only on luck or pattern recognition, thus limiting the development of logical and critical thinking that these activities seek to stimulate (Nicol, 2007). Both practices can induce a false perception of competence, compromising the development of the fundamental skills required for autonomous and creative problem solving.

On the other hand, there is also the exploratory approach, which requires students to go through each step of the logical process until they reach the result of a computational algorithm. This methodology promotes more qualified and meaningful learning by connecting programming and mathematics, developing crucial skills such as computational mathematical logic, algorithmic reasoning, and problem solving (Stephens, 2018). In addition, by building algorithms step by step, students strengthen their logical and analytical thinking, which facilitates the transfer of knowledge to new contexts and challenges. This practice not only fosters autonomy in learning, but also encourages a more comprehensive understanding of the concepts covered.

3 Interdisciplinary application between Mathematics and Programming

This section explores the duality of the approaches presented in the theoretical framework (superficial and exploratory) applied to mathematical problem solving using the *Python* programming language. The objective is to demonstrate different strategies for constructing computational algorithms based on two mathematical problems relevant to basic education. It is important to note that the methods presented do not exhaust all possible solutions. The steps can be adjusted, simplified, or reformulated in different ways, depending on each person's level of knowledge in mathematics and programming.

3.1 Approaches with Quadratic Equations

Second-degree polynomial problems are a mandatory component of the mathematics curriculum in Brazilian elementary education. They are addressed with the aim of developing analytical and logical skills and increasing the ability to solve problems in a structured manner. According to the BNCC, this topic is directly associated with several skills, such as EF08MA09, which proposes that students should be able to "Solve and develop, with and without the use of technology, problems that can be represented by second-degree polynomial equations." or EM13MAT302, which encourages students to "Build models using first- or second-degree polynomial functions to solve problems in different contexts." In addition to other skills such as EF09MA09, EM13MAT402, and EM13MAT502.

Contextualization of the Proposed Problem: Consider that a technology company specializing in the production of headphones buys each basic unit for \$1.00 and, in addition, incurs an additional cost of \$6.00 per unit for finishing and distribution, totaling a cost of \$7.00 per unit. Initially, the company resells each headset for R\$ 15.00. However, to encourage large sales, the company offers a progressive discount: for each unit sold, beyond the initial 100, the selling price per unit decreases by R\$ 0.01. The company also has a fixed monthly cost of \$200.00, which covers expenses such as rent and energy.

Given these costs and the progressive discount strategy, the entrepreneur realized the need to optimize sales to avoid losses and ensure the highest possible return. To do this, he hired a mathematician who, through detailed analysis, could determine the ideal number of headphones to be sold to maximize the company's profit.

Mathematical Formalization: The problem described can be modeled mathematically using the following variables and functions: let x be the number of headphones sold by the company. The selling price per unit, p(x), is given by:

$$p(x) = \begin{cases} 15, \ \text{if } x \leq 100 \\ 15 - 0.01\ (x - 100), \ \ \text{if } x > 100 \end{cases}$$

Total revenue R(x) is the product of the price per unit p(x) and the number of units sold x, that is, $R(x) = p(x)x$. Therefore, for $x \leq 100$, total revenue will be $R(x) = 15x$. For $x > 100$, total revenue can be expressed as:

$$R(x) = (16 - 0.01x)x = 16x - 0.01\ x^2.$$

The total cost C(x) consists of a fixed monthly cost of $200, which covers the company's general expenses, and a variable cost of $7 per unit produced, which includes the purchase cost ($1) and the production cost ($6). Thus, the total cost function is given by $C(x) = 200 + 7x$. Finally, the profit L(x) is given by the difference between the total revenue R(x) and the total cost C(x), $L(x) = R(x) – C(x)$. For $x \leq 100$, the profit is:

$$L(x)=15x - (200 + 7x) = 8x - 200.$$

For $x > 100$, the profit is:

$$L(x) = (16x - 0.01x^2) - (200 + 7x) = -0.01x^2 + 9x - 200.$$

To maximize profit, simply note that the profit function is increasing for $x \leq 100$, and the highest profit is obtained when the company sells 100 units, resulting in a profit of $ 600. Now, let's analyze how profit behaves when the number of units sold exceeds 100, with the profit function given by $L(x) = -0.01x^2 + 9x - 200$. This is a quadratic (parabolic) function, in which the coefficient of x^2 is negative (−0.01), which indicates that the parabola opens downward and, therefore, the vertex will represent the point of maximum profit. The vertex of a parabola $ax^2 + bx + c$ occurs at the coordinate x_v, given by $x_v = -b / 2a$.

In this case, we have $x_v = 450$. Therefore, the point of profit maximization occurs when the company sells 450 units. This is the number of units that maximizes profit. Now, to verify the maximum profit value, simply substitute $x = 450$ in the profit function L(x), that is, $L(450) = 1{,}825.00$. Therefore, the maximum profit is R$ 1,825.00.

Computational Algorithm (Direct Form): Using the *NumPy* library, solving a quadratic equation becomes a simple and straightforward process. With the `roots` command, you can quickly and efficiently calculate the roots of the equation, as shown in the code in Table 1.

Table 1: Code for solving a second-degree polynomial using the roots function.

```
In [1]:  from numpy import roots
In [2]:  coeficientes = [-0.01, 9, -200]
In [3]:  root = roots(coeficientes)
In [4]:  print("calculate the roots ", root)
```

Source: Prepared by the authors.

This method optimizes the computational process by speeding up the calculation and displaying the results almost instantly. In the first line of code, the `roots` function is imported directly from the *Numpy* library, allowing the roots of the second-degree polynomial to be displayed. In the second line, the coefficients of the equation $a = -0.01$, $b = 9$, and $c = -200$ are provided as input. Then, in the third line, the `roots` function performs the calculation and allocates the value(s) of “x” to a variable called root, which is finally printed in line 4. However, note that by using the ready-made `roots` function, the student is deprived of the opportunity to understand and apply the logical reasoning behind each step of the solution process, which limits their understanding of the problem and the mathematical steps involved.

Although this approach is efficient for obtaining the final solution to the problem, it has significant disadvantages in terms of assessing logical reasoning and understanding the mathematical concepts involved. Studies such as Sartori and Duarte (2021) highlight that the absence of an analysis of the solution process can lead students to focus more on memorization than on understanding the problem. In this specific case, it is possible to consider that students may simply memorize that any type of polynomial can be solved by simply calling the `roots` function, passing the coefficients of the polynomial they want to solve.

In mathematics, these types of problems that allow for memorization and guessing answers may be directly linked to activities that make up closed multiple-choice questions. As pointed out by Sartori and Duarte (2021), mathematical problems structured in a closed format, such as multiple-choice questions, often do not adequately reflect the reasoning used in the solution. This can lead students to resort to chance when choosing answers and to prioritize the result, disregarding the importance of the reflective process.

Thus, in the interdisciplinary context of Mathematics and Programming, using only ready-made algorithms to solve problems can compromise students' development in understanding essential fundamentals. This mechanism tends to generate a false perception of learning, both in relation to mathematical concepts and to logic and code writing. Without actively exploring the steps of each stage of algorithm construction, students run the risk of becoming mere executors of automated solutions (Greefrath, Hertleif & Siller, 2018), missing the opportunity to build critical and creative thinking (Perim *et al.,* 2023; Costa & Gontijo, 2024).

Computational Algorithm (Step by Step): On the other hand, when this computational algorithm is constructed following a logical and structured sequence, students have the opportunity to develop important skills and abilities, such as those recommended by the BNCC. For example, skill EF06MA23 encourages the construction of algorithms to solve situations step by step. In addition, skill EF07MA05 proposes solving the same problem using different algorithms, promoting flexibility in thinking and the ability to approach an issue from multiple perspectives. These are directly connected to computational mathematical thinking, since it encourages breaking down a problem and creating a solution that performs a set of operations in an orderly manner.

The solution to this problem is presented first through pseudocode that describes the logic of the process in an abstract way, and then with the code implemented in *Python* using the Math library (only to calculate the square root). Table 02 shows both solutions, highlighting how the logical structure translates into programming instructions. The pseudocode serves as a simplified representation, ideal for understanding the conceptual approach, while the algorithm in *Python* shows the practical application of this reasoning in a computational environment.

Table 2: Code for solving a second-degree polynomial step by step.

Pseudocode	Computational Algorithm

`1. Math library`
 	
`2. Start`
`3. Coefficients a, b, c`
`4. Check:`
	`In [1]: import math`
	
	`In [2]: a = -0.01`
	`In [3]: b = 9`
	`In [4]: c = -200`

```
5. If a = 0:
6. It is not a quadratic
   function
7. End of If
8. Else:
9. Calculate delta
10.  Check:
11.  If delta < 0:
12.  Complex root
13.  End of If
14.  Else delta = 0:
15.  Calculate the root
16.  End of Else then
17.  Else:
18.  Calculate two roots
19.  End of Else
```

```
In [5]: if a == 0:
In [6]:   print("not secondary")
In [7]: else:
In [8]:  delta = b**2 - 4 * a * c
In [9]:  print("Delta is: ", delta)
In [10]: if delta < 0:
In [11]:   print("Complex root")
In [12]: elif delta == 0:
In [13]:   x1= -b / 2*a
In [14]:   print("Root: ", x1)
In [15]: else:
In [16]:   x1=(-b+math.sqrt(delta))/2*a
In [17]:   x2=(-b-math.sqrt(delta))/2*a
In [18]:   print("Roots: ", x1, " ", x2)
```

Source: Prepared by the authors.

In pseudocode, students are first encouraged to think broadly about how to build the algorithm. The use of pseudocode aims to help students visualize the computational organization of the program and establish connections with the mathematical logic involved. In this way, students not only understand the functionality of the program, but also realize the importance of structural logic, which is necessary for the algorithm to execute correctly.

Now, note that in the computational algorithm, in line 5, a conditional command `if` is inserted, which leads the student to understand that a second-degree polynomial function only exists if, and only if, the value of the angular coefficient *a* is not equal to zero. Otherwise, the equation will result, at most, in a Linear Polynomial Function. Thus, by providing a value for *a* other than zero, the algorithm continues its execution in the block corresponding to the conditional command `else`, located between lines 7 and 18.

Thus, by remembering mathematical concepts on the subject, we know that to calculate the roots of a second degree function, the first step is to determine the value of delta (line 8). After this step, the algorithm can follow three different paths, defined by a structure of nested conditionals (lines 10, if delta is less than zero, line 12 if delta is equal to zero, and line 15 otherwise, that is, if delta is greater than zero). Given this, it is possible for students to clearly and objectively understand why, computationally, it is necessary to use the implementation of these three conditionals, since it is the requirements imposed by mathematical criteria that determine the possibility of correctly finding the roots of a specific polynomial.

Thus, in this case, the exploration of sequential steps enables the construction of knowledge in a robust and accurate manner, reflecting approaches widely discussed in educational and scientific theories, leading to the concept of learning, that is, when individuals explore diverse perspectives and actively construct their own understanding, breaking with passive teaching models. In this regard, Papert (1980) emphasizes that active exploration and the creation of multiple paths are fundamental to meaningful learning, especially in the field of educational computing.

Through the nested conditional structure sequence, it is possible to reinforce how mathematical concepts can be integrated into computational thinking, relating them to various practical problems. Contextualization is an important factor in teaching and learning, as it enables students to see mathematics in different everyday contexts, as shown by Araújo et al. (2024) and Matos et al. (2024). In this case, the nested conditional structure resembles the idea of piecewise defined functions, used in mathematics to model situations in which a function behaves differently depending on an interval or specific conditions. An example is illustrated below.

Example: In an electricity billing system, the monthly cost for a residence is calculated based on consumption (in kWh) divided into consumption brackets. The rules are as follows:

- *Up to 100 kWh: Minimum rate of R$ 80.*
- *From 101 kWh to 200 kWh: R$ 0.60 per additional kWh.*
- *Above 200 kWh: R$ 1.50 per additional kWh.*

When determining a function that represents this situation, it is possible to write:

$$f(x) = \begin{cases} 80, & \text{if } 0 \leq x \leq 100 \\ 80 + 0.60(x - 100), & \text{if } 101 \leq x \leq 200 \\ 80 + 0.60 \cdot 100 + 1.50(x - 200), & \text{if } x > 200 \end{cases}$$

where x is the consumption value in *(in kWh).*

In this context, a piecewise-defined function is presented, in which each interval or condition can be described by a conditional structure in programming language. Thus, the three mathematical conditions are translated into conditional commands in a logical way: `if x <= 100`: for the first range, `elif 100 < x and x <= 200`: for the second, and finally, an `else`: block that covers cases where `x > 200`. This transcription shows how mathematical concepts can be directly mapped to computational structures, facilitating both understanding and practical implementation.

Therefore, it is possible to say that Programming and Mathematics are totally connected, especially in the context of the problems presented. Computational algorithms are capable of generating correct and efficient results precisely because they are based on mathematical principles, such as logical operations and the structured use of conditional controls. Therefore, in the practice of Programming, the absence of mathematical precision can have serious consequences, such as incorrect results. Mathematical precision plays a fundamental role, ensuring not only the robustness of calculations, but also the reliability of the algorithms developed.

3.2 Approaches with Matrices

Another problem that can be used to demonstrate the connection between mathematics and programming is found in the study of matrices and basic operations between them. Although the BNCC does not explicitly mention the teaching of vectors and matrices as mandatory content, it is possible to connect these concepts to some of the skills described in the document. Skill EM13MAT203 highlights the application of mathematical concepts in the construction and analysis of spreadsheets for family budget control, simulators, and other practical purposes. In addition, similar skills are proposed in elementary and middle school, where the use of spreadsheets or alternatives is suggested. Thus, a contextualized problem of the following type is related:

Contextualization of the Proposed Problem: Consider a student who took three subjects: History, Mathematics, and Geography, in three teaching units. Their final grades for each teaching unit are shown in Table 3.

Table 3: Grades obtained by a student in three subjects.

Unit / Course	I	II	III
History	7.50	6.90	7.10
Mathematics	6.00	6.40	8.50
Geography	8.30	7.50	9.20

Source: Prepared by the authors.

This student's math teacher, when analyzing the information in Table 3, identified an opportunity to introduce the concepts of matrices. She encouraged him by pointing out that these concepts arise naturally in solving various problems and are essential, mainly because they organize and simplify calculations. In addition, this approach allows for practical exploration of how matrices can represent real-life situations and how mathematical operations can be applied to data analysis, facilitating the connection between computational thinking and mathematics teaching. This approach allows students to explore, in a practical way, how matrices can represent real-life situations and how mathematical operations can be applied to data analysis, facilitating the connection between computational thinking and mathematics teaching.

Mathematical Formalization: Based on Table 3, the table of elements organized in rows and columns can be represented by a matrix, referred to here as matrix M. Thus, let M be a matrix 3×3, in which each element m_{ij} corresponds to the grade for subject i in unit j, considering the three subjects (History, Mathematics, and Geography) and the three teaching units (I, II, III). Thus:

$$M = \begin{pmatrix} 7.5 & 6.9 & 7.1 \\ 6.0 & 6.4 & 8.2 \\ 8.3 & 7.7 & 9.2 \end{pmatrix}$$

From the M matrix, you can apply various mathematical operations, as shown below.

The sum of all elements of matrix M is given by $\sum_{i,j=1}^{3} m_{ij} = 67.3$. The sum of the History grades, which correspond to the first row of the matrix, is $\sum_{j=1}^{3} m_{1j} = 21.5$. The sum of the scores for unit I, represented by the first column of the matrix, is $\sum_{i=1}^{3} m_{i1} = 21.8$. Finally, the sum of the main diagonal scores is $\sum_{i=1}^{3} m_{ii} = 23.1$. This mathematical formalization illustrates how simple operations can be applied to matrices, allowing students to understand the concepts involved.

Computational Algorithm (Direct Form): When addressing content of this type, it is possible to explore various mathematical analyses through programming. In the case of matrices, the use of the *NumPy* library allows for the efficient manipulation of *arrays* (vectors and matrices). This tool provides several ready-made commands that facilitate mathematical operations. However, as in the previous example, the direct use of these resources may limit the development of students' mathematical skills if not accompanied by a contextualized and reflective approach. Table 4 shows some activities that can be related to this context.

Table 4: Proposed activities based on Table 3 and computational implementation.

a) Present the sum of all the grades in the matrix	`In [1]: import numpy as np`
	`In [2]: M = np.array([[7.5, 6.9, 7.1],`
	`                      [6.0, 6.4, 8.2],`

b) Present the sum of the	

history grades for the three units	`[8.3, 7.7, 9.2]])`
	`In [3]: print("Add all grades:", np.sum(M))`

c) Present the sum of all the grades for unit I	`Out[3]: 67.3`
	`In [4]: print("History:", np.sum(M[0]))`
	`Out[4]: 21.5`

d) Present the sum of the grades on the main diagonal	`In [5]: print("Unit I:", np.sum(M[:,0]))`
	`Out[5]: 21.8`
	`In [6]: print("D.Prin.:", np.sum(np.diag(M)))`
	`Out[6]: 23.1`

Source: Prepared by the authors.

In Table 4, lines 2 to 4, matrix M is constructed using the *array* command, which uses the *NumPy* library imported in line 1. On the left side of the table, activities are proposed from a) to d). Note that each question is quickly solved with relatively simple calls using the *sum* command, which performs addition operations. To sum all the grades in the matrix, simply use `np.sum(M)`. In addition, to sum the elements of a specific row, such as the first row (History grades), use `np.sum(M[0])`. To sum other rows, replace `[0]` with `[1]` (Mathematics) or `[2]` (Geography). To add the elements of a specific column, change the indexing of the matrix. For example, `np.sum(M[:,0])` adds the elements of the first column (unit I), while `[:,1]` corresponds to unit II and `[:,2]` to unit III. Finally, to sum all the scores on the main diagonal, simply write `np.sum(np.diag(M))`. This approach demonstrates the practicality of the *NumPy* library for performing mathematical calculations efficiently and quickly.

The goal here is not to teach programming language with this brief section, but rather to discuss that, from a computational point of view, there is an efficiency in the ready-made routines available to solve this type of problem. However, from a mathematical perspective, these approaches can make it difficult for students to understand logical structures, such as the process used to sum the values in rows, columns, and diagonals.

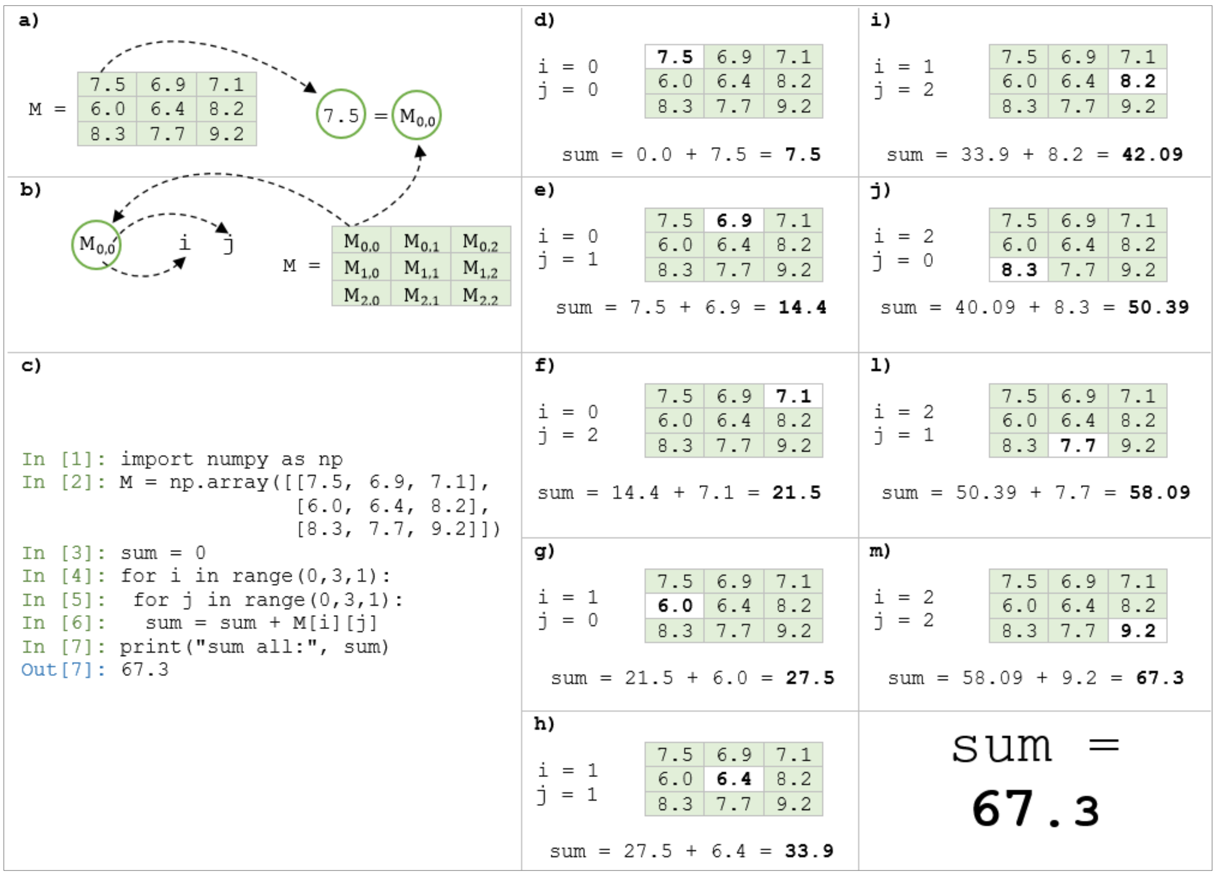
Computational Algorithm (Step by Step): Tackling this problem by following an approach that builds each step of the logical structural sequence step by step gives students an opportunity to understand the internal processes involved in creating a computational algorithm. This methodology goes beyond the simple application of ready-made commands, encouraging further reflection on how mathematical analytical structures are directly connected to the construction of algorithms.

To clarify the logic behind this computational routine, Figure 1 details, step by step, what happens during the execution of the code to add all the numbers in the matrix. This detail helps to visualize how the indices `i` and `j` evolve throughout the iterations. This brief presentation helps to develop fundamental skills necessary for students to understand and develop the algorithm related to this problem.

The first step in this sequence is to ensure that students understand how a conventional mathematical matrix, Figure 1 a), is related to the mathematical matrix with indices, Figure 1 b). This understanding includes the notion that each element of the matrix is associated with a pair of indices that identifies its position within the structure. For example, in this 3×3 matrix, the indices (0,0), (0,1), (0,2), (1,0), (1,1), (1,2), (2,0), (2,1), and (2,2) correspond to the specific positions of its elements. Each element of this matrix is associated with a pair of indices ranging from 0 to 2, allowing its representation in the structure $\mathtt{M}_{\mathtt{[i][j]}}$, where `i` and `j` take the values 0, 1, 2. This notation clarifies the organization of the elements of the matrix into rows and columns, which will be used to perform mathematical operations.

Assuming that the student already has prior knowledge of the `for` repetition structure, it is possible to establish the relationship between the indices `i` and `j` and their behavior within the repetition loop. This connection allows students to understand how the indices traverse each element of the matrix, row by row and column by column, facilitating the requested execution. Figure 1 c) shows an implementation option for adding all the elements of the array using the `for` repetition structure. This example illustrates how the indexes `i` and `j` traverse the array, accessing each element individually, while accumulating the values in a sum variable. This approach allows you to explore the concept of indexing from a computer language perspective, as well as allowing students to understand the logic that allows all grades to be added together.

Figure 1: Steps for adding the elements of a 3 x 3 matrix.



a)
M =
7.5 6.9 7.1
6.0 6.4 8.2
8.3 7.7 9.2
7.5 = $M_{0,0}$
b)
$M_{0,0}$ i j
M =
$M_{0,0}$ $M_{0,1}$ $M_{0,2}$
$M_{1,0}$ $M_{1,1}$ $M_{1,2}$
$M_{2,0}$ $M_{2,1}$ $M_{2,2}$
c)
d) i = 0 j = 0 sum = 0.0 + 7.5 = 7.5
e) i = 0 j = 1 sum = 7.5 + 6.9 = 14.4
f) i = 0 j = 2 sum = 14.4 + 7.1 = 21.5
g) i = 1 j = 0 sum = 21.5 + 6.0 = 27.5
h) i = 1 j = 1 sum = 27.5 + 6.4 = 33.9
i) i = 1 j = 2 sum = 33.9 + 8.2 = 42.09
j) i = 2 j = 0 sum = 40.09 + 8.3 = 50.39
l) i = 2 j = 1 sum = 50.39 + 7.7 = 58.09
m) i = 2 j = 2 sum = 58.09 + 9.2 = 67.3
sum = 67.3

```
In [1]: import numpy as np
In [2]: M = np.array([[7.5,  6.9,  7.1],
                      [6.0,  6.4,  8.2],
                      [8.3,  7.7,  9.2]])
In [3]: sum = 0
In [4]: for i in range(0,3,1):
In [5]:  for j in range(0,3,1):
In [6]:   sum = sum + M[i][j]
In [7]: print("sum all:", sum)
Out[7]: 67.3
```

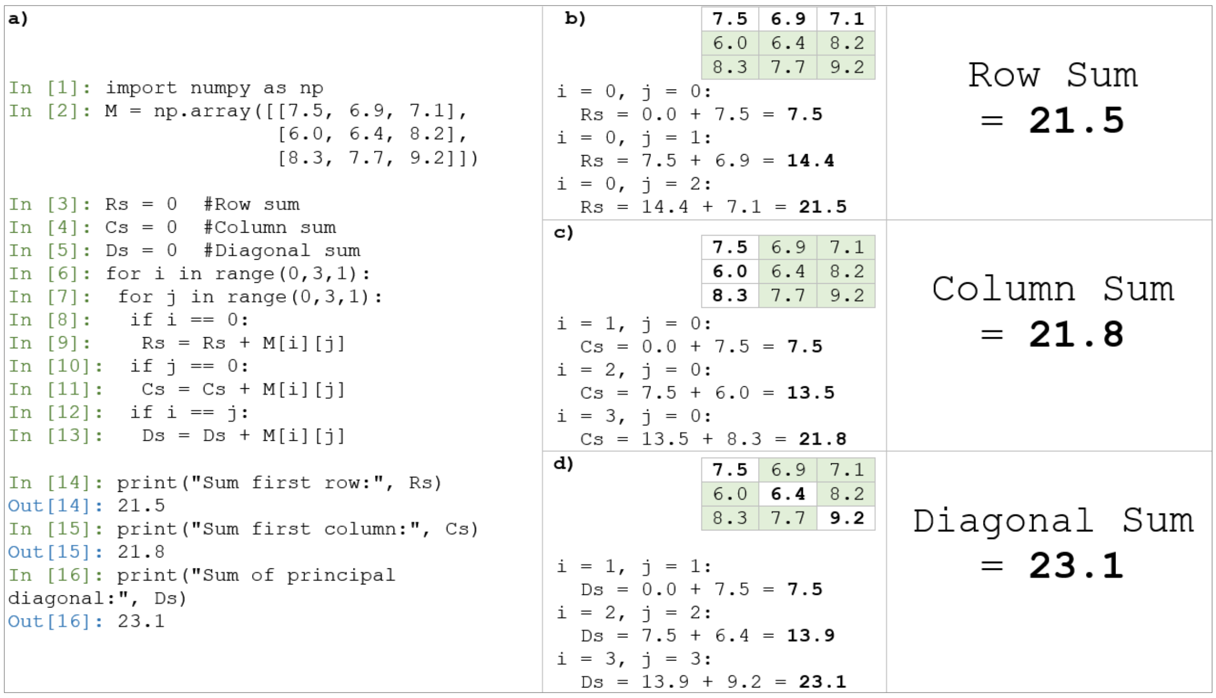
Source: Prepared by the authors.

Note in Figure 1 c) that the `sum` variable (line 3) is initialized outside the *loop*, as its value needs to be set to zero only once before the iterations begin. Next, the nested *loops* are started: the index `i` represents the rows of the array (line 4), while the index `j` represents the corresponding columns (line 5). Within these *loops*, the summation process is performed iteratively, element by element, until all values in the matrix have been analyzed. This mechanism ensures that each element is accessed and included in the total sum. The process is only completed after all combinations of `i` and `j` have been explored. Figure 1 d) to m) shows, step by step, what happens internally during the execution of the code to sum all the notes in the matrix. This detail helps to visualize how the indices `i` and `j` evolve throughout the iterations, highlighting the behavior of the algorithm and how the matrix is processed.

To solve questions b), c), and d) proposed in Table 4, it is essential to identify connections and patterns in the index matrix as shown in Figure 1 b). For example, when addressing the problem of adding all the values in row 1, corresponding to history grades, students can be prompted with the following question: *Based on the matrix shown, is it possible to detect any pattern that allows you to access only the values in row 1 when traversing the loops?* Similarly, the same reasoning can be applied to column 1 and the main diagonal. These reflections, which involve understanding how to adjust the conditions of the *loop* to access only the desired elements, encourage students to exercise logical and mathematical reasoning. This process, in turn, contributes directly to the development of fundamental skills for mathematical and computational thinking.

Figure 2 presents a proposed computational algorithm for solving activities b), c), and d) proposed in Table 4.

Figure 2: Operations on a 3x3 matrix. In a) sum of the elements of the first row, in b) sum of the elements of the first column, and in c) sum of the elements of the main diagonal.



a)

```
In [1]: import numpy as np
In [2]: M = np.array([[7.5, 6.9, 7.1],
                      [6.0, 6.4, 8.2],
                      [8.3, 7.7, 9.2]])

In [3]: Rs = 0  #Row sum
In [4]: Cs = 0  #Column sum
In [5]: Ds = 0  #Diagonal sum
In [6]: for i in range(0,3,1):
In [7]:  for j in range(0,3,1):
In [8]:   if i == 0:
In [9]:    Rs = Rs + M[i][j]
In [10]:  if j == 0:
In [11]:   Cs = Cs + M[i][j]
In [12]:  if i == j:
In [13]:   Ds = Ds + M[i][j]

In [14]: print("Sum first row:", Rs)
Out[14]: 21.5
In [15]: print("Sum first column:", Cs)
Out[15]: 21.8
In [16]: print("Sum of principal
diagonal:", Ds)
Out[16]: 23.1
```

b)

7.5	**6.9**	**7.1**
6.0	6.4	8.2
8.3	7.7	9.2

```
i = 0, j = 0:
  Rs = 0.0 + 7.5 = 7.5
i = 0, j = 1:
  Rs = 7.5 + 6.9 = 14.4
i = 0, j = 2:
  Rs = 14.4 + 7.1 = 21.5
```

Row Sum = **21.5**

c)

7.5	6.9	7.1
6.0	6.4	8.2
8.3	7.7	9.2

```
i = 1, j = 0:
  Cs = 0.0 + 7.5 = 7.5
i = 2, j = 0:
  Cs = 7.5 + 6.0 = 13.5
i = 3, j = 0:
  Cs = 13.5 + 8.3 = 21.8
```

Column Sum = **21.8**

d)

7.5	6.9	7.1
6.0	**6.4**	8.2
8.3	7.7	**9.2**

```
i = 1, j = 1:
  Ds = 0.0 + 7.5 = 7.5
i = 2, j = 2:
  Ds = 7.5 + 6.4 = 13.9
i = 3, j = 3:
  Ds = 13.9 + 9.2 = 23.1
```

Diagonal Sum = **23.1**

Source: Prepared by the authors.

To add all the values in row 1 of matrix `M`, it is necessary to identify that, in that row, all indices i are equal to zero. Therefore, it is possible to include a condition of the type `if i == 0` (line 8) in the algorithm, ensuring that only the elements of row 1 are considered in the sum. Similarly, to add the values in the first column, the student must note that, for that column, all indices j are equal to zero. Consequently, it is sufficient to add a condition `if j == 0` (line 10). Finally, to add the elements of the main diagonal, it is necessary to realize that, in this region of the matrix, the indices i and j are always equal. Thus, the inclusion of a conditional `if i == j` (line 12) in the algorithm solves the problem.

3.3 Pedagogical Analysis and Applications in Mathematics Education

The proposal to use algorithms in *Python* to work with quadratic equations and matrix operations has significant implications for the teaching and learning of mathematics. When this approach goes beyond simply executing ready-made code and involves students in the manual construction of algorithms, the process becomes pedagogically more effective. By programming step by step the solution of a quadratic equation or the multiplication of matrices, for example, students are led to revisit and consolidate fundamental concepts, such as the identification of roots, the structure of the quadratic function, and algebraic properties such as distributivity, associativity, neutral elements, among others. This practice favors the development of mathematical reasoning and brings students closer to a better understanding of the content. Thus, this approach corroborates the principles of Mathematics Education, as it stimulates the active construction of knowledge and increases students' autonomy in problem solving.

Therefore, the use of programming allows students to explore how and why operations work. From a didactic point of view, this strategy contributes to overcoming recurring conceptual difficulties in mathematics teaching by offering an interactive and experimental approach. As Ponte, Brocardo, and Oliveira (2009) point out, problem solving, with the support of technology, allows for greater cognitive involvement of students and the construction of deeper meanings about mathematical objects. Thus, integrating programming into mathematics teaching promotes interdisciplinarity. This perspective broadens mathematical literacy (D'Ambrósio, 1996) and strengthens student protagonism, as it transforms the classroom into a space for investigation, creation, and reconstruction of mathematical knowledge mediated by technology.

Final Thoughts

When introducing initial programming language education to elementary school students, it is essential to evaluate the methodologies adopted to ensure that everyone is encouraged to explore logical and mathematical reasoning. This process should be aimed at building both mathematical and computational skills and abilities. Considering that many programming languages offer a wide range of ready-made libraries that facilitate and accelerate the creation of algorithms, it is crucial to discuss the extent to which the use of these ready-made functions contributes to the construction of an active and effective methodology in the teaching and learning process.

This discussion is broad and of great interest to the academic and scientific community, given that many countries have recognized the accelerated potential of digital technologies and, consequently, have included computer science content in their basic education curricula. In this context, this paper analyzed the methodological risks related to "facilitating computational and mathematical learning" through the excessive use of ready-made *Python* libraries, in contrast to the approach of "developing computational and mathematical skills," which proposes teaching programming languages in a detailed and progressive, step-by-step manner.

To demonstrate these two approaches, two mathematical problems were carefully selected for their accessible algorithmic construction and lack of complexity, considering the target audience of elementary school students. The central idea in proposing these problems was to show that, without the use of mathematical logic and analytical precision, it is not possible to correctly identify where and how computational commands should be applied. This highlights the connection between programming and mathematics, demonstrating how computational thinking is intrinsically linked to mathematical thinking.

Thus, this text leaves us with a reflection on the importance of balancing, in the teaching of programming languages, the use of ready-made libraries with activities that encourage the construction and understanding of fundamental concepts. This balance is essential both for the development of computational skills and for the deepening of mathematical concepts, promoting coherent and effective learning.

Finally, as this is a theoretical study, this paper does not present empirical data confirming the effects of the approaches discussed in concrete classroom situations. This limitation opens space for future research that can systematically apply and evaluate the impacts of detailed programming education on the development of mathematical skills in elementary school students. We therefore suggest conducting empirical research using qualitative and quantitative methodologies, such as case studies, to test the hypotheses raised in this study, contributing practical evidence to the improvement of programming teaching methodologies in the context of Mathematics Education.

References

Araújo, F. C., Paiva, C. D. C., do Nascimento, R. A., D'arc de Oliveira Passos, J., Martins, G. F. O., Sousa, M. A. D. M. A. & Alves, C. S. (2024). Uso das tecnologias digitais no ensino de matemática: um relato de experiência com aporte do GeoGebra. *Caderno Pedagógico*, *21*(8), e6715-e6715.

Azevedo, G. T. D. (2022). *Processo formativo em matemática: invenções robóticas para o Parkinson*. 2022. 213f. Tese (Doutorado em Educação Matemática). Universidade Estadual Paulista. Rio Claro, SP.

Bacich, L. & Moran, J. (2018). *Metodologias ativas para uma educação inovadora: uma abordagem teórico-prática*. Porto Alegre, RS: Penso Editora.

Brasil. (2018). Ministério da Educação. *Base Nacional Comum Curricular: Educação é a Base* (BNCC). Brasília, DF.

Brown, N. C., Sentance, S., Crick, T. & Humphreys, S. (2014). Restart: The resurgence of computer science in UK schools. *Association for Computing Machinery Transactions on Computing Education*, *14*(2), 1-22.

Burton, S. J., Sudweeks, R. R., Merrill, P. F. & Wood, B. (1991). *How to prepare better multiple-choice test items: Guidelines for university faculty*. Brigham Young University. Department of Instructional Science.

Campos, F. R. & Dias, R. A. (2020). Currículo de referência–Itinerário Formativo em Tecnologia e Computação. *Centro de Inovação para a Educação Brasileira CIEB*. São Paulo, SP.

Costa, I. L. & Gontijo, C. H. (2024). Avaliação Formativa e o Pensamento Crítico e Criativo em Matemática: Mapeamento de Pesquisas e Aplicações. *Paradigma*, e2024004-e2024004.

D'Ambrosio, U. (1996). *Educação Matemática: da teoria à prática*. Papirus Editora.

Falkner, K., Vivian, R. & Falkner, N. (2014). The Australian digital technologies curriculum: challenge and opportunity. In *Proceedings of the Sixteenth Australasian Computing Education Conference.* (148), 3-12.

Garneli, V., Giannakos, M. N. & Chorianopoulos, K. (2015). Computing education in K-12 schools: A review of the literature. *Institute of Electrical and Electronics Engineers Global Engineering Education Conference*. 543-551.

Geldreich, K. & Hubwieser, P. (2020). Programming in primary schools: Teaching on the edge of formal and non-formal learning. *Non-Formal and Informal Science Learning in the Information and Communication Technology Era*, 99-116.

Giraldo, V., Caetano, P. & Mattos, F. (2012). *Recursos computacionais no ensino de Matemática*. Rio de Janeiro. RJ: Sociedade Brasileira de Matemática.

Greefrath, G., Hertleif, C. & Siller, H. S. (2018). Mathematical modelling with digital tools-a quantitative study on mathematising with dynamic geometry software. *Zentralblatt für Didaktik der Mathematik*, (50), 233-244.

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D. & Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357-362.

Junior, H. N. P. (2021). *Matemática e Programação: Uma nova abordagem de ensino.* 2021. 40f. Dissertação (Mestrado em Matemática). Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro, RJ.

Kwon, S. & Schroderus, K. (2017). *Coding in Schools: Comparing Integration of Programminginto Basic Education Curricula of Finland and South Korea*. Finnish Society on Media Education.

Linge, S. & Langtangen, H. P. (2020). *Programming for computations-Python: A gentle introduction to numerical simulations with Python 3.6.* p. 332. Springer Nature.

Lovatti, B. G., Vieira, L. S., Marques, K. & Scolforo, M. A. (2017). A programação no ensino básico: formando alunos para sociedade tecnológica. *Revista Ambiente Acadêmico*, *3*(1), 113-132.

Lutz, M. (2013). *Learning python: Powerful object-oriented programming*. O'Reilly Media, Inc.

Lv, W., Yang, C. & Zhang, W. (2022). Programming Education in Japanese Elementary Schools Integrated with Multiple Subjects: Origins, Practical Paths and Implications. *4th International Conference on Computer Science and Technologies in Education*. 11-21.

Matos, J. D. S. G., Paiva, C. D. C., Lima, F. F. R. R., do Nascimento, R. A., dos Santos, L. M. P., Pinheiro, A. A. & de Brito, I. E. G. (2024). A relação entre pensamento computacional e ensino de matemática no contexto da educação básica: oportunidades, desafios e perspectivas. *Caderno Pedagógico*, *21*(8), e6408-e6408.

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M. & Scopatz, A. (2017). SymPy: symbolic computing in Python. *PeerJ Computer Science*, *3*, e103.

Morais, A. D. D., Basso, M. V. D. A. & Fagundes, L. D. C. (2017). Educação Matemática & Ciência da Computação na escola: aprender a programar fomenta a aprendizagem de matemática? *Ciência & Educação (Bauru)*, *23*(2), 455-473.

Morán, J. (2015). Mudando a educação com metodologias ativas. *Coleção mídias contemporâneas. Convergências midiáticas, educação e cidadania: aproximações jovens*, *2*(1), 15-33.

Navarro, E. R. (2021). *O desenvolvimento do conceito de pensamento computacional na educação matemática segundo contribuições da teoria histórico-cultural*. 2021. 177f. Tese (Doutorado em Educação). Universidade Federal de São Carlos, São Carlos. SP.

Nicol, D. (2007). E-assessment by design: using multiple-choice tests to good effect. *Journal of Further and higher Education*, *31*(1), 53-64.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic books. Eugene, Oregon, United States of America: Harvester.

Perim, F. de C. R., Araújo, F. P. S., Trindade, M. C., Monteiro, M. R. A., Fragoso, N. da S. C., Batista, R. G. M., Benevides, S. R., Carvalho, V. M. & de Oliveira, F. G. (2023). Pensamento crítico na sala de aula: Capacitando alunos para o futuro em uma jornada educacional. *Revista Foco*, *16*(11), e3673.

Pinto, S. F., Ferreira, R. S., Costa, M. M. & Silva, A. M. (2020). A proposal for an active methodology for physics labs. *arXiv preprint arXiv:2005.06532*.

Ponte, J. P.; Brocardo, J.; Oliveira, H. (2009) *Investigação Matemática na Sala de Aula*. 2ª. Ed. Belo Horizonte: Autêntica.

Raabe, A. L., Brackmann, C. P. & Campos, F. R. (2018). *Currículo de referência em tecnologia e computação: da educação infantil ao ensino fundamental*. Centro de Inovação para a Educação Básica. São Paulo, SP.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. & Kafai, Y. (2009). Scratch: programming for all. *Communications of the Association for Computing Machinery*, *52*(11), 60-67.

Sartori, A. S. T. & Duarte, C. G. (2021). Repetir, memorizar, recitar: mecanismos para a fabricação de corpos dóceis pela Educação Matemática. *Jornal Internacional de Estudos em Educação Matemática*, *14*(1), 84-91.

Selwyn, N. (2021). *Education and technology: Key issues and debates*. 3° Ed. London, UK: Bloomsbury Publishing.

Siqueira, I. C. P. (2022). *Normas sobre computação na educação básica–complemento à base nacional comum curricular (BNCC)*. Technical report, Conselho Nacional de Educação-Câmara de Educação Básica.

Souza, E. C. D. (2016). *Programação no ensino de matemática utilizando Processing 2: Um estudo das relações formalizadas por alunos do ensino fundamental com baixo rendimento em matemática*. 2016. 189f. Dissertação (Mestrado em Educação para a Ciência). Universidade Estadual Júlio de Mesquita Filho, Bauru, São Paulo. SP.

Stephens, M. (2018). Embedding algorithmic thinking more clearly in the mathematics curriculum. *Proceedings of the International Commission on Mathematical Instruction Study*, *(24)*, 483-490.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D. & van Mulbregt, P. (2020). Fundamental algorithms for scientific computing in python and SciPy 1.0 contributors. SciPy 1.0. *Nature Methods*, *(17)*, 261-272.

Weisz, T. (2004). *O diálogo entre o ensino e a aprendizagem*. 2° Ed. São Paulo, SP: Ática.

Wing, J. M. (2006). Computational thinking. *Communications of the Association for Computing Machinery*, *49*(3), 33-35.