

Programação *Python* e Ensino de Matemática: Uma Proposta Interdisciplinar com Engenharia Didática

Jailson França dos Santos¹ Mercedes Bêta Quintano de Carvalho Pereira
dos Santos² Francisco Régis Vieira Alves³

Resumo

Contexto: Investigar como a linguagem de programação Python pode contribuir para o ensino e aprendizagem da Matemática na Educação Básica, tomando como referência metodológica a Engenharia Didática e a Teoria das Situações Didáticas. **Objetivo:** Analisar sob uma perspectiva teórica, uma sequência didática que articule comandos de programação a conceitos matemáticos, tais como funções, sequências numéricas e geometria analítica. **Justificativa:** A importância da integração entre programação e Matemática pode favorecer a compreensão conceitual, estimular o raciocínio lógico e aproximar os estudantes de práticas de resolução de problemas. **Metodologia:** Estudo teórico. Analisada as duas primeiras fases da Engenharia Didática: a análise preliminar e a análise a priori, por meio da construção e interpretação de problemas matemáticos hipotéticos. **Resultados:** Consistem na elaboração e análise teórica de uma proposta de sequência didática que relaciona elementos fundamentais de programação com conteúdos matemáticos. Como Considerações Finais, espera-se que a articulação entre Engenharia Didática e Teoria das Situações Didáticas forneça um quadro teórico para orientar professores na elaboração de propostas didáticas que integrem programação e Matemática. **Palavras-chave:** Programação em Python; Engenharia Didática; Sequência Didática.

Python Programming and Mathematics Teaching: An Interdisciplinary Proposal with Didactic Engineering

Abstract

Context: To investigate how the Python programming language can contribute to the teaching and learning of Mathematics in Basic Education, using Didactic Engineering and the Theory of Didactic Situations as methodological references. **Objective:** To analyze, from a theoretical perspective, a didactic sequence that articulates programming commands with mathematical concepts, such as functions, numerical sequences, and analytical geometry. **Justification:** The importance of integrating programming and Mathematics can favor conceptual understanding, stimulate logical reasoning, and bring students closer to problem-solving practices. **Methodology:** Theoretical study. The first two phases of Didactic Engineering were analyzed: preliminary analysis and a priori analysis, through the construction and interpretation of hypothetical mathematical problems. **Results:** These consist of the elaboration and theoretical analysis of a proposed didactic sequence that relates fundamental programming elements with mathematical content. As **Final Considerations**, it is expected that the articulation between Didactic Engineering and the Theory of Didactic Situations will provide a theoretical framework to guide teachers in the elaboration of didactic proposals that integrate programming and Mathematics.

Keywords: Python Programming; Didactic Engineering; Didactic Sequence.

¹ Doutor pela Universidade de Brasília (UnB). Professor na Universidade Federal do Oeste da Bahia (UFOP), Barreiras, Bahia, Brasil. E-mail: jailson.santos@ufop.edu.br.

² Doutora pela Pontifícia Universidade Católica de São Paulo (PUC/SP). Professora da Universidade Federal de Alagoas (UFAL), Maceió, Alagoas, Brasil. E-mail: mercedes.santos@cedu.ufal.br.

³ Doutor pela Universidade Federal do Ceará (UFC). Professor no Instituto Federal de Educação, Ciência e Tecnologia do Estado do Ceará (IFCE), Fortaleza, Ceará, Brasil. E-mail: fregis@ifce.edu.br

Programación en Python y enseñanza de las matemáticas: una propuesta interdisciplinar con ingeniería didáctica

Resumen

Contexto: Investigar cómo el lenguaje de programación Python puede contribuir a la enseñanza y el aprendizaje de las Matemáticas en la Educación Básica, utilizando la Ingeniería Didáctica y la Teoría de las Situaciones Didácticas como referencias metodológicas. **Objetivo:** Analizar, desde una perspectiva teórica, una secuencia didáctica que articula comandos de programación con conceptos matemáticos, como funciones, sucesiones numéricas y geometría analítica. **Justificación:** La integración de la programación y las Matemáticas favorece la comprensión conceptual, estimula el razonamiento lógico y acerca a los estudiantes a la resolución de problemas. **Metodología:** Estudio teórico. Se analizaron las dos primeras fases de la Ingeniería Didáctica: análisis preliminar y análisis a priori, mediante la construcción e interpretación de problemas matemáticos hipotéticos. **Resultados:** Consisten en la elaboración y el análisis teórico de una secuencia didáctica propuesta que relaciona elementos fundamentales de la programación con contenido matemático. Como **Consideraciones Finales**, se espera que la articulación entre la Ingeniería Didáctica y la Teoría de las Situaciones Didácticas proporcione un marco teórico que oriente a los docentes en la elaboración de propuestas didácticas que integren la programación y las Matemáticas.

Palabras clave: Programación en Python; Ingeniería Didáctica; Secuencia Didáctica.

INTRODUÇÃO

Diferentes pesquisas têm evidenciado a relevância da integração entre pensamento computacional e pensamento matemático, destacando seu potencial para contribuir com processo de ensino e aprendizagem (Voskoglou & Buckley, 2012; Lockwood & Mooney, 2017). Nesse contexto, a programação computacional tem sido concebida não apenas como uma habilidade técnica, mas como um recurso capaz de ampliar as formas de visualização, simulação e resolução interativa de problemas matemáticos (Ziatdinov & Valles JR, 2022; Brandsaeter & Berge, 2025).

Apesar das evidências, a aplicação da linguagem de programação como mediadora no ensino de Matemática, especialmente no contexto da Educação Básica, enfrenta desafios como a falta de formação docente específica (Humble, Mozellius & Sallvin, 2020; Liu *et al.*, 2024), baixo número de materiais curriculares, falta de estrutura tecnológica básica (Macrides, Miliou & Angeli, 2022; TPE, 2024) e escassez de estudo além de ausência de sequências didáticas estruturadas que considerem tanto as dimensões pedagógicas quanto as computacionais (León, Robles & Román-González, 2016).

Quanto à linguagem de programação, destaca-se o *Python* (Lutz, 2013), amplamente reconhecida por sua simplicidade sintática, disponibilidade gratuita e diversas aplicações em diferentes áreas do conhecimento. No contexto educacional, *Python* tem-se consolidado como recurso introdutório para programação, por possibilitar que estudantes compreendam conceitos computacionais ao mesmo tempo em que exploram ideias matemáticas, favorecendo a articulação entre resolução de problemas, raciocínio lógico e representação simbólica (Chongchitnan, 2023).

Portanto, com base nas diferentes aplicabilidades do *Python* e buscando oferecer uma contribuição acadêmica diante do baixo número de materiais curriculares disponíveis, emerge a necessidade de referenciais teórico-metodológicos capazes de orientar a construção de propostas didáticas que articulem Matemática e programação. Nesse sentido, a Engenharia Didática (Artigue, 2002; 2020) é adotada como aporte metodológico neste estudo, por oferecer uma estrutura sistemática para a elaboração e análise de situações de ensino. Assim, formula-se a seguinte questão de pesquisa: *como utilizar a linguagem Python para desenvolver propostas de ensino com diferentes problemas matemáticos, articulando conceitos computacionais e matemáticos por meio de uma abordagem fundamentada na Engenharia Didática?*

Isso posto, a pesquisa adota uma abordagem de natureza teórica e qualitativa, direcionada a duas frentes complementares: (i) uma análise teórica, fundamentada na revisão de literatura, que evidencia a conectividade entre matemática e programação, conforme discutido por Santos e Menezes (2025), e aprofunda-se no desenvolvimento e aplicação de conceitos e competências matemáticas mediadas pela programação, como apontam Wu e Yang (2022) e Brandsaeter e Berge (2025); e (ii) a elaboração de atividades matemáticas com propostas de resolução mediadas pela linguagem *Python*, no âmbito do Ensino Básico, que servirão de base para a construção de uma sequência didática. Para isso, utiliza-se a Engenharia Didática em suas duas fases iniciais: análise preliminar e análise *a priori* (Artigue, 1988) orientando a construção e reflexão sobre a sequência didática, a qual se apoia na Teoria das Situações Didáticas (TSD) de Brousseau (1986; 2008; 2011).

Tendo isso em vista, o artigo não busca validar empiricamente a sequência didática, mas propor uma análise de caráter exploratória, voltada à projeção sobre como a integração entre programação e matemática pode ser planejada e conduzida em sala de aula. Nessa perspectiva, apresenta-se um quadro de referência para professores, concebido como suporte metodológico ao ensino da linguagem de programação no contexto da Educação Básica, em conteúdos como funções, sequências numéricas e geometria analítica, ao mesmo tempo em que orienta futuras investigações e contribui para a construção de propostas didáticas nessa vertente.

LINGUAGEM DE PROGRAMAÇÃO COMO APOIO NO ENSINO DE MATEMÁTICA

A utilização da linguagem de programação como instrumento no ensino da matemática tem-se consolidado como uma estratégia relevante e amplamente discutida em pesquisas educacionais (Misfeldt & Ejning-Duun, 2015; Saha, 2015, Rais & Xuezhi, 2023). Essa integração favorece não apenas conceitos abstratos, mas também o desenvolvimento de habilidades cognitivas. A matemática e programação compartilham fundamentos que se

conectam diretamente ao pensamento matemático e ao pensamento computacional. Entre essas conexões, destacam-se: o raciocínio lógico (Papert, 1980) necessário para estruturar e resolver problemas de forma coerente e analítica; a abstração (Wing, 2006; Simonot, Homps & Bonnot, 2012), que possibilita decompor estruturas complexas em partições que possibilitem o entendimento de forma mais simples; o pensamento algorítmico (Yusuf & Noor, 2024), fundamental para elaborar sequências lógicas de resolução; habilidades analíticas (Krulik & Rudnick, 1989), que permitem identificar padrões, decompor problemas e formular estratégias eficientes e, por fim, criatividade (Binotto, Maltempi, & Silva, 2023).

Um marco significativo nessa conexão entre matemática e programação foi dado por Seymour Papert (1928–2016), ao propor o uso de computadores como meio de promover metodologias ativas e engajamento dos estudantes. Em seus trabalhos, Papert orientou pesquisas que resultaram no desenvolvimento da linguagem LOGO. Seguindo essa linha, o *Scratch* (Resnick *et al.*, 2009) foi criado sob liderança de Mitchel Resnick, ex-orientando de doutorado de Papert, tornando a programação mais acessível e lúdica para crianças e jovens. A proposta de Papert dialoga diretamente com o Construtivismo, proposto a partir das contribuições de Jean Piaget (1986–1980) sobre a formação do pensamento lógico e simbólico na infância. Essa ideia construcionista tem consolidado bases para o uso da programação como ambiente de aprendizagem ativa em Matemática (Auccahuasi, 2018; Farid, Yahya & Atmasani, 2021), ao mesmo tempo em que abre espaço para o surgimento de abordagens como o pensamento computacional.

O pensamento computacional, conforme definido por Wing (2006), tem sido amplamente incorporado à Educação Matemática como um paradigma transformador. Estudos recentes (Ye, Liang, Ng & Chai, 2023) indicam que o pensamento computacional assim como o pensamento matemático atuam como uma ferramenta cognitiva essencial em etapas do processo de resolução de problemas, incluindo compreensão, planejamento, execução e reflexão (Zhang & Savard, 2023). Na prática, o uso de linguagens de programação e suas bibliotecas matemáticas possibilitam a exploração de conceitos matemáticos por meio de sequências, condições, visualizações dinâmicas e experimentações interativas com bibliotecas, como exemplo: a *Numpy*, para álgebra linear e estatística básica (Harris *et al.*, 2020); *Sympy*, direcionada para a matemática simbólica, derivadas, integrais (Meurer *et al.*, 2017) e a *Scipy*, desenvolvida para cálculos científicos avançados, como otimização, integração numérica, equações diferenciais (Virtanen *et al.*, 2020).

Estudos apontam tanto as potencialidades quanto os desafios da integração entre programação e ensino de Matemática. Uma investigação sobre livros didáticos suecos do ensino fundamental revelou “[...] um domínio da matemática como contexto para a programação, em vez da programação como ferramenta para a aprendizagem da matemática” (Nyman, Brating & Kilhamn, 2025, p. 1261). Essa constatação indica que, em muitos casos, a programação ainda é utilizada apenas como um recurso ilustrativo, sem explorar plenamen-

te seu potencial como mediadora da construção de conceitos matemáticos. Nesse sentido, Morais, Basso e Fagundes (2017, p. 470) argumentam que “[...] faz-se necessário o desenvolvimento de novas experiências em sala de aula, de metodologias de ensino de matemática para que mais educadores se aventurem a ensinar matemática a partir da programação”. Essa declaração reforça a ideia de que a integração entre ensino de programação e matemática deve ir além do uso instrumental.

Assim, mesmo a programação atuando como mediadora no ensino de Matemática, persistem desafios relevantes. Entre eles, destaca-se a formação docente, frequentemente considerada insuficiente para explorar o potencial dos recursos computacionais. Humble, Mozellius & Sallvin (2020, p. 309) observam que “Muitos professores consideram a programação divertida, mas não têm tempo para aprendê-la e implementá-la, e veem diferentes ferramentas de programação como complementares entre si e com oportunidades e desafios individuais”. Nessa mesma linha, Liu *et al.* (2024, p. 1) ressaltam que mesmo o pensamento computacional se tornando cada vez mais predominante na educação, “[...] muitos professores consideram um desafio integrá-lo à aprendizagem em sala de aula”.

Diante desse cenário, percebe-se que a integração entre programação e ensino de matemática exige mais do que a simples inserção de ferramentas digitais no contexto escolar. É necessário investir na formação inicial e continuada de professores, em abordagens metodológicas que articulem conceitos matemáticos e computacionais. Nesse sentido, a Engenharia Didática nesse artigo é utilizada para estruturar e investigar propostas que integrem a programação ao ensino da Matemática.

Portanto, na próxima seção, são discutidos alguns comandos básicos de programação em *Python*, explorando de que forma podem ser conectados a conceitos matemáticos. A partir disso, será proposta uma articulação entre tais comandos de linguagem de programação e os conteúdos de Matemática, evidenciando como a programação pode funcionar como meio para resolver problemas matemáticos. Essa relação será analisada através da Engenharia Didática, com foco em suas duas primeiras fases: a análise preliminar e a análise *a priori*, que permitirão fundamentar teoricamente a proposta e estruturar uma sequência didática. Tal sequência se apoiará nos aportes da Teoria das Situações Didáticas de Brousseau.

PROGRAMAÇÃO BÁSICA E MATEMÁTICA: ANÁLISE PRELIMINAR E A PRIORI

Com as discussões preliminares levantadas na seção anterior, as quais evidenciaram a relevância da integração entre programação e ensino de Matemática, bem como seus potenciais e desafios, a proposta aqui é seguir os princípios da Engenharia Didática, conforme a ampla literatura de Artigue (1985, 1988, 1996). Ampliam-se as discussões da Análise

Preliminar, voltada à identificação dos saberes matemáticos envolvidos, das dificuldades conceituais e técnicas, além dos aspectos curriculares que orientam a introdução da programação no ensino da matemática. Em seguida, procede-se com a Análise *a Priori*, que consiste na antecipação de estratégias de resolução, possíveis obstáculos de aprendizagem e validação do problema como instrumento de mediação didática.

Análise Preliminar

Como a proposta deste artigo demanda que se tenham noções básicas de linguagem de programação, esta etapa deve iniciar com um levantamento das experiências prévias do público-alvo em relação ao contato com a programação. Esse mapeamento é essencial para compreender o grau de conhecimento dos estudantes com conceitos elementares, como variáveis, operadores, estruturas condicionais e de repetição. Essa delimitação é relevante porque o objetivo deste estudo não é ensinar a linguagem de programação em si, mas sim direcionar o foco para a integração entre os fundamentos matemáticos e os códigos computacionais que os representam. Dessa forma, garante-se que o estudante não disperse esforços em aprender a sintaxe da programação, mas concentre-se em compreender como a Matemática pode ser traduzida em algoritmos computacionais.

A escolha dos problemas matemáticos discutidos nessa seção não se restringe a uma apresentação ilustrativa, mas constitui parte essencial da fundamentação metodológica da proposta. Os três problemas selecionados, envolvendo funções por partes e sequências numéricas (progressão aritmética e sequência de Fibonacci), foram definidos a partir de dois critérios centrais: (i) a relevância curricular, por se tratarem de conteúdos estruturantes da Matemática no Ensino Básico; e (ii) a possibilidade de articulação direta com comandos elementares de programação, como estruturas condicionais, repetições e classificações. Essa articulação amplia a viabilidade da proposta ao permitir que conceitos matemáticos sejam explorados de forma dinâmica, por meio da simulação.

Nessa perspectiva, os problemas assumem dupla função. Por um lado, possibilitam antecipar estratégias de resolução, erros recorrentes e obstáculos conceituais em nível conjectural; por outro, oferecem ao docente um quadro de referência para projetar práticas de ensino. Assim, mais do que propor exercícios pontuais, busca-se estruturar situações didáticas mediadas pelo professor, nas quais a linguagem *Python* atue como recurso metodológico e mediador do ensino de Matemática.

Portanto, para elucidar a integração entre programação em *Python* e conceitos matemáticos, são mostrados como os três tipos de problemas dialogam diretamente com as estruturas computacionais básicas. O primeiro aborda funções definidas por partes, nas quais o uso de condicionais permite estabelecer comparações e delimitar intervalos de validade. O segundo e o terceiro exploram sequências numéricas, a sequência de Fibonacci e

a progressão aritmética, destacando como as estruturas de repetição se tornam centrais na formulação algorítmica.

Problema 1. Decisões e funções por parte: nessa situação, é explorado um dos comandos mais elementares e fundamentais da linguagem de programação, o comando condicional if-else. Este recurso permite que decisões sejam tomadas com base em condições lógicas, sendo amplamente utilizado tanto na construção de algoritmos quanto na resolução de problemas matemáticos que envolvem classificações. O objetivo deste problema é articular esse comando com o conceito matemático de funções por partes, como indica a BNCC (Brasil, 2018) na habilidade EM13MAT404, que exige do aluno a capacidade de analisar diferentes comportamentos de uma função a depender do valor da variável.

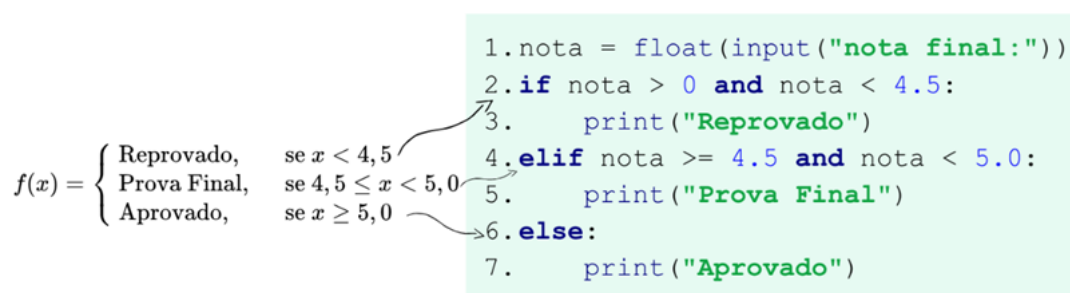
Neste problema, projeta-se uma situação em que o professor formaliza a expressão matemática, enquanto os estudantes poderiam, em um cenário didático, desenvolver um algoritmo computacional capaz de classificar um aluno com base em sua nota final, seguindo os critérios matemáticos previamente definidos. Para operacionalizar a classificação hipotética do algoritmo, consideram-se os seguintes critérios matemáticos definidos pelo professor, com notas expressas em uma casa decimal:

- Se a nota for inferior a 4,5, o aluno está *reprovado*;
- Se a nota estiver entre 4,5 (inclusive) e menor que 5,0, o aluno está em *prova final*;
- Se a nota for maior ou igual a 5,0, o aluno está *aprovado*.

A proposta apresenta uma situação hipotética em que os estudantes poderiam desenvolver o pensamento algorítmico, ao mesmo tempo em que reforçam a compreensão de conceitos matemáticos, como desigualdades, intervalos numéricos e funções por partes.

A Figura 1 ilustra essa integração entre domínio matemático e computacional em um cenário projetado: à esquerda, apresenta-se a expressão matemática que representa a função definida por partes; à direita, mostra-se uma possível implementação utilizando a estrutura condicional if-elif-else. Essa relação entre representação simbólica e codificação evidencia como, no contexto de ensino, a lógica computacional pode reproduzir o raciocínio matemático, permitindo que o professor projete atividades que conduzam os estudantes a estabelecer conexões entre condições matemáticas e blocos de decisão do algoritmo.

Figura 1 - Problema matemático: função por partes



Fonte: Desenvolvido pelos autores (2025)

No que se refere às hipóteses de dificuldades e possíveis erros, tanto do ponto de vista matemático quanto computacional, destacam-se alguns aspectos relevantes. Do ponto de vista matemático, uma dificuldade comum pode ser a escrita incorreta dos intervalos, especialmente no que se refere à distinção entre intervalos abertos e fechados. Muitos estudantes tendem a confundir os símbolos de desigualdade ($\geq$, $>$, $\leq$, $<$) linhas 2 e 4 da Figura 1, o que pode levá-los a definir incorretamente as faixas numéricas associadas a cada classificação (por exemplo, considerar que 4.5 pertence aos reprovados, quando na verdade marca o início da prova final)

Do ponto de vista computacional, um obstáculo na aprendizagem pode ser a tendência de os estudantes poderem atribuir ao bloco `else` um valor ou condição específica, como se este funcionasse de forma explícita. Entretanto, por definição, o `else` corresponde a uma condição complementar, isto é, sua execução ocorre unicamente quando nenhuma das condições anteriores são satisfeitas (linha 6 da Figura 1). Tal equívoco remete ao que Artigue (1989) denomina de obstáculos epistemológicos, uma vez que decorre da transposição de concepções prévias inadequadas para a interpretação de novos registros de representação. Antecipar essas dificuldades é essencial para análises preliminares, permitindo que o professor planeje intervenções adequadas, proponha questionamentos direcionados e possa estimular os estudantes a confrontarem diferentes representações da lógica condicional.

Problemas 2 e 3. Repetições e sequências numéricas: sequências numéricas é um tema obrigatório no ensino de Matemática, especialmente no estudo de Progressões Aritméticas (PA) e sequências definidas por recorrência, como a sequência de Fibonacci. Essa temática é proposta na BNCC (Brasil, 2018) através da habilidade EF08MA10 “Identificar a regularidade de uma sequência numérica ou figural não recursiva” e EF08MA11 “Identificar a regularidade de uma sequência numérica recursiva e construir um algoritmo por meio de um fluxograma”.

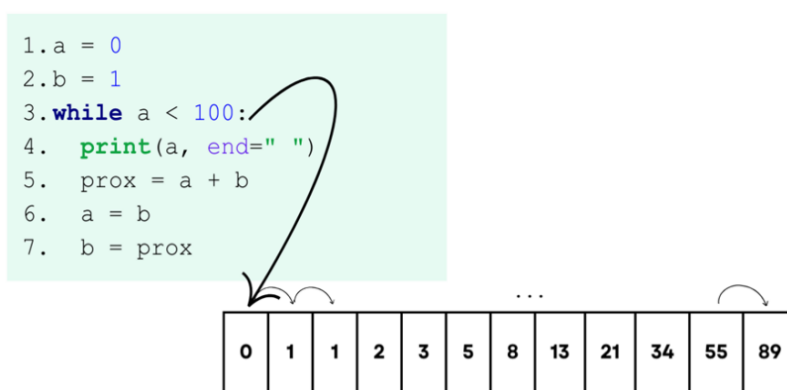
No contexto da programação, os comandos de repetição `for` e `while` oferecem meios de representar matematicamente a ideia de gerar termos sucessivos de uma sequência. Tal recurso pode ser explorado em propostas didáticas que busquem reforçar conceitos como regularidade, padrão, termo geral e recorrência. Nesse sentido, o professor pode propor dois algoritmos: (i) um que gera termos de uma PA a partir da fórmula do termo geral; e (ii) outro que gera a sequência de Fibonacci por meio de um processo iterativo que reproduz sua definição recorrente. Dessa forma, abre-se espaço para uma possibilidade de integração entre matemática e programação ao considerar:

- Sequências definidas por recorrência, como a de Fibonacci: $F_n = F_{n-1} + F_{n-2}$, que podem se alinhar à lógica do `while`, onde o próximo termo depende dos anteriores, atualizados em cada laço.

- Sequências com termo geral explícito, como na PA: $a_n = a_1 + (n - 1)r$, que podem se conectar com a estrutura `for`, onde cada iteração gera um novo termo.

Na Figura 2 é apresentado uma proposta de resolução computacional para o problema de Fibonacci. Entre as linhas 3 e 7, encontra-se a estrutura de repetição `while`, que corresponde ao processo recursivo: o valor atual é somado ao anterior para gerar o próximo termo da sequência. A variável `a`, inicialmente igual a 0 (linha 1), representa o primeiro termo, sendo atualizada a cada iteração dentro do *loop*. Essa correspondência permite discutir diferentes formas de modelar a sequência, seja pela fórmula fechada, seja pela dependência recursiva. O paralelismo entre a construção matemática e a computacional favorece o desenvolvimento do pensamento algorítmico e a compreensão da estrutura das sequências numéricas.

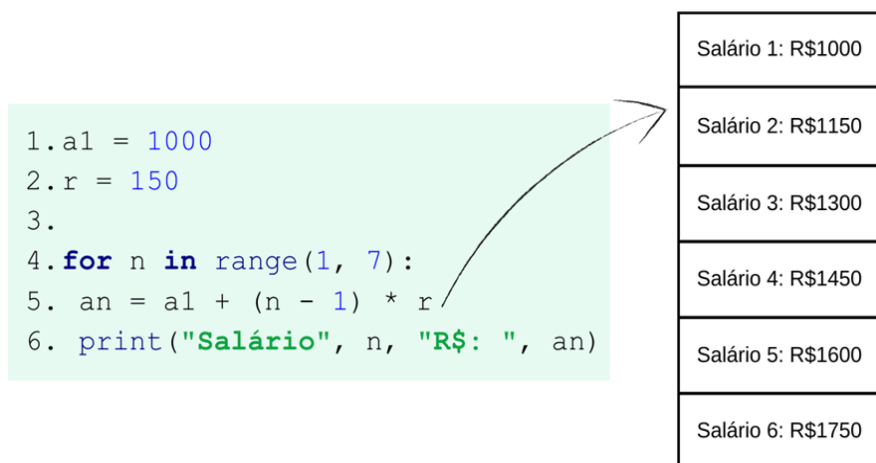
Figura 2 - Problema matemático: determinação de termos da sequência de Fibonacci



Fonte: Desenvolvido pelos autores (2025)

O próximo problema a ser explorado refere-se à construção de uma PA. Suponha que uma empresa deseje calcular os salários de seus estagiários. O primeiro estagiário recebe R\$ 1.000,00, e cada novo contratado recebe um acréscimo de R\$ 150,00 em relação ao anterior. A tarefa é exibir os salários dos 6 (seis) primeiros estagiários. Essa situação permite discutir tanto o conceito matemático da PA quanto sua modelagem computacional com laços de repetição. A Figura 3 apresenta uma proposta de resolução utilizando o comando `for` para repetir o cálculo por 6 vezes. Na linha 4 do código, o `for` inicia o *loop*, que percorre os valores de 1 a 6, representando o número do estagiário. Na linha 5, a equação da PA é implementada: o salário é calculado com base no valor inicial e na razão da sequência. A cada iteração, o programa exibe o número do estagiário e o respectivo salário.

Figura 3 - Problema matemático: determinar os valores da sequência de uma PA



Fonte: Desenvolvido pelos autores (2025)

Neste problema, podem surgir naturalmente dúvidas do tipo: por que o cálculo é feito de 1 a 6, se no comando for foi especificado o intervalo até 7? Neste momento, o professor deve lembrar aos estudantes que, em *Python*, a função `range(início, fim)` gera uma sequência que inclui o valor inicial, mas não inclui o valor final. Assim, ao escrever `range(1, 7)`, o laço percorre os valores 1, 2, 3, 4, 5 e 6, correspondendo ao intervalo $[1, 7)$. Esse detalhe, chamado de “intervalo semiaberto”, possibilita ao professor relembrar noções de conjuntos em intervalos abertos e fechados.

Considerando as discussões apresentadas até aqui, cabe frisar que os problemas discutidos nesta seção não correspondem a uma aplicação empírica realizada em sala de aula. Eles foram criteriosamente selecionados como parte da análise preliminar, cujo propósito é analisar os conteúdos matemáticos envolvidos, antecipar possíveis dificuldades conceituais e delimitar as condições didáticas para a futura sequência, tratando-se, portanto, de uma etapa de investigação teórico-projetiva. Portanto, essa fase preliminar encontra-se em consonância com Almouloud (2010) ao pontuar vertentes como:

“[...] analisar a estrutura matemática do conceito investigado [...] evidenciar os saberes (matemáticos) e os conhecimentos (matemáticos) [...] analisar as condições e fatores de que depende a construção didática efetiva da situação de ensino [...] considerar os objetivos específicos da pesquisa [...] definir a(s) questão(ões) da pesquisa e justificar as escolhas feitas” (Almouloud, 2010, p. 172-173).

Dessa forma, os problemas apresentados estabelecem o mapeamento conceitual e computacional necessário para compreender como a programação em *Python* pode servir de recurso metodológico no ensino de Matemática, além de oferecer uma base estruturante para a construção da sequência didática. Portanto, trata-se de apresentar de forma fundamentada, o quadro de referência que sustentará a elaboração da proposta didática.

Análise a Priori

A segunda etapa da Engenharia Didática refere-se à análise *a priori*, que consiste na proposta de antecipação das possíveis estratégias metodológicas, dificuldades e interações que os estudantes poderão apresentar diante de um problema (Artigue, 1988). Essa análise tem um papel fundamental no planejamento pedagógico, pois permite fundamentar, com base em conhecimentos didáticos e epistemológicos, as hipóteses sobre os processos de aprendizagem que se pretende promover. Assim, configura-se como um elemento essencial para a validação interna do dispositivo didático, contribuindo para que a intervenção seja coerente com os objetivos formativos e com os saberes mobilizados.

A proposta consiste em articular o desenvolvimento de conceitos da Geometria Analítica integrada com a linguagem de programação discutidas na análise preliminar, explorando práticas de resolução. Nesse cenário, enfatiza-se o plano cartesiano e a classificação de triângulos, mobilizando noções como sistema de coordenadas cartesianas, cálculo da distância entre dois pontos, condição de existência de triângulos e sua classificação segundo os lados. Assim, operações como a verificação da existência de triângulos (desigualdade triangular), a classificação segundo as medidas dos lados e a determinação de quadrantes no plano cartesiano podem ser traduzidas em estruturas de decisão e laços de repetição, favorecendo a construção de raciocínio lógico.

Esses conteúdos da Matemática são amplamente contemplados na BNCC (Brasil, 2018), bem como propostos no Ensino Médio, particularmente nas habilidades EM13MAT401 e EM13MAT402, que envolvem a análise e resolução de problemas geométricos em contextos diversos, com uso de representações gráficas e métricas. Além disso, essa proposta também dialoga com a habilidade EM13MAT405, que orienta o uso de conceitos iniciais de uma linguagem de programação na implementação de algoritmos em linguagem corrente e/ou matemática.

Ao propor uma atividade matemática mediada pela linguagem de programação nessa vertente, o docente deve estar atento aos possíveis erros e dificuldades que podem surgir, antecipando-os em suas hipóteses de ensino. É relativamente comum que os estudantes confundam intervalos abertos e fechados ao traduzirem condições para o código, esqueçam de verificar a existência do triângulo antes de classificá-lo ou invertam a lógica de comparações. Além disso, também podem aparecer dificuldades mais sutis, como a leitura literal de símbolos matemáticos que não têm correspondência direta em sintaxe computacional, como por exemplo: (Matemática: $(x_2 - x_1)^2$), *Python* ($(x_2 - x_1)**2$) ou até mesmo o excesso de confiança em resultados numéricos sem validação conceitual.

Apresentação da Situação Didática

As análises preliminares e *a priori* realizadas anteriormente forneceram os subsídios teóricos necessários para a construção da situação didática. A seleção e discussão dos problemas, envolvendo funções por partes e sequências numéricas, permitiram mapear os conceitos matemáticos relevantes, antecipar possíveis dificuldades de aprendizagem e identificar as estruturas computacionais indispensáveis para a integração entre programação em *Python* e conteúdos de Matemática.

Com base nesses elementos, foi estruturada a situação didática apresentada a seguir, fundamentada na Teoria das Situações Didáticas, proposta por Brousseau (2008). O problema de Geometria Analítica com Funções Condicionais (Quadro 1) foi escolhido por sintetizar os conhecimentos mapeados e possibilitar a articulação entre representações matemáticas e lógicas computacionais. A atividade é organizada em uma sequência que, ao mesmo tempo, constitui-se como um modelo que pode ser aplicável em contextos reais de ensino e como suporte para o desenvolvimento da análise *a priori*. Essa construção teórica permite antecipar como os estudantes poderiam interagir com o problema, favorecendo a reflexão docente acerca das condições didáticas que estimulam a produção e a validação do conhecimento.

Quadro 1 - Problema proposto para avaliação baseada na Teoria das Situações Didáticas

Disciplina / Área: Matemática / Geometria Analítica

Série: 2ª série do Ensino Médio

Conteúdo: Classificação de triângulos

Problema Proposto

Uma empresa contratou uma equipe para mapear um terreno em formato triangular. O engenheiro responsável forneceu as coordenadas dos três vértices do terreno: $A(0, 0)$, $B(2, 0)$ e $C(1, 2)$.

- Construa um algoritmo computacional que calcule a distância entre cada par de vértices (AB, BC, e CA).
- Determine em quais quadrantes os pontos A, B e C estão localizados no plano cartesiano e verifique se os três pontos formam um triângulo.
- Classifique o triângulo formado com base nas medidas dos lados (equilátero, isósceles ou escaleno).

Fonte: Desenvolvido pelos autores (2025)

Para resolver o problema proposto no Quadro 1, é preciso reconhecer que são necessárias diferentes noções matemáticas e computacionais. Do ponto de vista matemático, é necessário aplicar a fórmula da distância euclidiana para calcular as medidas entre os vértices do triângulo. No ambiente da programação, essa resolução requer o uso de estruturas condicionais, como `if` ou `match-case`, para verificar os quadrantes nos quais os pontos estão localizados e classificar o tipo de triângulo formado. Além disso, são mobilizadas operações com listas e conjuntos, que permitem identificar lados com medidas iguais, recurso essencial para determinar se o triângulo é escaleno, isósceles ou equilátero.

As Figuras 4, 5 e 6 apresentam uma proposta de implementação computacional com a qual o docente pode trabalhar com os estudantes. A Figura 4 apresenta a primeira etapa do algoritmo, que trata do cálculo da distância entre dois pontos no plano cartesiano. O programa inicia com a definição das coordenadas A, B e C (linhas 4 a 6) seguido da criação de uma função que calcula a distância entre dois pontos no plano cartesiano, utilizando a fórmula da distância euclidiana (linhas 9 e 10) e retorna esses valores em (linhas 13 a 15).

Figura 4 - Algoritmo computacional para cálculo da distância entre dois pontos

```

1. import math
2.
3. # Coordenadas dos pontos
4. x1, y1 = 0, 0
5. x2, y2 = 2, 0
6. x3, y3 = 1, 2
7.
8. # Função para calcular a distância entre dois pontos
9. def distancia(xa, ya, xb, yb):
10.     return round(math.sqrt((xb - xa)**2 + (yb - ya)**2), 2)
11.
12. # Cálculo das distâncias (lados do triângulo)
13. ab = distancia(x1, y1, x2, y2)
14. bc = distancia(x2, y2, x3, y3)
15. ca = distancia(x3, y3, x1, y1)
16.
17. print(f"Lados do triângulo: AB = {ab}, BC = {bc}, CA = {ca}")
18.

```

Fonte: Desenvolvido pelos autores (2025)

A Figura 5 concentra-se na verificação da condição de existência do triângulo e na classificação quanto ao tipo. A checagem inicial utiliza a desigualdade triangular, princípio fundamental segundo o qual a soma das medidas de dois lados deve sempre ser maior que a medida do terceiro, verificação realizada na (linha 20). Confirmada a validade, o código realiza a classificação do triângulo como escaleno, isósceles ou equilátero (linhas 24 a 31). Esse bloco estimula o raciocínio lógico e o pensamento condicional, ao conectar definições matemáticas a estruturas de decisão computacional.

Figura 5 - Algoritmo computacional para condição de existência e classificação quanto ao tipo de triângulo

```

19. # Verificar se formam um triângulo (soma de dois lados > terceiro)
20. if ab + bc > ca and ab + ca > bc and bc + ca > ab:
21.     print("Os pontos formam um triângulo.")
22.
23.     # Classificação do triângulo com match-case
24.     lados = len(set([ab, bc, ca]))
25.     match lados:
26.         case 1:
27.             print("Triângulo Equilátero")
28.         case 2:
29.             print("Triângulo Isósceles")
30.         case 3:
31.             print("Triângulo Escaleno")
32.     else:
33.         print("Os pontos não formam um triângulo.")
34.

```

Fonte: Desenvolvido pelos autores (2025)

Por fim, a Figura 6 apresenta a identificação dos quadrantes em que os pontos A, B e C estão localizados no plano cartesiano (linhas 36 a 50). Para isso, o programa utiliza uma estrutura condicional if-elif-else, estabelecendo regras de acordo com os sinais das coordenadas (x, y). Essa etapa possibilita consolidar a compreensão da relação entre localização espacial e coordenadas numéricas, reforçando os conceitos da Geometria Analítica.

Figura 6 - Algoritmo computacional que determina quais quadrantes encontram-se os pontos no plano cartesiano

```
35. # Determinação dos quadrantes
36. def quadrante(x, y):
37.     if x > 0 and y > 0:
38.         return "1° Quadrante"
39.     elif x < 0 and y > 0:
40.         return "2° Quadrante"
41.     elif x < 0 and y < 0:
42.         return "3° Quadrante"
43.     elif x > 0 and y < 0:
44.         return "4° Quadrante"
45.     elif x == 0 and y != 0:
46.         return "Sobre o eixo Y"
47.     elif y == 0 and x != 0:
48.         return "Sobre o eixo X"
49.     else:
50.         return "Na origem"
51.
52. print(f"Ponto A está no {quadrante(x1, y1)}")
53. print(f"Ponto B está no {quadrante(x2, y2)}")
54. print(f"Ponto C está no {quadrante(x3, y3)}")
```

Fonte: Desenvolvido pelos autores (2025)

DISCUSSÃO DA ANÁLISE DA SITUAÇÃO DIDÁTICA A PARTIR DAS QUATRO FASES DA TSD

A situação didática proposta é examinada à luz das quatro fases da TSD (Brousseau, 2008), a saber: ação, formulação, validação e institucionalização. Trata-se de uma análise de caráter teórico, que não descreve dados empíricos, mas projeta ações esperadas e possíveis interações dos estudantes diante do problema. Essas antecipações indicam o que se espera que eventualmente possa ocorrer em sala de aula, permitindo ao professor organizar previamente as condições didáticas e planejar intervenções que favoreçam a produção e a validação do conhecimento matemático.

Na fase da ação, hipoteticamente, espera-se que os estudantes, ao entrar em contato com o problema, possam explorá-lo por meio de tentativas, testes e conjecturas, mobilizando conhecimentos prévios. Do ponto de vista matemático, essa fase pode revelar dificuldades em lembrar ou aplicar a fórmula da distância entre dois pontos, omissões na verificação da condição de existência do triângulo (desigualdade triangular) ou dificuldades nos critérios de classificação dos triângulos. Do ponto de vista computacional, podem surgir erros no uso da função sqrt (cálculo de raiz quadrada), dificuldades em compreender estruturas condicionais, problemas ao lidar com entradas de dados e mensagens de erro. Nessa fase,

é fundamental que os estudantes tenham a oportunidade de manipular o problema livremente, promovendo o confronto com as dificuldades de aprendizagem.

Na formulação, é o momento que consiste na elaboração de hipóteses e estratégias por parte dos estudantes para resolver o problema. É nesse momento que ocorre a transição do agir para o refletir sobre o agir. Espera-se, em termos teóricos, que os estudantes passem a organizar suas ideias matemáticas e computacionais, podendo, por exemplo, explorar possibilidades de modularização do código (como a separação de uma função para calcular a distância ou a identificação dos quadrantes). Nesse movimento, são projetadas as estruturas primárias a serem desenvolvidas como aporte para as etapas seguintes, a definição dos dados de entrada e a formulação de classificações a partir dos lados do triângulo.

A validação na TSD refere-se ao momento em que os estudantes testam e verificam as soluções encontradas, avaliando sua adequação às condições do problema proposto. No contexto desta situação didática, essa validação pode ocorrer por meio da comparação entre os resultados obtidos pelo código computacional e aqueles gerados manualmente. Essa confrontação, em termos teóricos, favorece tanto a confiança na lógica algorítmica quanto na compreensão conceitual. Por exemplo, considerando os pontos A, B e C (linhas 4 a 6), projeta-se que a execução do programa imprima, na linha 17, as distâncias $AB = 2$ e $BC = CA = 2,24$. Diante desse resultado, seria esperado que o estudante identificasse que trata-se de um triângulo isósceles. Ao antecipar tal correspondência, entre a solução computacional e a dedução geométrica, delineia-se uma hipótese de validação da resposta.

Além disso, pode-se projetar, em caráter hipotético, o uso de testes de verificação no processo de resolução, como parte da própria dinâmica de programação. Essa possibilidade favorece a compreensão de erros e ajustes de estratégias, fortalecendo a autonomia no enfrentamento de problemas. Tal etapa, concebida em nível teórico, é relevante para consolidar o vínculo entre matemática e programação, ao evidenciar como a lógica algorítmica permite validar construções e explorar diferentes caminhos de resolução.

Por fim, a institucionalização ocorre quando o professor sistematiza os saberes construídos ao longo da situação, destacando os conceitos matemáticos e computacionais validados e formalmente reconhecidos. É nesse momento que se pode reafirmar a fórmula da distância entre pontos, os critérios de existência e classificação de triângulos, os conceitos de quadrantes, bem como os padrões de estruturação algorítmica utilizados na programação do problema. Cabe ao professor atuar como mediador, recuperando os conhecimentos mobilizados, apontando generalizações e valorizando a matemática como linguagem formal e a programação como ferramenta de modelagem e expressão lógica.

Portanto, o percurso metodológico delineado neste estudo contempla as duas primeiras fases da Engenharia Didática: a análise preliminar e a análise *a priori*, ambas de natureza teórica, centradas na formulação de hipóteses e na antecipação de possíveis interações.

Nesse movimento, buscou-se evidenciar um encadeamento coerente entre tais análises e a Teoria das Situações Didáticas. Assim, completa-se o elo fulcral entre análise, projeção e sistematização, garantindo que a proposta não seja composta por exemplos isolados, mas por um quadro de referência teórico-didático articulado, capaz de oferecer subsídios aos professores na elaboração de planos pedagógicos semelhantes.

CONSIDERAÇÕES FINAIS

Este artigo buscou explorar as conexões entre a linguagem de programação e a Matemática, destacando suas possibilidades no contexto educacional. O objetivo central foi discutir caminhos que permitissem responder à questão norteadora: como utilizar a linguagem *Python* para desenvolver propostas de ensino com diferentes problemas matemáticos, articulando conceitos computacionais e matemáticos por meio de uma abordagem fundamentada na Engenharia Didática? A análise, conceitualmente articulada nas duas primeiras fases da Engenharia Didática, mostrou que a programação pode atuar como um mediador eficaz entre a computação e o raciocínio matemático, favorecendo novas formas de representação, exploração e resolução de problemas.

A pesquisa evidenciou como os comandos de estruturas básicas em *Python* podem ser diretamente alinhados a conceitos matemáticos, especialmente no que diz respeito ao rigor analítico e à precisão dos resultados. Essa aproximação não apenas amplia as possibilidades de representação e resolução de problemas, mas também favorece uma aprendizagem mais significativa, em que o aluno compreende a matemática não como um conjunto isolado de fórmulas, mas como uma linguagem que dialoga em códigos computacionais.

Além dessa análise conectiva, a pesquisa buscou também explorar uma proposta de sequência didática por meio da TSD, demonstrando como comandos básicos de programação, quando articulados a conteúdos matemáticos, favorecem não apenas a compreensão conceitual, mas também o desenvolvimento de habilidades que às interligam. A sequência didática, estruturada e fundamentada em pesquisas preliminares, configurou-se como um referencial metodológico consistente para orientar a elaborar, analisar e avaliar a proposta metodológica do artigo. Nesse sentido, a integração entre tecnologia e matemática é apresentada não apenas como um recurso complementar, mas como um caminho promissor para enriquecer os processos de ensino-aprendizagem.

Mesmo com as conexões entre programação e matemática e suas potencialidades destacadas a partir da vertente hipotética nas primeiras fases da Engenharia Didática, algumas lacunas permanecem em aberto. A pesquisa não avançou para as fases experimentais e de validação, o que restringe a análise empírica da eficácia das atividades propostas em contextos reais de sala de aula. Dessa forma, não foi possível verificar, de maneira concreta, como os estudantes poderiam responder à situação didática planejada, nem identi-

car ajustes necessários para aprimorar a proposta. Esses limites apontam a necessidade de futuras investigações que contemplem a aplicação prática em ambientes escolares, com acompanhamento sistemático de resultados e análise do impacto no desenvolvimento do pensamento matemático e computacional dos estudantes.

| AGRADECIMENTOS

O primeiro autor agradece o apoio recebido para a realização da pesquisa desenvolvida durante o estágio de supervisão de pós-doutorado, vinculado ao Programa de Pós-Graduação em Ensino da Rede Nordeste de Ensino (RENOEN)–UFAL, sob a supervisão do Prof. Dr. Francisco Régis Vieira Alves, bolsista de produtividade em pesquisa do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), no período de 2020 a 2026, e Professor Titular do Departamento de Matemática e Física do IFCE.

| REFERÊNCIAS

- ARTIGUE, M. **Modélisation et reproductibilité en didactique des mathématiques**. IREM, Université Paris VII, 1985.
- ARTIGUE, M. Ingénierie didactique. **Recherches en didactique des mathématiques**, v. 9, n. 3, p. 281-308, 1988.
- ARTIGUE, M. **Épistémologie et didactique**. IREM de Paris, 1989.
- ARTIGUE, M. Engenharia didática. **Didáctica das Matemáticas**. Lisboa: Instituto Piaget, p. 193-217, 1996.
- ARTIGUE, M. Ingénierie didactique: quel rôle dans la recherche didactique aujourd'hui?. **Les dossiers des sciences de l'éducation**, v. 8, n. 1, p. 59-72, 2002.
- ARTIGUE, M. Didactic engineering in mathematics education. In: **Encyclopedia of mathematics education**. Cham: Springer International Publishing, 2020. p. 202-206. DOI: 10.1007/978-3-030-15789-0_44.
- AUCCAHUASI, W., SANTIAGO, G. B., NÚÑEZ, E. O., & SERNAQUE, F. Interactive online tool as an instrument for learning mathematics through programming techniques, aimed at high school students. In: **Proceedings of the 6th International conference on information technology: IoT and Smart City**. 2018. p. 70-76. DOI: <https://doi.org/10.1145/3301551.330158>.
- BINOTTO, R. R; MALTEMPI, M. V; DA SILVA, R. A. B. Potencialidades da programação em python para o desenvolvimento do pensamento criativo em matemática. **Zetetike**, v. 31, p. e023015-e023015, 2023. DOI: 10.20396/zet.v31i00.8672180.
- BRANDSAETER, A.; BERGE, R. L. Promoting mathematical competence development through programming activities. **Educational Studies in Mathematics**, v. 119, n. 2, p. 225-247, 2025. DOI: <https://doi.org/10.1007/s10649-024-10380-y>.

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

BROUSSEAU, G. Fondements et méthodes de la didactique des mathématiques. **Recherches en didactique des mathématiques (Revue)**, v. 7, n. 2, p. 33-115, 1986. Disponível em: <https://revue-rdm.com/1986/fondements-et-methodes-de-la/>. Acesso em: 15 jun. 2025.

BROUSSEAU, G. **Introdução ao estudo das situações didáticas: conteúdo e métodos de ensino**. São Paulo: Ática, 2008.

BROUSSEAU, G. Theory of didactical situations in mathematics. **Education Didactique**, v. 5, n. 1, p. 101–104, 2011.

CHONGCHITNAN, S. **Exploring University Mathematics with Python**. Cham: Springer, 2023.

FARID, M.; YAHYA, M.; ATMASANI, D. The correlation between digital simulation learning, basic programming, mathematics and students' knowledge: optimizing students' skill program for learning results. **Journal of Educational Science and Technology (EST)**, v. 7, n. 3, p. 298–306, 2021. DOI: 10.26858/est.v7i3.24538.

HARRIS, C. R., MILLMAN, K. J., VAN DER WALT, S. J., GOMMERS, R., VIRTANEN, P., COURNAPEAU, D., ... & OLIPHANT, T. E. Array programming with NumPy. **Nature**, v. 585, n. 7825, p. 357–362, 2020. DOI: <https://doi.org/10.1038/s41586-020-2649-2>.

HUMBLE, N.; MOZELIUS, P.; SÄLLVIN, L. Remaking and reinforcing mathematics and technology with programming: teacher perceptions of challenges, opportunities and tools in K-12 settings. **The International Journal of Information and Learning Technology**, v. 37, n. 5, p. 309–321, 2020. DOI:10.1108/IJILT-02-2020-0021.

KRULIK, S.; RUDNICK, J. A. **Problem solving: a handbook for senior high school teachers**. Boston: Allyn & Bacon, 1989.

LEÓN, J. M.; ROBLES, G.; ROMÁN-GONZÁLEZ, M. Code to learn: where does it belong in the K-12 curriculum? **Journal of Information Technology Education: Research**, v. 15, p. 283–303, 2016. DOI: <https://doi.org/10.28945/3521>.

LIU, Z., GEARTY, Z., RICHARD, E., ORRILL, C. H., KAYUMOVA, S., & BALASUBRAMANIAN, R. (2024). Bringing computational thinking into classrooms: a systematic review on supporting teachers in integrating computational thinking into K-12 classrooms. **International Journal of STEM Education**, v. 11, n. 1, p. 51, 2024. DOI: <https://doi.org/10.1186/s40594-024-00510-6>.

LOCKWOOD, J.; MOONEY, A. Computational thinking in education: where does it fit? A systematic literary review. **arXiv preprint arXiv:1703.07659**, 2017. DOI: <https://doi.org/10.48550/arXiv.1703.07659>.

LUTZ, M. **Learning Python: powerful object-oriented programming**. Sebastopol: O'Reilly

Media, 2013.

MACRIDES, E.; MILIOU, O.; ANGELI, C. Programming in early childhood education: a systematic review. **International Journal of Child-Computer Interaction**, v. 32, 100396, 2022. DOI: <https://doi.org/10.1016/j.ijcci.2021.100396>.

MISFELDT, M.; EJSING-DUUN, S. Learning mathematics through programming: an instrumental approach to potentials and pitfalls. In: **CERME 9 – Ninth Congress of the European Society for Research in Mathematics Education**. p. 2524–2530, 2015. Disponível em: <https://hal.science/hal-01289367v1>. Acesso em: 25 jul. 2025.

MORAIS, A. D.; BASSO, M. V. A.; FAGUNDES, L. C. Educação Matemática & Ciência da Computação na escola: aprender a programar fomenta a aprendizagem de matemática? **Ciência & Educação (Bauru)**, v. 23, n. 2, p. 455–473, 2017. DOI: <https://doi.org/10.1590/1516-731320170020011>.

MEURER, A., SMITH, C. P., PAPROCKI, M., ČERTÍK, O., KIRPICHEV, S. B., ROCKLIN, M., ... & SCOPATZ, A. SymPy: symbolic computing in Python. **PeerJ Computer Science**, v. 3, e103, 2017. DOI: <https://doi.org/10.7717/peerj-cs.103>.

NYMAN, R.; BRÅTING, K.; KILHAMN, C. Can programming support mathematics learning? An analysis of Swedish lower secondary textbooks. **International Journal of Mathematical Education in Science and Technology**, v. 56, n. 7, p. 1261–1279, 2025. DOI: <https://doi.org/10.1080/0020739X.2024.2329345>.

PAPERT, S. **Mindstorms: children, computers, and powerful ideas**. New York: Basic Books, 1980.

RAIS, D.; XUEZHI, Z. Human cognitive: learning mathematics through Python programming to support students' problem-solving skills. **Anatolian Journal of Education**, v. 8, n. 2, p. 85–98, 2023. DOI: <https://doi.org/10.29333/aje.2023.826a>.

RESNICK, M., MALONEY, J., MONROY-HERNÁNDEZ, A., RUSK, N., EASTMOND, E., BRENNAN, K., MILLNER, A., ROSENBAUM, E., SILVER, J., SILVERMAN, B., & KAFI, Y. **Communications of the ACM**, v. 52, n. 11, p. 60–67, 2009. DOI: <https://doi.org/10.1145/1592761.1592779>.

SAHA, A. **Doing math with Python: use programming to explore algebra, statistics, calculus, and more!** San Francisco: No Starch Press, 2015.

SANTOS, J. F. dos; MENEZES, I. F. de. Between facilities and learning: connecting computational thinking and mathematical skills through Python programming. **Revista Internacional de Pesquisa em Educação Matemática**, v. 15, n. 3, p. 1–19, 2025. <https://doi.org/10.37001/ripem.v15i3.4528>.

SIMONOT, M.; HOMPS, M.; BONNOT, P. Teaching abstraction in mathematics and computer science: a computer-supported approach with Alloy. In: **CSEDU (2)**. p. 239–245, 2012. DOI:10.5220/0003898302390245.

TODOS PELA EDUCAÇÃO (TPE). **Anuário brasileiro da educação básica 2024**. 2024. <https://anuario.todospelaeducacao.org.br/capitulo-12-infraestrutura.html>.

VIRTANEN, P., GOMMERS, R., OLIPHANT, T. E., HABERLAND, M., REDDY, T., COURNAPEAU, D., ... & VAN MULBREGT, P. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. **Nature Methods**, v. 17, n. 3, p. 261–272, 2020. <https://doi.org/10.1038/s41592-019-0686-2>.

VOSKOGLOU, M. G.; BUCKLEY, S. Problem solving and computational thinking in a learning environment. **arXiv preprint arXiv:1212.0750**, 2012. DOI:

<https://doi.org/10.48550/arXiv.1212.0750>.

WING, J. M. Computational thinking. **Communications of the ACM**, v. 49, n. 3, p. 33–35, 2006. DOI: <https://doi.org/10.1145/1118178.1118215>.

WU, W. R.; YANG, K. L. The relationships between computational and mathematical thinking: a review study on tasks. **Cogent Education**, v. 9, n. 1, 2098929, 2022. DOI: <https://doi.org/10.1080/2331186X.2022.2098929>.

YE, H.; LIANG, B.; NG, O. L.; CHAI, C. S. Integration of computational thinking in K-12 mathematics education: a systematic review on CT-based mathematics instruction and student learning. **International Journal of STEM Education**, v. 10, n. 1, p. 3, 2023. DOI: <https://doi.org/10.1186/s40594-023-00396-w>.

YUSUF, A.; NOOR, N. M. Modeling students' algorithmic thinking growth trajectories in different programming environments: an experimental test of the Matthew and compensatory hypothesis. **Smart Learning Environments**, v. 11, n. 1, p. 38, 2024. DOI: <https://doi.org/10.1186/s40561-024-00324-7>.

ZHANG, Y.; SAVARD, A. Defining computational thinking as an evident tool in problem-solving: comparative research on Chinese and Canadian mathematics textbooks. **ECNU Review of Education**, v. 6, n. 4, p. 677–699, 2023. DOI: <https://doi.org/10.1177/20965311231158393>.

ZIATDINOV, R.; VALLES Jr, J. R. Synthesis of modeling, visualization, and programming in GeoGebra as an effective approach for teaching and learning STEM topics. **Mathematics**, v. 10, n. 3, 398, 2022. DOI: <https://doi.org/10.3390/math10030398>.

Apêndice – Informação sobre el artículo

Historico editorial

Submetido: 22 de Junio de 2025.

Aprobado: 15 de Diciembre de 2025.

Publicado: 05 de Enero de 2026.

Como Citar — APA

Santos, J. F. dos., Santos, M. B. Q. de C. P. dos., & Alves, F. R. V. (2026). Programação Python e Ensino de Matemática: Uma Proposta Interdisciplinar com Engenharia Didática. *PARADIGMA*, XLVII(1), e2026004. <https://doi.org/10.37618/PARADIGMA.1011-2251.2025.e2026004.id1761>.

Como Citar — ABNT

SANTOS, Jailson França dos; SANTOS, Mercedes Bêta Quintano de Carvalho Pereira dos; ALVES, Francisco Régis Vieira. Programação Python e Ensino de Matemática: Uma Proposta Interdisciplinar com Engenharia Didática. **PARADIGMA**, Maracay, v. XLVII, n. 1, e2026004, Ene./Jun., 2026. <https://doi.org/10.37618/PARADIGMA.1011-2251.2026.e2026004.id1761>.

Conflicto de intereses

Nada que declarar.

Declaración de disponibilidad de datos

Todos los datos han sido presentados/generados en este artículo.

Derechos autorales

Los derechos de autor pertenecen a los autores, que conceden a revista **Paradigma** los derechos exclusivos de primera publicación. Los autores no serán remunerados por la publicación de sus artículos en esta revista. Los autores están autorizados a celebrar contratos adicionales por separado, para la distribución no exclusiva de la versión del artículo publicado en esta revista (por ejemplo, publicación en un repositorio institucional, en un sitio web personal, publicación de una traducción o como capítulo de un libro), con reconocimiento de autoría y primera publicación en esta revista. Los editores de la revista **Paradigma** tienen derecho a realizar ajustes textuales y adecuación normativas en este artículo.

Acceso libre

Este artículo es de acceso abierto (**Open Access**) y sin gastos de envío ni de procesamiento del artículo (**Article Processing Charges – APCs**). El acceso abierto es un amplio movimiento internacional que pretende proporcionar acceso en línea libre y gratuito a la información académica, como publicaciones y datos. Una publicación se define como de acceso abierto cuando no existen barreras financieras, legales o técnicas para acceder a ella; en otras palabras, cuando cualquiera puede leerla, descargarla, copiarla, distribuirla, imprimirla, investigarla o utilizarla en la educación o de cualquier otra forma dentro de los acuerdos legales.



Licencia de uso

Este artículo es licenciado con **Creative Commons Atribución-NoComercial-SinDerivadas 4.0 Internacional (CC BY-NC-ND 4.0)**. Esta licencia le permite compartir, copiar y redistribuir el artículo en cualquier medio o formato. La licencia no permite utilizar el material con fines comerciales ni adaptarlo, remezclarlo o transformarlo.



Comprobación de similitud

Este artículo fue sometido a una comprobación de similitud utilizando el software de detección de texto **iThenticate** de Turnitin, a través del servicio de **Similarity Check** de la Crossref.



Proceso de evaluación

Revisión por pares a doble ciego (**Double blind peer review**).

Editor en Jefe

Fredy E. González

Editores Asociados

Luis Andrés Castillo
Ivonne C. Sánchez S.

Publisher

Este artículo ha sido publicado en la revista **Paradigma** vinculada al Centro de Investigaciones Educativas Paradigma (CIEP) del Departamento del Componente Docente de la **Universidad Pedagógica Experimental Libertador** (Núcleo Maracay). La revista **Paradigma** publica artículos de carácter técnico-científico, derivados de estudios e investigaciones que sirvan de apoyo al desarrollo del conocimiento educativo, propiciando el diálogo entre los diferentes campos de la educación. Las ideas expresadas en este artículo son de los autores y no representan necesariamente la opinión del consejo editorial o de la universidad. En Brasil, la revista **Paradigma** obtuvo la calificación **Qualis A1** en la **Evaluación CAPES (2017-2020)**.

