UNIVERSIDADE DO ESTADO DA BAHIA

Autorização Decreto nº 9237/86. DOU 18/07/96. Reconhecimento: Portaria 909/95, DOU 01/08-95

DEPARTAMENTO DE EDUCAÇÃO

CAMPUS X – TEIXEIRA DE FREITAS

GRADUAÇÃO EM LICENCIATURA EM MATEMÁTICA

ATOS SILVA DE ARAÚJO

JULIANA SANTOS CONCEIÇÃO

RAFAELA SÁ DE JESUS

A APLICAÇÃO DA LINGUAGEM PYTHON NO ENSINO DE EXPRESSÕES ALGÉBRICAS E FUNÇÕES: SEQUÊNCIAS DIDÁTICAS PARA DOCENTES DE MATEMÁTICA

Teixeira de Freitas – BA

2025

UNIVERSIDADE DO ESTADO DA BAHIA

Autorização Decreto nº 9237/86. DOU 18/07/96. Reconhecimento: Portaria 909/95, DOU 01/08-95

DEPARTAMENTO DE EDUCAÇÃO

CAMPUS X – TEIXEIRA DE FREITAS

UNIVERSIDADE DO ESTADO DA BAHIA - UNEB
DEPARTAMENTO DE EDUCAÇÃO - CAMPUS X
COLEGIADO DE MATEMÁTICA

ATOS SILVA DE ARAÚJO

JULIANA SANTOS CONCEIÇÃO

RAFAELA SÁ DE JESUS

A APLICAÇÃO DA LINGUAGEM PYTHON NO ENSINO DE EXPRESSÕES ALGÉBRICAS E FUNÇÕES: SEQUÊNCIAS DIDÁTICAS PARA DOCENTES DE MATEMÁTICA

Trabalho de Conclusão de Curso apresentado ao Colegiado de Matemática, da Universidade do Estado da Bahia (UNEB), Campus X, para obtenção do título de Licenciado(a) em Matemática.

Orientador (a): Prof. Me. Francis Miller Barbosa Moreira
Coorientador (a): Prof. Esp. Deivson Leonardo Santana Da Silva

Teixeira de Freitas – BA

2025

UNIVERSIDADE DO ESTADO DA BAHIA

Autorização Decreto nº 9237/86. DOU 18/07/96. Reconhecimento: Portaria 909/95, DOU 01/08-95

DEPARTAMENTO DE EDUCAÇÃO

CAMPUS X – TEIXEIRA DE FREITAS

DEDC - CAMPUS X
Departamento
de Educação

UNEB
UNIVERSIDADE DO
ESTADO DA BAHIA

ATOS SILVA DE ARAÚJO

JULIANA SANTOS CONCEIÇÃO

RAFAELA SÁ DE JESUS

A APLICAÇÃO DA LINGUAGEM PYTHON NO ENSINO DE EXPRESSÕES ALGÉBRICAS E FUNÇÕES: SEQUÊNCIAS DIDÁTICAS PARA DOCENTES DE MATEMÁTICA

Trabalho de Conclusão de Curso apresentado ao Colegiado de Matemática, da Universidade do Estado da Bahia (UNEB), Campus X, para obtenção do título de Licenciado(a) em Matemática.

Comissão Examinadora

_____

Prof. Me. Francis Miller B. Moreira

_____

Prof. Esp. Deivson Leonardo Santana Da Silva

_____

Prof. Esp. Pollyana Soares De Novaes

Resultado: _____

Teixeira de Freitas – BA, _____ de _____ de 2025

AGRADECIMENTOS

A realização deste trabalho só foi possível graças ao apoio, incentivo e presença de muitas pessoas, às quais expressamos a mais profunda gratidão. Agradecemos primeiramente a Deus, por nos conceder saúde, força e sabedoria ao longo desta jornada. Aos nossos familiares, que estiveram ao nosso lado em todos os momentos, com palavras de encorajamento, amor incondicional e apoio constante.

Aos nossos orientadores, Professor Mestre Francis Miller Barbosa Moreira e Professor Especialista Deivson Leonardo Santana Da Silva pela dedicação, paciência e contribuições fundamentais para o desenvolvimento deste trabalho. Suas orientações foram essenciais para que pudéssemos amadurecer academicamente e pessoalmente. Agradecemos também a Professora Mestre Tatiana Dias que ofereceu total apoio e incentivo na construção da pesquisa.

Aos professores e colegas do curso de Licenciatura em Matemática, que contribuíram para nossa formação com saberes, experiências e amizades que levaremos para a vida. Aos amigos que nos acompanharam nos desafios da graduação, pelo apoio, compreensão e momentos de descontração que tornaram essa caminhada mais leve e significativa. À Universidade do Estado da Bahia – UNEB, pelo espaço de aprendizado e crescimento intelectual.

Nenhuma construção é feita sozinha. Cada página deste trabalho carrega um pouco de cada pessoa que nos inspirou, apoiou e acreditou no nosso caminho — a vocês, nossa eterna gratidão.

RESUMO

Este trabalho apresenta uma proposta de sequência didática para o ensino de expressões algébricas e funções de primeiro e segundo grau, utilizando a linguagem de programação *Python* como recurso didático e a metodologia da Aprendizagem Baseada em Problemas (ABP) como eixo metodológico. Fundamenta-se nos pressupostos do Construcionismo, propostos por Seymour Papert, e nos princípios da Base Nacional Comum Curricular (BNCC), que orientam a inserção das tecnologias digitais no ambiente escolar. A escolha do tema parte da constatação, durante experiências em estágios e monitorias, de dificuldades recorrentes dos estudantes com conteúdos matemáticos abstratos e da necessidade de abordagens pedagógicas que dialoguem com a realidade digital vivenciada por eles. Nesse contexto, a programação surge como instrumento mediador do pensamento matemático, potencializando o raciocínio lógico, a autonomia e a resolução de problemas. O trabalho propõe um guia introdutório à linguagem Python e uma sequência didática estruturada em etapas, com propostas de atividades contextualizadas, visando tornar a aprendizagem mais significativa e investigativa. Embora a proposta ainda não tenha sido aplicada empiricamente, configura-se como uma contribuição teórico-prática para professores da Educação Básica e para a formação inicial em cursos de Licenciatura em Matemática. A conclusão destaca a importância da modernização das práticas pedagógicas, do domínio das tecnologias pelos docentes e do uso da programação como aliada no processo de ensino e aprendizagem da Matemática, abrindo caminhos para estudos futuros, com aprofundamento quantitativo e aplicação em contextos reais.

Palavras-chave: Ensino de Matemática. Linguagem Python. Sequência Didática. ABP. Construcionismo.

LISTA DE QUADROS E TABELAS

QUADROS

TABELAS

SUMÁRIO

7. 733

1. CONSIDERAÇÕES INICIAIS

O presente trabalho tem como foco a elaboração de sequências didáticas para o ensino de expressões algébricas e funções de primeiro e segundo grau, com apoio da linguagem de programação *Python* e fundamentada na metodologia da Aprendizagem Baseada em Problemas (ABP). A proposta se estrutura a partir da Base Nacional Comum Curricular (BNCC) e visa fornecer suporte ao trabalho do docente, tanto os que estão em formação quanto os que já atuam em sala de aula, por meio de um guia introdutório disposto no capítulo 4 e de quatro sequências didáticas estruturadas, possibilitando a inserção de práticas computacionais no ensino de Matemática.

Como embasamento teórico, a pesquisa utiliza de apoio central a perspectiva do *Construcionismo*, proposta por Seymour Papert (1980), que compreende o aprendizado como um processo ativo, em que o estudante constrói o conhecimento a partir da criação de artefatos significativos. No contexto educacional, a programação é vista por Papert como uma linguagem poderosa para explorar conceitos matemáticos, estimular a autonomia intelectual e favorecer a aprendizagem significativa.

1.1. TEMA E DELIMITAÇÃO DO TEMA

Este trabalho tem como tema A Aplicação da Linguagem *Python* no Ensino de Expressões Algébricas e Funções: sequências didáticas para docentes de Matemática. No qual, o foco é a elaboração de sequências didáticas voltadas ao ensino de expressões algébricas e funções de primeiro e segundo grau, destinada a professores de Matemática. A proposta está delimitada à utilização da linguagem de programação *Python* como ferramenta pedagógica complementar, considerando especialmente docentes em formação inicial ou em exercício que possuem ou não familiaridade com recursos computacionais. A abordagem está fundamentada na Base Nacional Comum Curricular (BNCC) e utiliza a metodologia da Aprendizagem Baseada em Problemas (ABP) como estratégia didática. Importa destacar que o estudo se limita à construção do material didático, não abrangendo a fase de aplicação prática, por parte dos autores, em contextos escolares.

1.2. PROBLEMATIZAÇÃO

O processo educacional tem sido profundamente impactado pelas transformações tecnológicas das últimas décadas, especialmente com a consolidação das Tecnologias Digitais da Informação e Comunicação (TDIC's). A pandemia da Covid-19, iniciada em 2020, intensificou a transformação digital no ensino, ao impor, de maneira abrupta, a adoção do ensino remoto e a virtualização das práticas pedagógicas. Como destaca Moran (2021), esse cenário exigiu dos educadores uma rápida adaptação às tecnologias digitais, muitas vezes sem o preparo prévio necessário, evidenciando fragilidades e oportunidades na formação docente. Esse contexto evidenciou a necessidade de integrar as TDIC's à formação docente e ao cotidiano escolar, impulsionando discussões sobre as competências exigidas do professor na contemporaneidade.

No ensino da Matemática, os desafios se tornam ainda mais evidentes. Como apontam pesquisadores como Skovsmose (2000) e D'Ambrósio (2002), a prática pedagógica tradicional, centrada em metodologias expositivas e no ensino conteudista, dificulta o engajamento dos estudantes e a compreensão conceitual dos conteúdos. Além disso, essa abordagem frequentemente se mostra desconectada das inovações tecnológicas e das demandas da sociedade contemporânea, limitando o potencial das tecnologias digitais de promover um aprendizado mais significativo e contextualizado (BRASIL, 2018; MORAN, 2021).

A Base Nacional Comum Curricular (BNCC) já reconhece a importância do pensamento computacional e da cultura digital no currículo da educação básica, incluindo nas aulas de Matemática (BRASIL, 2018). No entanto, propostas práticas que orientem o professor quanto ao uso efetivo dessas ferramentas ainda são escassas. Boa parte dos docentes formadores pertence a gerações cuja formação ocorreu em contextos anteriores à difusão massiva das TDIC's. Esse fator contribui para a limitada inserção de experiências formativas que integrem conteúdos matemáticos com linguagens computacionais, como o *Python*. Como resultado, há licenciandos que concluem seus cursos com pouco ou nenhum contato com essas ferramentas, o que limita sua atuação futura no ensino básico.

Em muitos cursos de Licenciatura em Matemática, especialmente em instituições públicas, os currículos seguem prevalecendo nas abordagens tradicionais de ensino, com foco em aulas teóricas e pouca articulação com práticas interativas e tecnológicas. Severino (2007, p. 97) observa que "o ensino ainda se organiza sob a forma de mera

transmissão do saber, cabendo ao professor a função de expositor e ao aluno a de receptor passivo". Essa lógica pedagógica encorpada em modelos tradicionalistas contribui para uma formação inicial distante das demandas atuais da educação, dificultando a apropriação crítica das TDIC's como aliadas no ensino da Matemática.

A necessidade de uma formação mais integrada às tecnologias ficou ainda mais evidente durante a pandemia. Kenski (2012, p. 58) destaca que "o professor que atua nas escolas contemporâneas precisa ser, além de um especialista em sua área, um conhecedor e usuário das tecnologias da informação, integrando-as de forma crítica e criativa às suas atividades pedagógicas". A ausência dessa competência compromete o desenvolvimento de práticas mais atrativas e eficazes em sala de aula.

Ao ingressar na docência, muitos professores replicam metodologias tradicionais e utilizam softwares ultrapassados, sem explorar as possibilidades de linguagens mais atuais como o *Python*. Isso reduz o potencial de inovação no ensino e o engajamento dos alunos, que vivem imersos em uma cultura digital. A Matemática, nesse cenário, permanece muitas vezes desconectada da realidade dos estudantes, pois apesar de estar presente em diversas áreas no nosso dia a dia, a conexão dos conteúdos com o que é real ainda está no campo de abstração para boa parte dos discentes.

Papert (1980), ao propor o *Construcionismo*, afirma que a aprendizagem se torna mais significativa quando o aluno está envolvido na construção de algo com sentido pessoal. Para ele, ao programar, o aluno manipula ideias matemáticas de modo concreto, favorecendo a compreensão conceitual e a motivação. Assim, a programação não é apenas uma habilidade técnica, mas uma linguagem de expressão e de resolução de problemas. Nesse sentido, propor ao estudante o uso da tecnologia para resolver desafios cotidianos é também promover um aprendizado mais crítico e ativo.

Essa visão motiva o presente trabalho, que propõe uma abordagem acessível, prática e fundamentada para o uso do *Python* no ensino de Matemática. O objetivo é oferecer aos professores uma sequência didática que integre conteúdos como expressões algébricas e funções de primeiro e segundo grau a projetos simples de programação, ampliando as possibilidades pedagógicas e conectando o conteúdo escolar ao universo digital dos alunos.

1.3. JUSTIFICATIVAS

A proposta deste trabalho emerge de uma necessidade real e contemporânea: tornar o ensino da Matemática mais conectado à cultura digital e às transformações tecnológicas do século XXI. Embora a Base Nacional Comum Curricular (BNCC) estabeleça, entre suas competências gerais, a promoção do pensamento computacional e da cultura digital, tais diretrizes ainda enfrentam barreiras concretas em sua efetiva implementação, especialmente no campo da Matemática (Brasil, 2018).

A linguagem Python, nesse cenário, surge como um recurso didático promissor. Por possuir uma sintaxe simples, ser gratuita, versátil e fortemente aplicada em contextos matemáticos e científicos, a linguagem Python possibilita aos professores criar situações de aprendizagem mais dinâmicas, contextualizadas e investigativas. Papert (1980, p. 19), ao defender o uso de linguagens de programação no ensino, afirma que "aprender a programar é, em essência, aprender a pensar", reforçando a ideia de que o pensamento computacional pode ser uma poderosa ferramenta de construção de conhecimento matemático.

Do ponto de vista da prática docente, integrar a linguagem de programação ao ensino da Matemática pode potencializar o raciocínio lógico, a resolução de problemas e a criatividade dos estudantes. Segundo Valente (1999, p. 7), "as tecnologias devem permitir a reconstrução do conhecimento por meio da ação, da reflexão e da experimentação, favorecendo o desenvolvimento de competências cognitivas complexas". Assim, o uso pedagógico do *Python* não apenas viabiliza a aprendizagem de conteúdos matemáticos, como também amplia as capacidades cognitivas dos estudantes.

Esse projeto também é justificável sob a ótica da democratização do acesso ao conhecimento tecnológico. Ao construir um material didático acessível e estruturado, como a sequência didática juntamente a um guia introdutório sobre o uso do *Python* com base na BNCC e na metodologia da Aprendizagem Baseada em Problemas (ABP), pretende-se oferecer suporte prático aos professores que desejam inovar, mas que talvez não tiveram formação tecnológica durante sua trajetória acadêmica. A partir disso, educar na era digital exige não apenas conhecer tecnologias, mas reinventar as práticas pedagógicas a partir delas.

Por fim, a justificativa deste trabalho também se estende ao benefício direto dos estudantes da Educação Básica. Ao terem contato com a linguagem *Python* em contextos significativos, esses alunos poderão desenvolver competências essenciais para o século

XXI, tais como pensamento lógico, autonomia, criatividade e resolução de problemas, ao mesmo tempo em que aprofundam sua compreensão dos conteúdos matemáticos curriculares. Visando não apenas propor uma sequência didática estruturada, mas também colaborar com um movimento maior de transformação da prática docente em Matemática, aproximando o ensino da realidade tecnológica atual e ampliando as possibilidades formativas para professores e alunos.

1.4. OBJETIVOS

O presente trabalho tem como objetivo geral desenvolver um guia introdutório e sequências didáticas por meio da metodologia de ensino Aprendizagem Baseada em Problemas (ABP), apoiadas a linguagem *Python* para auxiliar docentes de Matemática no ensino de Expressões Algébricas e Funções de primeiro e segundo grau, promovendo a inserção das tecnologias e da programação no contexto educacional, em conformidade com a BNCC. Ademais, objetiva-se demonstrar a aplicabilidade da programação na resolução de problemas matemáticos, sejam eles conteúdos específicos ou aplicados ao cotidiano, incentivando o uso de ferramentas tecnológicas como potencializadoras da aprendizagem dos discentes, e, por fim, estimular a modernização do ensino no curso de Licenciatura em Matemática, promovendo o uso de linguagens de programação como recurso didático complementar às abordagens tradicionais.

1.5. ESTRUTURA DO TRABALHO

Este trabalho está organizado em sete capítulos, distribuídos de forma a conduzir o leitor por uma compreensão progressiva do tema abordado. No Capítulo 1, são apresentadas as considerações iniciais, tema e delimitação do tema, a problematização, a justificativa, os objetivos e a própria estrutura do trabalho. O Capítulo 2, a tecnologia no ensino da Matemática, discute o ensino e aprendizagem da Matemática no Ensino Médio e no Ensino Superior, o auxílio das novas tecnologias na construção do conhecimento matemático e o papel das instituições de ensino diante do uso das tecnologias em sala de aula. Durante o capítulo 3, o trabalho discorre sobre o campo da linguagem de programação, desde a programação e o pensamento computacional no ensino da Matemática, raciocínio lógico e o computador e algoritmo, tópicos fundamentais para entender a base de qualquer linguagem associada à Matemática.

No capítulo 4, há a introdução direta à linguagem científica *Python*, um capítulo de suma importância, visto que o mesmo funciona como um "guia introdutório" com a sugestão de alguns projetos pilotos a serem feitos em sala de aula, antes da aplicação das sequências didáticas em si, introduzindo o que é a linguagem, software, diferenças básicas e plataforma de ensino, introdução da linguagem com exemplos e por fim, a construção desses pilotos como suporte inicial. Quanto à metodologia é apresentada no capítulo 5, trazendo o tipo da pesquisa, a metodologia utilizada Aprendizagem Baseada em Problemas – ABP e a sequência didática. No capítulo 6, dedicado especialmente às sequências didáticas produzidas como produto final do trabalho, há a divisão em: expressões algébricas, função de primeiro grau e função e segundo grau. Por fim, no capítulo 7, as considerações finais, destacando as principais conclusões e possíveis desdobramentos da pesquisa.

2. TECNOLOGIA NO ENSINO DA MATEMÁTICA

2.1. ENSINO E APRENDIZAGEM DA MATEMÁTICA NO ENSINO MÉDIO E ENSINO SUPERIOR

O ensino da Matemática, tanto no Ensino Médio quanto no Ensino Superior, especialmente na formação inicial de professores, enfrenta desafios históricos e estruturais que impactam diretamente o processo de aprendizagem dos estudantes. A disciplina, frequentemente percebida como abstrata e descontextualizada, é marcada por métodos tradicionais de ensino, predominantemente expositivos, abstratos e teóricos, que nem sempre favorecem a construção significativa do conhecimento matemático. Essas afirmações, podem ser presenciadas tanto no Ensino Básico quanto no Ensino Superior, experiência relatada pelos autores durante a graduação e experiências de estágio em sala de aula.

No Ensino Médio, os conteúdos curriculares de Matemática são, em geral, organizados de forma fragmentada, priorizando a memorização de fórmulas e procedimentos em detrimento da compreensão conceitual e do desenvolvimento do raciocínio lógico. Segundo Fiorentini e Miorim (1990), a prática docente tende a seguir um modelo transmissivo, com baixa articulação entre teoria e prática e pouca exploração de estratégias que promovam a investigação e a autonomia do aluno. Isso contribui para a percepção da Matemática como uma disciplina difícil e inacessível, gerando desmotivação e evasão escolar.

Esse panorama é reforçado pelas formas de avaliação predominantes, centradas em provas objetivas ou discursivas que privilegiam a reprodução mecânica do conteúdo, sem considerar as diferentes formas de aprender. Como destaca Libâneo (2013), os instrumentos avaliativos deveriam ir além da mensuração do desempenho e atuar como recursos mediadores do processo ensino-aprendizagem, permitindo ao professor compreender as necessidades e avanços dos estudantes.

No contexto do Ensino Superior, especialmente nos cursos de Licenciatura em Matemática, observa-se uma distância significativa entre a formação teórica oferecida e as demandas contemporâneas da prática docente. Embora os cursos contemplem disciplinas fundamentais da Matemática, como Álgebra, Geometria e Cálculo, muitas vezes negligenciam aspectos práticos relacionados ao uso de tecnologias digitais e à didáticas inovadoras. Segundo Kenski (2012), é imprescindível que os cursos formadores

de professores incorporem as tecnologias como elementos estruturantes da aprendizagem, e não apenas como acessórios.

A falta de familiaridade com recursos computacionais e linguagens de programação, tanto por parte dos alunos quanto dos professores formadores, representa uma das maiores lacunas na formação docente atual. Papert (1980) já defendia, há décadas, que o uso de computadores poderia revolucionar a aprendizagem matemática, tornando-a mais significativa e exploratória. No entanto, ainda hoje, muitas instituições resistem à adoção de ferramentas tecnológicas como parte integrante do currículo da licenciatura.

Além disso, a fragmentação entre as disciplinas pedagógicas e os conteúdos específicos da Matemática dificulta uma formação integrada, capaz de preparar o futuro professor para enfrentar os desafios reais da sala de aula. Severino (2007) destaca que a formação docente deve estar alinhada a uma visão crítica da prática educacional, promovendo a articulação entre teoria, prática e realidade social.

Dessa forma, tanto no Ensino Médio quanto no Ensino Superior, o ensino da Matemática ainda carece de abordagens que promovam o protagonismo estudantil, a contextualização dos conteúdos e o uso de recursos que ampliem as possibilidades de compreensão e aplicação do conhecimento. A ausência de tecnologias em sala de aula, a resistência a novos métodos e as limitações na formação inicial dos professores contribuem para manter um ensino centrado na repetição e na rigidez metodológica.

Frente a esse cenário, a inserção de propostas que integram a linguagem de programação ao ensino de Matemática surge como uma alternativa promissora, capaz de desenvolver habilidades cognitivas relevantes, como pensamento computacional, resolução de problemas e autonomia intelectual, além de já inserir o aluno às demandas do meio profissional. No mundo atual as empresas exigem cada vez mais em seus currículos, solicitando que os candidatos entendam sobre tecnologias e saibam no mínimo o básico de uma linguagem de programação. Essa abordagem, além de atender às competências previstas na Base Nacional Comum Curricular (BRASIL, 2018), também contribui para a modernização dos processos formativos e para a valorização do papel do professor como mediador ativo da aprendizagem.

Outro aspecto relevante a ser considerado diz respeito à formação continuada dos docentes, especialmente os que atuam no Ensino Médio. Muitos professores encontram-se afastados das discussões mais recentes sobre metodologias ativas, uso de tecnologias e inovação didática. Como destaca Moran (2021, p. 25), “o professor precisa aprender

continuamente, mudar seus papéis, integrar as tecnologias e desenvolver metodologias mais participativas". Essa postura reflexiva e investigativa é essencial diante das transformações da cultura digital. A ausência de políticas efetivas de formação continuada, contudo, dificulta esse movimento, mantendo modelos tradicionais de ensino mesmo em contextos em que alunos e escolas já estão imersos em ambientes digitais.

Além disso, é importante considerar o perfil dos estudantes contemporâneos. Nativos digitais, esses alunos estão em constante contato com múltiplas linguagens e plataformas tecnológicas, o que exige da escola uma adaptação não apenas nos recursos utilizados, mas também nas formas de ensinar. Como aponta Moran (2021, p. 22), "a cultura digital exige que a aprendizagem seja mais ativa, aberta, conectada, colaborativa e híbrida, com ênfase em projetos, desafios e experimentações". Assim, é essencial que o ensino de Matemática dialogue com essa nova lógica cognitiva, rompendo com a linearidade e a rigidez que muitas vezes caracterizam as aulas.

Outro ponto ainda pouco explorado nos cursos de licenciatura é o desenvolvimento de competências interdisciplinares, que articulem a Matemática com outras áreas do conhecimento e com problemas do cotidiano. A integração de projetos que envolvam programação, por exemplo, pode contribuir para uma visão mais ampla e aplicada da disciplina, favorecendo a compreensão de sua utilidade social e prática. Segundo Moran (2015), a escola do século XXI precisa preparar os alunos para pensar criticamente, resolver problemas reais e atuar em um mundo em constante transformação — desafios que exigem inovação no currículo, nas metodologias e na própria formação docente.

É interessante ressaltar que a inserção de tecnologias como a linguagem *Python* não pretende substituir o conteúdo matemático, mas sim enriquecê-lo. A proposta é utilizar essas ferramentas como pontes entre a abstração matemática e a experiência concreta do aluno, criando oportunidades para a experimentação, a simulação de situações e o desenvolvimento do pensamento lógico de forma mais interativa e atrativa. Como defende Papert (1980), "o computador pode ser uma incubadora de ideias matemáticas", proporcionando um ambiente fértil para o aprendizado criativo e autônomo.

2.2. O AUXÍLIO DAS NOVAS TECNOLOGIAS NA CONSTRUÇÃO DO CONHECIMENTO MATEMÁTICO

O avanço das tecnologias digitais tem provocado profundas transformações nas formas de ensinar e aprender, especialmente no campo da Matemática, tradicionalmente associado a métodos expositivos e práticas pouco interativas. As novas tecnologias, quando utilizadas de forma pedagógica, podem contribuir significativamente para a construção ativa do conhecimento, potencializando o raciocínio lógico, a resolução de problemas e a modelagem de situações reais.

Segundo Papert (1980), um dos pioneiros no estudo da relação entre tecnologia e educação Matemática, "o computador pode ser usado não apenas como uma ferramenta de cálculo, mas como uma incubadora de ideias matemáticas". Para o autor, ambientes computacionais bem estruturados favorecem a aprendizagem significativa, permitindo que o aluno construa o conhecimento por meio da experimentação, da simulação e da interação com objetos matemáticos digitais.

Do ponto de vista pedagógico, Gil (2019) destaca que o uso de tecnologias na educação precisa ser acompanhado de uma intencionalidade didática clara. Não basta apenas inserir ferramentas digitais em sala de aula; é preciso que elas estejam articuladas com os objetivos de aprendizagem e com os conteúdos curriculares. No caso da Matemática, essa articulação pode se concretizar, por exemplo, por meio do uso de linguagens de programação como o *Python*, que permitem criar algoritmos, resolver equações, construir gráficos e modelar funções, tornando os conceitos abstratos mais concretos e acessíveis.

As tecnologias digitais também favorecem uma abordagem mais construtivista da educação matemática, em que o estudante deixa de ser um receptor passivo e passa a atuar como sujeito ativo do seu processo de aprendizagem. Como afirma Severino (2016), a educação contemporânea deve valorizar a construção do saber pelo próprio educando, mediante sua atividade intelectual. Esse princípio é perfeitamente compatível com o uso de recursos computacionais, que exigem a manipulação ativa das informações e a tomada de decisões em tempo real.

Além disso, o uso de *softwares* e ambientes de programação em sala de aula permite ao professor inovar na forma de avaliar os alunos. Em vez de avaliações exclusivamente baseadas na reprodução de cálculos e fórmulas, pode-se propor projetos, simulações e desafios computacionais que exijam a aplicação prática dos conhecimentos

matemáticos. Segundo Moran (2015), as tecnologias criam condições para uma avaliação mais processual, diagnóstica e formativa, rompendo com a rigidez dos modelos tradicionais.

É importante também considerar o papel da mediação docente nesse processo. Como destaca Fiorentini (2009), as tecnologias não ensinam sozinhas, mas ampliam as possibilidades do ensinar e do aprender quando integradas criticamente ao projeto pedagógico da escola. Assim, a formação inicial e continuada dos professores torna-se peça-chave para garantir o uso qualificado dessas ferramentas, sobretudo nos cursos de Licenciatura em Matemática, que ainda carecem de uma inserção mais sistemática das tecnologias digitais em seus currículos.

A inserção de linguagens de programação no ensino de Matemática não deve ser vista como uma substituição dos conteúdos tradicionais, mas como um complemento enriquecedor. A linguagem *Python*, por exemplo, permite que os alunos programem expressões algébricas, visualizem gráficos de funções em tempo real e simulem problemas matemáticos aplicados. Essas experiências, além de desenvolverem competências lógico-matemáticas, promovem a interdisciplinaridade, a criatividade e o protagonismo discente.

Nesse cenário, o uso de tecnologias deixa de ser um diferencial e passa a ser uma necessidade, especialmente diante das diretrizes da BNCC (Brasil, 2018), que apontam para a importância do pensamento computacional, da cultura digital e da resolução colaborativa de problemas como competências essenciais a serem desenvolvidas no Ensino Básico.

Apesar do potencial transformador das tecnologias digitais no ensino da Matemática, sua inclusão ainda enfrenta obstáculos significativos, sobretudo nas redes públicas de ensino. A desigualdade no acesso a equipamentos e à internet de qualidade representa uma barreira concreta à democratização do uso de ferramentas computacionais em sala de aula. Além disso, muitos professores não tiveram, durante sua formação inicial, a oportunidade de explorar pedagogicamente os recursos tecnológicos, o que gera insegurança e resistência frente às inovações. Como discute Kenski (2012), a presença das tecnologias na escola não garante, por si só, a transformação das práticas pedagógicas; é preciso formação, condições estruturais e apoio institucional, assim o docente deve trabalhar em conjunto com as instituições de ensino.

A ausência desses elementos dificulta a efetivação de propostas mais integradoras e inovadoras, como o uso da linguagem *Python* no ensino de conceitos matemáticos.

Assim, é necessário pensar em políticas educacionais que promovam tanto o acesso quanto a capacitação docente continuada, para que as tecnologias realmente cumpram seu papel de ampliar horizontes e qualificar a aprendizagem.

2.3. O PAPEL DAS INSTITUIÇÕES DE ENSINO DIANTE O USO DAS TECNOLOGIAS EM SALA DE AULA

As instituições de ensino, enquanto espaço social de formação crítica e desenvolvimento humano, têm um papel fundamental na mediação entre os avanços tecnológicos e os processos de ensino-aprendizagem. Com a intensificação do uso de recursos digitais no cotidiano e no mundo do trabalho, especialmente após a pandemia da Covid-19, tornou-se urgente que o ambiente escolar ou universitário acompanhe essas transformações e prepare os sujeitos para atuarem de forma crítica e produtiva na sociedade contemporânea.

Entretanto, é importante destacar que a simples presença das tecnologias em sala de aula não é garantia de inovação pedagógica. Como afirma Kenski (2012), as tecnologias não transformam a educação por si sós; é o modo como são apropriadas pelos sujeitos que determina seu impacto pedagógico. Nesse sentido, cabe à escola não apenas inserir equipamentos, mas criar uma cultura pedagógica voltada ao uso significativo e contextualizado das tecnologias, oferecendo aos professores e alunos as condições materiais e formativas necessárias para esse processo.

Além da escola básica, as universidades e instituições de Ensino Superior desempenham um papel fundamental na consolidação da cultura digital na educação, são esses espaços que formam os futuros professores e, portanto, têm a responsabilidade de preparar profissionais capazes de atuar com competência nas demandas do século XXI.

No entanto, ainda se observa uma lacuna significativa no que diz respeito à formação tecnológica de docentes, especialmente nos cursos de licenciatura, que muitas vezes priorizam conteúdos teóricos em detrimento da prática pedagógica com recursos digitais. Segundo Gil (2023), uma formação profissional completa deve considerar as transformações sociais e tecnológicas como elementos estruturantes dos processos educativos, e não apenas como complementos opcionais.

A Base Nacional Comum Curricular (BNCC) também reforça esse papel institucional ao definir a competência geral nº 5, que aponta para a necessidade de compreender, utilizar e criar tecnologias digitais de forma crítica, significativa e ética.

Tal diretriz evidencia que a escola não pode se eximir da responsabilidade de preparar os estudantes para atuarem nesse novo cenário. No campo da Matemática, por exemplo, o uso de linguagens de programação como o Python pode representar uma ponte entre o conteúdo tradicional e as habilidades do século XXI, favorecendo o raciocínio lógico, a resolução de problemas e a interdisciplinaridade.

Contudo, para que isso aconteça, é necessário superar o modelo escolar centrado na memorização e na transmissão de conteúdo. Papert (1994), ao propor a ideia do *Construcionismo*, destaca que as tecnologias devem ser usadas para criar ambientes em que os alunos aprendam ativamente, construindo conhecimento por meio da experimentação e da descoberta. A escola, nesse contexto, precisa se transformar em um espaço de pesquisa, criação e interação, em que o erro não seja punido, mas compreendido como parte do processo de aprendizagem.

Além disso, o papel da gestão educacional é central nesse processo. É ela quem pode criar políticas internas de formação docente contínua, aquisição de recursos, reformulação do projeto pedagógico e parcerias com universidades e centros tecnológicos. Como ressalta Gil (2023), a qualidade da ação educacional está diretamente relacionada à capacidade institucional de planejamento e inovação. Portanto, para que o uso das tecnologias seja efetivo e não apenas decorativo, é necessário o envolvimento de toda a comunidade escolar: professores, alunos, gestores e famílias.

Assim, é importante lembrar que o ambiente educacional é também um espaço de resistência. Diante de tantas desigualdades sociais e digitais, cabe à instituição lutar para garantir o acesso equitativo às tecnologias, combatendo a exclusão digital e promovendo oportunidades de aprendizagem significativas para todos. Nesse aspecto, iniciativas que busquem integrar linguagens de programação ao ensino da Matemática, como a proposta deste trabalho, podem contribuir não apenas para o avanço pedagógico, mas também para a construção de uma escola mais justa, atualizada e conectada com o futuro.

3. A LINGUAGEM DE PROGRAMAÇÃO

3.1. PROGRAMAÇÃO E PENSAMENTO COMPUTACIONAL NO ENSINO DA MATEMÁTICA

A consolidação de uma sociedade cada vez mais mediada por dados, algoritmos e dispositivos computacionais, o desenvolvimento do pensamento computacional tornou-se uma competência fundamental para o século XXI. Este tipo de raciocínio, que envolve a decomposição de problemas, o reconhecimento de padrões, a abstração e a criação de algoritmos, é especialmente convergente com a lógica matemática, tornando-se uma aliada poderosa no processo de ensino e aprendizagem dessa área do conhecimento.

No contexto educacional, o pensamento computacional vem sendo associado ao ensino de programação como forma de desenvolver habilidades cognitivas complexas, como a resolução de problemas, a modelagem de situações reais e o raciocínio lógico. De acordo com Papert (1980), precursor dessa abordagem, a programação pode ser uma ferramenta para transformar a maneira como os alunos aprendem, pois, aprender a programar é uma forma de aprender a pensar. Nessa perspectiva, a linguagem de programação não é apenas uma técnica, mas um meio de expressão intelectual que permite ao aluno construir conhecimento de maneira ativa e significativa.

A integração da programação ao ensino da Matemática, especificamente, oferece oportunidades para dinamizar conteúdos tradicionalmente abstratos, como expressões algébricas, funções e geometria analítica. Ao construir códigos que resolvem problemas matemáticos, os alunos passam a visualizar os conceitos de forma prática e funcional, tornando a aprendizagem mais concreta. Segundo Wing (2006), o pensamento computacional é uma forma de pensamento universal que deve ser ensinada a todos, pois ultrapassa as barreiras disciplinares e estimula o pensamento crítico e criativo.

A Base Nacional Comum Curricular (BNCC) também reforça essa necessidade ao destacar, entre suas competências gerais, o uso consciente e crítico das tecnologias digitais e o incentivo à cultura digital como parte essencial da formação dos estudantes brasileiros. Essa diretriz abre espaço para que disciplinas como a Matemática estabeleçam vínculos mais profundos com o mundo contemporâneo por meio de práticas como a programação.

É importante destacar que o uso da programação em sala de aula não se resume à simples introdução de *softwares* ou linguagens, trata-se de uma mudança de postura

pedagógica que exige planejamento, formação docente e intencionalidade didática. Como aponta Gil (2017), o processo educativo, para ser eficiente, precisa considerar o contexto social e tecnológico do aluno, suas formas de interagir com o mundo e os instrumentos que ele utiliza para construir conhecimento. Desse modo, o uso de linguagens como *Python* no ensino da Matemática deve ser orientado por propostas que articulem teoria, prática e tecnologia, valorizando a autonomia e a experimentação dos estudantes.

Nesse sentido, é pertinente destacar uma citação de Papert (1985, p. 43), que resume a importância da programação na educação:

> Meu argumento é que a criança que aprende a programar comete erros que não são mais vistos como fracassos, mas como partes naturais do processo de aprendizagem. A programação permite que ela construa hipóteses, as teste e as modifique com base em resultados concretos. A Matemática, nesse contexto, deixa de ser uma disciplina passiva e abstrata, passando a ser explorada como um sistema vivo, interativo e manipulável.

Essa concepção representa uma ruptura com os métodos tradicionais de ensino, centrados na memorização e na repetição de procedimentos. O pensamento computacional, aliado ao uso de linguagens acessíveis como *Python*, contribui para a formação de alunos mais autônomos, críticos e preparados para lidar com os desafios contemporâneos, reforçando o papel da Matemática como ferramenta de empoderamento intelectual e social.

3.2. RACIOCÍNIO LÓGICO E O COMPUTADOR

O raciocínio lógico é um dos pilares fundamentais do pensamento matemático. Ele permite ao indivíduo formular hipóteses, organizar argumentos, estabelecer relações e resolver problemas com base em inferências coerentes. No contexto do ensino da Matemática, estimular o raciocínio lógico é promover o desenvolvimento da autonomia intelectual, da criatividade e da argumentação, competências essas indispensáveis na formação de cidadãos críticos.

Com o avanço das tecnologias digitais e a popularização do acesso aos computadores, o raciocínio lógico passou a ser cada vez mais associado ao universo computacional. Isso ocorre porque, para que um computador funcione corretamente, ele depende de comandos lógicos, estruturas condicionais, algoritmos e sequências bem definidas de instruções. Como ressalta Papert (1985), ao programar, o aluno não apenas utiliza a lógica matemática, mas também aprende sobre ela: a programação pode se tornar

um ambiente poderoso para a construção do conhecimento, pois obriga o estudante a pensar sobre o próprio pensar.

Essa conexão entre lógica e computação fica ainda mais clara quando se considera o uso de linguagens de programação no ensino. Linguagens como Python exigem que o aluno compreenda estruturas de decisão, repetição e organização sequencial — exatamente os mesmos princípios que regem a lógica matemática. Dessa forma, ao resolver um problema utilizando um código, o estudante está, ao mesmo tempo, exercitando a Matemática de maneira prática e contextualizada.

Um exemplo simples e eficaz é a criação de algoritmos com comandos básicos em *Python* para resolver expressões matemáticas, como calcular o valor de uma equação do primeiro grau. Esse tipo de atividade exige que o aluno compreenda a lógica do problema antes de traduzi-lo em código, fortalecendo sua compreensão e raciocínio. A aplicação prática da lógica computacional pode ser observada também em atividades como a construção de programas que identificam se um número é par ou ímpar, simulando uma calculadora simples ou resolvendo problemas geométricos por meio da entrada de variáveis. Em atividades como essas, os estudantes precisam compreender conceitos matemáticos, identificar padrões e estruturar passos lógicos para que o computador possa executar corretamente — o que representa, na prática, a materialização do pensamento lógico por meio de uma linguagem formal

Em um paralelo criativo, pode-se evocar a canção "Tempos Modernos", de Lulu Santos, que diz: “vamos viver tudo que há pra viver, vamos nos permitir”. Essa mensagem, embora poética, ressoa com a proposta pedagógica de ampliar os horizontes do ensino tradicional, permitindo que a tecnologia e a lógica computacional entrem na sala de aula como ferramentas libertadoras do pensamento matemático engessado. A escola que acolhe o computador como instrumento de raciocínio lógico oferece ao aluno a possibilidade de construir seu próprio caminho de aprendizado, de forma investigativa e personalizada, permitindo que o aluno utilize a sua criatividade e sua forma de raciocínio na resolução dos problemas propostos.

No entanto, não se trata apenas de “usar o computador”, mas de utilizá-lo como uma extensão da mente — como defende Moran (2021), ao discutir a reinvenção das práticas pedagógicas na era digital. Segundo o autor, as tecnologias digitais potencializam a capacidade humana de pensar, criar e interagir com o conhecimento, sendo a programação uma dessas ferramentas essenciais. Ao incorporar essas tecnologias no

ensino, o processo educativo se torna mais dinâmico e significativo, favorecendo o desenvolvimento do pensamento crítico e a resolução de problemas complexos.

Desse modo, o computador deixa de ser apenas uma máquina executora de tarefas e passa a ser um parceiro pedagógico, que dialoga diretamente com o raciocínio lógico e a construção do conhecimento matemático. Integrar essa perspectiva ao ensino é, portanto, mais do que uma inovação: é uma resposta necessária aos desafios de uma educação matemática mais crítica, interativa e significativa.

3.3. ALGORITMO

A palavra "algoritmo" remete, etimologicamente, ao matemático persa Al-Khwarizmi, cujas obras influenciaram profundamente o desenvolvimento da álgebra e da aritmética no mundo ocidental. No contexto atual, especialmente na interseção entre a Matemática e a Computação, o termo representa uma sequência finita e bem definida de instruções que conduz à resolução de um problema ou à realização de uma tarefa.

A compreensão de algoritmos não se restringe ao domínio da tecnologia: ela é fundamental para o pensamento matemático, pois exige organização lógica, clareza na formulação de etapas e análise de variáveis. Segundo Cormen et al. (2012, p. 23), "um algoritmo é qualquer procedimento computacional bem definido que toma algum valor ou conjunto de valores como entrada e produz algum valor ou conjunto de valores como saída". Essa definição reforça o caráter prático e funcional do conceito, estabelecendo uma ponte entre o raciocínio matemático e a aplicabilidade computacional.

No âmbito educacional, a abordagem algorítmica permite que os estudantes desenvolvam habilidades como abstração, decomposição de problemas e planejamento estratégico. Papert (1985) argumenta que, quando o aluno constrói algoritmos, ele não apenas aprende conteúdos, mas aprende sobre como aprende, criando uma "metacognição" do próprio processo de aprendizagem, que significa pensar sobre o próprio pensamento. Esse aspecto dialoga com a proposta construcionista, na qual o conhecimento é gerado por meio da experiência ativa e significativa do sujeito.

Além disso, o ensino de algoritmos em ambientes escolares contribui para o desenvolvimento do chamado pensamento computacional, uma habilidade descrita por Wing (2006) como uma abordagem para resolver problemas, projetar sistemas e compreender comportamentos humanos, baseada nos conceitos fundamentais da ciência da computação. Ao elaborar um algoritmo, o estudante exercita a lógica, a precisão, a

capacidade de prever consequências e, sobretudo, o controle sobre os processos — características essenciais tanto no campo da Matemática quanto na formação para a cidadania digital.

Na prática, o ensino de algoritmos pode ser introduzido por meio de fluxogramas, pseudocódigos ou linguagens de programação acessíveis, como o *Python*. Por exemplo, pedir que os alunos elaborem um algoritmo para calcular a média aritmética de três números permite que eles mobilizem habilidades matemáticas básicas enquanto estruturam etapas lógicas com clareza. Abaixo, um exemplo simples de algoritmo representado em pseudocódigo:

Quadro 1 – Algoritmo em pseudocódigo

Início
Ler número 1
Ler número 2
Ler número 3

Média ← (número1 + número2 + número3) / 3

Exibir média
Fim

Fonte: Autoria própria.

Outro exemplo visual pode ser feito com fluxograma, uma representação gráfica muito utilizada no Ensino Fundamental e Médio. O fluxograma para esse mesmo algoritmo seria composto por:

- Um bloco de **início**;
- Três blocos de **entrada de dados** (número1, número2 e número3);
- Um bloco de **processamento** (cálculo da média);
- Um bloco de **saída** (mostrar a média);
- Um bloco de **fim**.

Esses recursos visuais ajudam o estudante a compreender de forma mais clara o funcionamento de um algoritmo e podem ser criados com ferramentas gratuitas como *draw.io* ou *Lucidchart*.

Outro exercício clássico envolve algoritmos de ordenação de dados ou a verificação de números primos, que também ajudam a fixar conceitos fundamentais da

Matemática, como a divisibilidade e sequências numéricas. Esse exercício poderia facilmente ser construído e executado em uma linguagem de programação como o *Python*. Vale ressaltar que a aprendizagem significativa ocorre quando o aluno é capaz de relacionar o novo conhecimento com estruturas cognitivas previamente adquiridas. Nesse sentido, o ensino de algoritmos torna-se um instrumento não apenas de conteúdo, mas de articulação entre saberes, incentivando conexões entre diferentes áreas do conhecimento.

Dessa forma, consolidar o conceito de algoritmo desde as etapas iniciais da formação em Matemática é uma estratégia essencial para preparar os futuros docentes para uma prática pedagógica crítica, criativa e integrada ao mundo digital. A proposta vai ao encontro dos objetivos da BNCC (Base Nacional Comum Curricular), que reconhece o pensamento computacional como uma competência a ser desenvolvida ao longo da Educação Básica, promovendo a formação de sujeitos autônomos e preparados para os desafios contemporâneos.

4. A LINGUAGEM CIENTÍFICA DE PROGRAMAÇÃO – PYTHON

4.1. O QUE É UMA LINGUAGEM CIENTÍFICA DE PROGRAMAÇÃO

A revolução digital e os avanços nas Tecnologias Digitais da Informação e Comunicação (TDICs) têm provocado mudanças profundas na educação, especialmente no ensino de Matemática. Nesse novo cenário, o uso de linguagens de programação como recurso didático surge como uma alternativa potente, capaz de tornar o processo de aprendizagem mais dinâmico e significativo. Mais do que uma ferramenta técnica, programar se torna uma forma de pensar e de explorar conceitos matemáticos de maneira ativa e criativa. Com isso, desenvolvem-se habilidades como o pensamento computacional, o raciocínio lógico e a capacidade de resolver problemas de forma autônoma.

Dentro desse universo, as chamadas linguagens científicas de programação ocupam um lugar de destaque. Elas são projetadas para lidar com problemas matemáticos e científicos, permitindo desde a criação de fenômenos até a análise de dados complexos. Entre elas, a linguagem *Python* tem ganhado cada vez mais espaço, tanto por sua estrutura simples e intuitiva quanto pela riqueza de suas bibliotecas voltadas à Matemática. Trata-se de uma linguagem acessível, que pode ser utilizada por iniciantes, mas que também atende às necessidades de profissionais em diferentes áreas do conhecimento.

Para o educador e teórico Seymour Papert (1980), aprender é, acima de tudo, construir. Ele afirma que a aprendizagem se torna mais significativa quando o aluno está engajado em criar algo que tenha sentido para ele. No caso da programação, esse "algo" é construído a partir da manipulação de ideias matemáticas, o que torna os conceitos mais palpáveis e compreensíveis. Assim, o ato de programar transforma-se em uma linguagem para pensar: uma forma de organizar ideias, testar hipóteses e refletir sobre o próprio conhecimento.

A partir desse pressuposto, uma linguagem científica de programação pode ser compreendida como uma ferramenta computacional voltada à representação, exploração e resolução de problemas nas ciências exatas, com forte presença na Matemática. Ela permite que ideias abstratas sejam traduzidas em ações concretas, simuladas e testadas por meio da codificação. Sob o olhar de Papert (1980), tais linguagens não apenas viabilizam soluções, mas ampliam a capacidade de pensar sobre o problema, tornando-se instrumentos intelectuais que favorecem o aprender fazendo. O *Python*, por exemplo, ao

possibilitar que estudantes construam algoritmos, visualizem gráficos e testem conjecturas, contribui para a formação de uma postura ativa e investigativa diante do conhecimento matemático.

Além de promover o engajamento ativo dos estudantes, o uso de linguagens científicas de programação, como o Python, está alinhado às diretrizes da Base Nacional Comum Curricular (BNCC), que reconhece a importância da integração das Tecnologias Digitais da Informação e Comunicação (TDICs) ao processo educativo. Segundo o documento, é papel da escola "proporcionar aos estudantes o desenvolvimento do pensamento computacional, a compreensão crítica das tecnologias digitais e a capacidade de usá-las de maneira ética e criativa" (Brasil, 2018, p. 15). Nesse sentido, a programação deixa de ser um conhecimento restrito a cursos técnicos ou à área de computação, assumindo um papel formativo transversal, com grande potencial para o ensino de Matemática.

A Matemática, enquanto componente curricular, tem entre suas competências gerais o desenvolvimento da capacidade de modelar e resolver problemas, interpretar dados e construir argumentos logicamente consistentes. A utilização de uma linguagem como o Python potencializa essas competências ao permitir a visualização de padrões, a construção de modelos matemáticos interativos e a validação de conjecturas por meio da simulação computacional. Tal abordagem atende não apenas aos objetivos específicos da disciplina, mas também às competências gerais da educação básica, como o estímulo ao pensamento científico, crítico e criativo, conforme previsto no Art. 35-A da LDB (Lei nº 9.394/96, alterada pela Lei nº 13.415/2017).

Outro ponto relevante é o caráter interdisciplinar que a linguagem *Python* pode assumir no ambiente escolar. Ao articular conhecimentos de Matemática com elementos da computação, ela favorece uma abordagem pedagógica integrada, que rompe com a fragmentação dos saberes e amplia o sentido da aprendizagem, além do discente poder associar as aprendizagens a qualquer outro componente da instituição de ensino ou situação do cotidiano. Esse aspecto está em consonância com as exigências da formação docente contemporânea, que requer do professor não apenas domínio técnico, mas também sensibilidade para criar experiências de aprendizagem contextualizadas e significativas.

Dessa forma, compreender o que é uma linguagem científica de programação não se limita à definição de seus aspectos técnicos, mas envolve reconhecer seu potencial educativo, sua capacidade de mediar a construção de conhecimento e sua relevância para

uma educação matemática crítica, atualizada e conectada aos desafios do século XXI. Ao incorporar ferramentas como o *Python* ao ensino, o professor contribui para a formação de sujeitos mais autônomos, reflexivos e preparados para lidar com a complexidade do mundo digital.

4.2. *SOFTWARES*, DIFERENÇAS BÁSICAS E PLATAFORMAS DE ENSINO

A escolha do ambiente de desenvolvimento é uma etapa essencial quando se propõe integrar o ensino da Matemática à linguagem de programação. Considerando que muitos professores e estudantes estarão tendo o primeiro contato com ferramentas computacionais, é fundamental apresentar opções claras, acessíveis e funcionais, que se adequem tanto aos objetivos pedagógicos quanto às condições da escola e da turma. Neste contexto, duas plataformas se destacam: o *Google Colab* e o *Visual Studio Code* (*VSCode*). Chamadas de *IDE* (do inglês *Integrated Development Environment*, ou Ambiente de Desenvolvimento Integrado) é um *software* que reúne diversas ferramentas usadas na programação em um único lugar, facilitando o processo de escrever, testar, depurar (debugar) e executar códigos.

O Google Colab, ou *Google Colaboratory*, é uma plataforma gratuita oferecida pela *Google* que permite a escrita e execução de códigos *Python* diretamente do navegador. Seu uso é semelhante ao do *Google Docs*: o usuário acessa um arquivo *online*, escreve em células interativas, insere explicações teóricas, executa comandos e visualiza os resultados em tempo real, sem necessidade de instalar qualquer programa no computador. Isso faz do *Colab* uma excelente alternativa para ambientes escolares com pouca infraestrutura ou que trabalham com equipamentos compartilhados. Além disso, por ser baseado em notebooks *Jupyter*, ele permite que teoria e prática sejam organizadas lado a lado, favorecendo metodologias ativas e colaborativas.

Por outro lado, o *VSCode* que requer instalação local, mas oferece uma estrutura mais robusta e personalizável. *IDE's*, de modo geral, funcionam como cadernos digitais completos, em que se escreve o código, organiza os arquivos, executa os comandos e acompanha os resultados, tudo em um só lugar. O *VSCode*, desenvolvido pela Microsoft, é gratuito, leve e possui uma vasta gama de extensões que ampliam suas funcionalidades, como a execução de *notebooks.ipynb*, integração com bibliotecas matemáticas e até controle de versão com *Git*. Ele é amplamente utilizado no mercado profissional e,

quando bem explorado, proporciona uma experiência mais próxima do desenvolvimento real de projetos computacionais.

Na prática, ambas as plataformas permitem trabalhar com a linguagem *Python* de maneira eficiente, e a escolha entre uma ou outra deve levar em conta a realidade do professor, o nível de complexidade desejado e o grau de autonomia dos alunos. Para tornar essa escolha mais objetiva, segue abaixo uma tabela comparativa com os principais pontos diferenciais entre as duas *IDE'S*:

Tabela 1 – *Google Colab* ***vs*** *Vscode*

CARACTERÍSTICA	GOOGLE COLAB	VSCODE
Instalação necessária	Não	Sim
Acesso	Navegador (qualquer SO)	Depende do sistema operacional
Recomendado para iniciantes	Sim	Exige orientação inicial
Uso offline	Não	Sim
Facilidade de compartilhamento	Muito fácil (estilo Google Docs)	Mais técnico (via Git ou exportações)
Controle do ambiente	Limitado	Avançado (extensões, terminal, etc.)
Organização de projetos	Linear (por arquivo)	Modular (várias pastas e arquivos)
Projetos grandes/profissionais	Não	Muito indicado
Interface com notebooks Jupyter	Navegador (qualquer SO)	Com extensão

Fonte: Autoria própria.

Ao observar a tabela, nota-se que o *Google Colab* atende bem aos primeiros contatos com a linguagem Python, sobretudo em contextos educacionais de acesso limitado ou com foco em aulas introdutórias. Sua interface simplificada e funcionamento direto do navegador permitem que os alunos se concentrem mais na lógica do código do que nas configurações técnicas. Já o *VSCode* é ideal para propostas que busquem aprofundar o conhecimento na linguagem, desenvolver projetos mais complexos e fomentar maior controle sobre os processos computacionais, ainda que requeira maior familiaridade com o ambiente.

A escolha da plataforma de ensino deve ser feita de maneira estratégica, considerando os objetivos de aprendizagem e as condições de implementação. Neste trabalho, optamos pelo uso do *VSCode* na construção da sequência didática, apresentada no Capítulo 6, por entendermos que, ao trabalhar com expressões algébricas e funções de

primeiro e segundo grau em um ambiente mais técnico, conseguimos explorar de maneira mais completa as potencialidades da linguagem *Python*. Ainda assim, é importante destacar que, caso o leitor opte por um ambiente mais simples e direto, como o *Google Colab*, será possível aplicar a sequência didática com os mesmos recursos computacionais e resultados pedagógicos equivalentes.

4.3. INTRODUÇÃO A LINGUAGEM PYTHON E A CONSTRUÇÃO DE PROJETOS MATEMÁTICOS

Nos últimos anos, as linguagens de programação deixaram de ser ferramentas restritas ao universo da computação e passaram a integrar, com crescente naturalidade, o campo da educação. Entre essas linguagens, o *Python* tem se destacado não apenas por sua aplicabilidade técnica, mas também por sua potência pedagógica. Sua sintaxe clara, estrutura simples e versatilidade permitem que até mesmo pessoas sem experiência prévia consigam se familiarizar com seus comandos de forma intuitiva. É justamente essa característica que torna o *Python* um aliado promissor na construção de projetos voltados para o ensino da Matemática.

A linguagem de programação *Python* surgiu no final dos anos 1980, desenvolvida pelo programador holandês Guido Van Rossum, enquanto trabalhava no *Centrum Wiskunde & Informatica (CWI)*, na Holanda. O projeto foi iniciado como uma alternativa ao ABC, uma linguagem educacional também criada no *CWI*, mas que apresentava algumas limitações técnicas. O *Python* foi pensado como uma linguagem simples, com código legível e de fácil manutenção, que pudesse ser utilizada tanto para fins educacionais quanto para aplicações profissionais. A primeira versão oficial foi lançada em 1991, já com diversos recursos que permanecem até hoje, como estruturas de dados poderosas e suporte à orientação a objetos.

O nome "*Python*" não tem relação direta com a serpente, como muitos pensam. Segundo o próprio Van Rossum, a inspiração veio do grupo de comédia britânico *Monty Python's Flying Circus*, do qual ele era fã. Ele buscava um nome curto, impactante e ligeiramente misterioso, que refletisse o espírito leve e acessível da linguagem. Essa escolha traduz bem o princípio que orienta a linguagem até hoje: tornar a programação mais divertida, criativa e menos intimidadora para iniciantes. Por isso, *Python* é frequentemente a linguagem escolhida em cursos introdutórios de ciência da computação e em projetos educacionais.

Do ponto de vista técnico, o *Python* é uma linguagem de alto nível, interpretada e multiparadigma, o que significa que ela permite diferentes estilos de programação — como o imperativo, o orientado a objetos e, em certa medida, o funcional. Seu design prioriza a legibilidade do código e a simplicidade da sintaxe, reduzindo a complexidade que costuma afastar estudantes iniciantes. Além disso, a linguagem possui uma vasta biblioteca padrão e uma comunidade ativa, o que facilita sua aplicação em áreas diversas como ciência de dados, inteligência artificial, automação, web, e, mais recentemente, na educação matemática. Como destaca Van Rossum (1995), a linguagem deve ser acessível o suficiente para que o foco esteja na lógica do problema, e não na complexidade da linguagem em si.

Neste capítulo, propomos uma introdução leve e estruturada à linguagem *Python*, especialmente voltada a docentes e estudantes do Ensino Médio e Ensino Superior que desejam explorar novas formas de tornar a Matemática mais significativa, criativa e conectada com a realidade digital dos alunos. Visto que, a linguagem se encaixa como uma opção acessível e compatível ao ensino do componente. A escolha da linguagem, se deu pela sua estrutura e fácil compreensão tanto para aqueles que já se familiarizam com números e linguagens quanto aos que sentem dificuldades.

Para compreender a potência educativa da linguagem *Python*, é essencial que a introdução ao código não seja feita de maneira mecânica ou puramente técnica. De acordo com David Ausubel (2003, p. 83), "a aprendizagem significativa ocorre quando uma nova informação se conecta de maneira substantiva e não arbitrária à estrutura cognitiva do aluno". Assim, o aprendizado de programação precisa dialogar com os conhecimentos prévios do estudante, preferencialmente aqueles adquiridos no próprio contexto escolar, como operações algébricas, gráficos de funções, proporcionalidade e raciocínio lógico.

David Ausubel foi um psicólogo educacional norte-americano que desenvolveu a teoria da aprendizagem significativa, uma das abordagens mais relevantes para o ensino e a aprendizagem nas últimas décadas. Para Ausubel, o fator mais importante que influencia a aprendizagem é aquilo que o aluno já sabe — ou seja, seus conhecimentos prévios. Segundo ele, a aprendizagem significativa ocorre quando o novo conteúdo é relacionado de maneira não arbitrária e substantiva ao que o estudante já conhece, como citado anteriormente, promovendo uma compreensão mais profunda e duradoura do saber.

Diferentemente da aprendizagem mecânica, que se baseia na repetição e memorização sem conexão com o conhecimento anterior, a aprendizagem significativa

valoriza a integração de novas ideias a estruturas cognitivas já existentes. Isso requer que o conteúdo seja potencialmente significativo (isto é, logicamente organizado) e que o aluno esteja disposto a aprender de forma consciente e ativa. A proposta de Ausubel reforça a importância de estratégias pedagógicas que contextualizem o conhecimento, tornando-o relevante para a realidade do estudante — como é o caso do uso de linguagens computacionais no ensino da Matemática, por exemplo.

Complementando essa ideia, Novak (2010), em seus estudos sobre mapas conceituais e construção de sentido, afirma que, aprender não é apenas memorizar fatos, fórmulas ou algoritmos; é organizar o conhecimento em estruturas mentais coerentes, integradas e acessíveis para novos usos. Quanto mais significativa for essa organização, mais provável será a transferência do conhecimento para situações reais.

Nesse sentido, aprender *Python* não deve ser um fim em si mesmo, mas um meio para a construção de projetos matemáticos com propósito, que envolvam criatividade, colaboração e solução de problemas reais, uma vez que na realidade escolar os discentes queixam que a Matemática "nunca será utilizada em contextos fora da escola", o que resulta em uma visão errônea do componente, pois a mesma está em tudo ao nosso redor, principalmente nas tecnologias.

A proposta de introduzir o *Python* no ensino de Matemática se apoia fortemente no pensamento de Seymour Papert, criador do construcionismo e defensor da ideia de que o computador pode ser uma poderosa ferramenta de aprendizagem quando utilizado para construir algo significativo para o próprio aprendiz. Papert (1980, p. 135) descreve:

> O *construcionismo* sustenta que o aprendizado é mais eficaz quando as pessoas estão ativamente envolvidas na construção de um produto externo — algo que seja compartilhável e passível de reflexão. Esse produto pode ser um castelo de areia, um robô ou um programa de computador. O importante é que ele sirva como objeto de pensamento.

Assim, ensinar *Python* por meio da criação de projetos matemáticos concretos — como uma calculadora algébrica, um gerador de gráficos, ou um sistema de resolução de equações — permite que os alunos vejam sentido no que estão fazendo, sintam-se desafiados e desenvolvam habilidades cognitivas e socioemocionais de forma integrada.

Ao introduzir uma linguagem de programação no contexto educacional, especialmente para alunos iniciantes, é essencial que o processo se inicie com os fundamentos mais simples e próximos da linguagem humana, pois pode parecer, à primeira vista, um salto complexo. A linguagem, nesse sentido, destaca-se justamente por

oferecer uma sintaxe acessível e próxima do português estruturado, o que favorece a aprendizagem significativa — conforme propõe Ausubel et al. (1980) — ao relacionar novos conteúdos com conhecimentos previamente existentes.

A seguir, apresentaremos os principais tópicos introdutórios que devem compor a base de um primeiro contato com *Python*. Os principais pontos para compreensão da linguagem serão separados por tópicos: i. Definição e instalação das ferramentas – *Python* e *VSCode*; ii. Sintaxe básica e estrutura de um programa; iii. Tipos de dados; iv. Operadores em python; v. Estruturas de controle; vi. Funções; vii. Módulos e bibliotecas; viii. Estruturas de dados compostas; iv. Entrada e saída de dados; x. Tratamento de erros (introdução).

Observação: Todos os códigos utilizados como exemplos foram criados dentro do próprio programa, com autoria própria, para melhor compreensão, é possível recriar os comandos dentro da *IDE Vscode* ou *Google Colab*, de acordo com a preferência. Neste trabalho, como mencionado anteriormente, optamos pelo uso do *VScode*.

Cada termo dentro do código está classificado em uma cor específica, vale lembrar que as cores podem variar dependendo do editor e do tema usado. O importante é que o sistema de cores sempre siga a mesma lógica: destacar a função de cada elemento do código para facilitar a leitura. Deixaremos uma legenda para melhor compreensão da escolha das cores na construção dos códigos que virão posteriormente e que tipo de dado cada uma delas representa.

Tabela 2 – Legenda de códigos

COR – VISUAL	TIPO DE ELEMENTO	EXEMPLO	EXPLICAÇÃO
Azul 	Palavras-chave (*Keywords*)	*for, if, else, while, def, return, import, in*	Comandos da linguagem usados para estruturação de controle e definição de funções.
Ciano 	*Strings* (textos)	"Olá", 'Texto'	Sequências de caracteres delimitadas por aspas simples ou duplas.
Preto 	Nomes de variáveis, bibliotecas e funções definidas	nota, resposta, soma números(), *math*	Identificadores criados pelo programador.
Verde escuro 	Comentários	# isso é um comentário	Texto que não é executado; serve para explicação no código.
Rosa 	Números e literais	5, 3.14, -10	valores numéricos constantes.
Laranja 	Delimitadores e operadores	=, +, ==, :, (,)	Usados para expressar operações, atribuições ou agrupar instruções.
Marrom 	Funções embutidas (*built-in*)	*print(), input(), range(), round()*	Funções já existentes na linguagem *Python*.
Vermelho 	Tipo de dados	*int, float, str, bool*	O tipo de dado que será utilizado

Fonte: Autoria própria.

i. Definição e instalação das ferramentas – *Python* e *VScode*

Python é uma linguagem de programação de alto nível, o que significa que seu código é mais próximo da linguagem humana do que das instruções que o computador realmente executa. Criada por Guido van Rossum em 1991, ela foi pensada para ser simples, clara e eficiente. Diferente de outras linguagens que exigem comandos complexos já no início, *Python* permite que até pessoas com pouca familiaridade com

computadores consigam escrever instruções básicas de forma compreensível. Veja o exemplo:

Quadro 2 – Primeiro programa

```
print ("Olá, mundo!")
```

Fonte: Autoria própria.

Esse é o clássico primeiro programa. Ao rodar esse código, o computador exibirá a frase "Olá, mundo!" na tela. Isso já nos mostra a simplicidade de começar a programar com *Python*. A tradução de *print* é "imprimir", ou seja, o desenvolvedor pede que o programa imprima essa frase na tela.

Instalação das ferramentas:

- *Python*: https://www.python.org/downloads/
- IDE: *VScode*: https://code.visualstudio.com/

Caso haja qualquer dúvida sobre a instalação das ferramentas, como uma alternativa prática, utilize a plataforma *YouTube*, lá existem diversos vídeos curtos e explicativos que ensinam passo a passo.

ii. Sintaxe básica e estrutura de um programa

A sintaxe de uma linguagem é como sua gramática: ela define as regras de escrita do código. Em *Python*, essas regras são muito simples. O que mais se destaca é a indentação, ou seja, o recuo que se faz com o botão de espaço ou "*tab*". Esse recuo não é apenas estético: ele indica onde começam e terminam os blocos de código. Exemplo:

Quadro 3 – Indentação e comentários

```
# esse exemplo significa uma indentação
if 5 > 3:
    print ("5 é maior que 3")
```

Fonte: Autoria própria.

Nesse exemplo, o *print* está recuado porque pertence ao bloco do *if*. Se o recuo for retirado, o programa dará erro. Diferentemente de outras linguagens, Python não exige ponto e vírgula no final das linhas e utiliza a indentação (espaçamento) como parte da

estrutura do código, o que o torna mais limpo visualmente. Outro exemplo são comentários, o uso do "jogo da velha" seguindo de uma frase.

iii. Tipos de dados

A linguagem consegue entender diferentes tipos de informação, como números, textos e valores lógicos (verdadeiros ou falsos). Para melhor compreensão e visualização dos tipos de dados, há um glossário logo abaixo dividido em colunas, tais como: significado, tradução, tipo (*python*) e observação.

Tabela 3 – Tipos de dados

TIPOS DE DADOS			
Significado	**Tradução – português**	**Tipo – *python***	**Observação**
número inteiro	inteiro	*int*	Não tem limite definido
número de ponto flutuante	real	*float*	-
um único caractere	caractere	*str*	Valores literais devem ter aspas duplas ou aspas simples. Exemplo: "A" ou 'A'
texto	caractere	*str*	Valores literais devem ter aspas duplas ou aspas simples. Exemplo: "João" ou 'João'
valor lógico	lógico	*bool*	Valores possíveis: *True, False*

Fonte: Autoria própria.

- Inteiros (*int*): números sem vírgula – 10, -3, 2025;
- Decimais (*float*): números com vírgula – 3.14, 2.0;
- Textos (*str*): devem estar entre aspas – "Olá";
- Booleanos (*bool*): apenas *True* (verdadeiro) ou *False* (falso).

Veja o exemplo no código abaixo:

Quadro 4 – Variáveis

```
idade = 17                          #inteiro
altura = 1.65                       #decimal
nome = "Ana"                        #texto
maior_idade = True                  #booleano
```

Fonte: Autoria própria.

iv. Operadores em *Python*

Os operadores são os símbolos usados para realizar cálculos ou comparações, o ponto principal para construção de fórmulas e programas matemáticos. Utilizaremos o mesmo método do tipo de dados, a visualização em formato de glossário, divididos em: operadores aritméticos, operadores comparativos e operadores lógicos.

Tabela 4 – Operadores aritméticos

OPERADORES ARITMÉTICOS	
Operador	**Significado**
+	adição
-	subtração
*	multiplicação
/	divisão
%	porcentagem
**	exponenciação
//	divisão inteira

Fonte: Autoria própria.

Tabela 5 – Operadores comparativos

OPERADORES COMPARATIVOS	
Operador	**Significado**
<	menor
>	maior
<=	menor ou igual
>=	maior ou igual
= =	igual
! = ou <>	diferente

Fonte: Autoria própria.

Tabela 6 – Operadores lógicos

OPERADORES LÓGICOS	
Operador	**Significado**
and	e
or	ou
not	não

Fonte: Autoria própria.

Quadro 5 – Operadores

```
a = 10
b = 5
print(a + b)                    # soma
print(a > b)                    # comparação
print(a > b and b > 0)          # lógico
```

Fonte: Autoria própria.

v. Estruturas de controle

As estruturas de controle são elementos fundamentais na programação, pois permitem que o computador tome decisões e repita tarefas de forma automatizada. Em termos educacionais, compreender essas estruturas auxilia no desenvolvimento do raciocínio lógico e da capacidade de resolução de problemas — habilidades amplamente valorizadas nos processos de aprendizagem matemática (GIL, 2008).

Em *Python*, as principais estruturas de controle são as condicionais e as estruturas de repetição. Ambas são simples em sua sintaxe, mas poderosas em suas aplicações. Da mesma forma que aparecem na lógica da programação, surge na lógica matemática.

Tabela 7 – Estruturas de controle

ESTRUTURAS DE CONTROLE			
Condicionais	**Função**	**Repetições**	**Função**
if – se	tomar decisões	*for* - por	executam algo várias vezes
else – senão	tomar decisões	*while* – enquanto	executam algo várias vezes
elif – *else*+*if*	tomar decisões	-	-

Fonte: Autoria própria.

Condicionais: *if*, *elif*, *else*

A estrutura condicional permite que o programa tome decisões com base em uma condição lógica. Em linguagem natural, *if* e *else* equivalem a frases do tipo: "Se (isso acontecer), senão (faça aquilo)". Veja o exemplo:

Quadro 6 – Condicionais *if – else*

```
nota = float(input("Digite  a nota do aluno: "))
if nota >= 6:
    print("Aprovado")
else:
    print("Reprovado")
```

Fonte: Autoria própria.

Neste exemplo, o programa avalia se a variável nota é maior ou igual a 6. Se for, imprime "Aprovado". Caso contrário, imprime "Reprovado". Isso é útil, por exemplo, em avaliações automatizadas ou jogos matemáticos. Observe a indentação sendo aplicada nesse caso.

Podemos também usar o *elif* (abreviação de "*else if*") para criar mais de duas opções:

Quadro 7 – Condicionais *if – else – elif*

```
nota = float(input("Digite  a nota: "))
if nota >= 9:
    print("Excelente")
elif nota >= 6:
    print("Aprovado")
else:
    print("Reprovado")
```

Fonte: Autoria própria.

Essa estrutura permite uma ramificação lógica mais sofisticada, o que enriquece os projetos em sala de aula ao permitir múltiplos resultados conforme a entrada do usuário. Alguns termos como *input*, ainda não foram explicados, apenas exemplificados. Nesse primeiro momento, falaremos de cada termo em sua sequência lógica e posteriormente como escrever os códigos em sua forma estrutural.

Repetições: *for* e *while*

As estruturas de repetição permitem executar um bloco de código várias vezes. Isso é essencial em situações em que se deseja aplicar o mesmo cálculo a diversos dados ou repetir comandos até que uma determinada condição seja satisfeita.

Laço – *for*: O *for* é usado quando sabemos com antecedência o número de repetições. Em Python, ele é geralmente utilizado com a função *range*() traduzido como "faixa", com a função de gerar uma sequência de números inteiros, que normalmente é usada com laços de repetição.

Quadro 8 – Laço

```
for i in range(5):
    print(i)
```

Fonte: Autoria própria.

Esse código imprime os números de 0 a 4. O *range*(5) cria uma sequência de cinco números (0, 1, 2, 3, 4), e o for repete o print(i) para cada número dessa sequência.

vi. Funções

Funções são blocos de código que executam uma tarefa específica. Elas permitem organizar o programa e evitar repetições. O exemplo abaixo exemplifica que ao utilizar o *def* a função saudação() pode ser utilizada quantas vezes for necessário, sem reescrever o código.

Quadro 9 – *Def*

```
def saudacao():
    print("Bem-vindo ao mundo da programação!")
saudacao()
```

Fonte: Autoria própria.

vii. Módulos e bibliotecas

Python permite importar módulos prontos, com funções específicas para diferentes tarefas, como matemática, estatística ou gráficos. Assim as tarefas conseguem ser direcionadas com base nas bibliotecas selecionadas para executar aquela tarefa. Um exemplo acontece quando exportamos a biblioteca *math* que significa matemática e nela conseguimos utilizar todas as funções matemáticas no código gerado:

Quadro 10 – Bibliotecas

```
import math

print((math.sqrt(16))                                    # raiz quadrada de 16
```

Fonte: Autoria própria.

Os módulos e bibliotecas ampliam o que se pode fazer com a linguagem, principalmente em contextos educacionais, o docente por sua vez pode separar as componentes e ensinar exportar as bibliotecas para cada um deles.

viii. Estruturas de dados compostas

Essas estruturas armazenam vários dados ao mesmo tempo e são muito usadas para organizar informações. Observe o uso dos colchetes, parênteses, aspas e chaves. Cada um deles corresponde a uma estrutura de dado, com sua função em específico.

- Listas (*list*): sequência mutável – [1, 2, 3];
- Tuplas (*tuple*): sequência imutável – (1, 2, 3);
- Dicionários (*dict*): chave e valor – {"nome": "Rafaela"};
- Conjuntos (*set*): não permite repetição – {1, 2, 3}.

Quadro 11 – Estrutura de dados

```
numeros = [10, 20, 30]

print(numeros[1])                                    # mostra o número na posição 1 (20)
```

Fonte: Autoria própria.

ix. Entrada e saída de dados

Como já mencionado nos exemplos anteriores, para que um código rode um programa é necessário que exista uma entrada e uma saída dos dados. Para isso, em *Python* existem os comandos:

- Saída (*print*): mostra algo na tela;
- Entrada (*input*): permite ao usuário digitar algo, inserir uma informação.

Quadro 12 – Entrada e saída de dados

```
nome = input("Digite seu nome: ")

print( "Olá,", nome)
```

Fonte: Autoria própria.

Esse simples diálogo entre humano e computador já permite inúmeras atividades em sala de aula, é uma excelente opção para iniciar as codificações com os alunos, várias atividades podem ser propostas a partir desses dois comandos, para que o aluno se familiarize com a linguagem computacional.

x. Tratamento de erros

Quando um desenvolvedor está programando, nem sempre o código sai perfeito, o que é normal. Entretanto o *Python* oferece algumas funções que ajudam a melhorar e tratar esses erros, permitindo que o programa não trave quando algo inesperado acontece, assim permite que o aluno desenvolva o raciocínio lógico e responsabilidade sobre os comandos. Observe o exemplo:

Quadro 13 – Tratamento de dados

```
try:
    numero = int(input("Digite um número: "))
    print(10 / numero)
except:
    print("Algo deu errado. Verifique o valor digitado.")
```

Fonte: Autoria própria.

Observe os termos em inglês, *try* – tentar e *except* – exceto, servem como um alerta para que o programa não trave ao inserir comandos errados.

A familiarização com esses dez elementos constitui o alicerce para a criação de projetos matemáticos com *Python*. Assim, devem ser contextualizados de acordo com os problemas reais ou desafios estimulantes do aluno, conforme propõe a Aprendizagem Baseada em Problemas (ABP). Além disso, segundo a BNCC (Brasil, 2018), o uso de tecnologias digitais deve promover o desenvolvimento da autonomia, do pensamento crítico e da capacidade de resolver problemas — objetivos plenamente atendidos com a construção de projetos em *Python*.

Assim como defende Papert (1980, p. 12), ao usar linguagens de programação, o estudante "não apenas aprende a programar, mas programa para aprender". Assim, compreender esses fundamentos é o primeiro passo para transformar o computador em uma ferramenta de pensamento, criatividade e expressão matemática.

A partir dessa introdução a linguagem, no próximo ponto serão sugeridos alguns projetos simples, já utilizando os códigos apresentados anteriormente, aplicados a Matemática ensinada em sala de aula. Lembrando que a construção desses projetos pilotos, antes da aplicação da sequência didática, fica a critério do professor, podendo elaborar com tópicos mais pertinentes e de necessidade maior da turma analisada. Assim, a criatividade do docente pode ser explorada, promovendo maior interatividade com a linguagem.

4.4. CONSTRUÇÃO DE PROJETOS MATEMÁTICOS COM A LINGUAGEM *PYTHON*

A inserção de projetos iniciais simples e contextualizados constitui uma etapa essencial para a familiarização dos estudantes — e, sobretudo, dos docentes — com a linguagem *Python* no ambiente escolar. Antes de propor uma sequência didática mais estruturada, é necessário construir uma base de compreensão técnica e pedagógica que favoreça a apropriação da linguagem de forma significativa. Tais projetos, que chamaremos aqui de projetos piloto, funcionam como pontes entre os saberes prévios dos alunos e os novos conhecimentos a serem construídos, conforme propõe Ausubel (2003) com a teoria da aprendizagem significativa.

Nesse sentido, projetos de pequena escala, como calculadoras matemáticas, simuladores de média escolar ou até mesmo geradores de tabuada, podem atuar como

elementos mediadores na transição entre o ensino tradicional da matemática e sua aplicação em ambientes computacionais. Como destaca Papert (1980), quando o aluno programa, ele aprende não apenas sobre o computador, mas sobre o pensamento. Ao escrever códigos simples, os estudantes são desafiados a organizar ideias, testar hipóteses e visualizar resultados concretos, transformando a matemática em uma linguagem viva e interativa.

Além disso, tais projetos dialogam com os princípios metodológicos defendidos por autores como Hernández (1998), que vê na pedagogia de projetos um caminho para a construção do conhecimento interdisciplinar, ativo e centrado no sujeito. Ao permitir que o aluno pense com o código, como propõe Papert, e veja sentido nas atividades desenvolvidas, o professor se coloca em um papel de mediador de experiências que articulam tecnologia, lógica e conteúdo matemático de forma acessível.

A seleção dos conteúdos de multiplicação, aritmética e geometria para este projeto fundamenta-se em observações empíricas, coletadas pelos autores durante vivências em estágios e monitorias. Nesses contextos, constatou-se que tais temas representam os principais desafios de aprendizagem relatados pelos alunos. Trata-se de conceitos basilares, introduzidos nos anos iniciais do ensino, cuja complexidade aumenta progressivamente com o avanço das séries, podendo gerar lacunas no conhecimento.

Diante disso, a elaboração de projetos didáticos que revisitem e consolidem esses saberes mostra-se uma estratégia de grande valor. Para o discente, representa uma oportunidade de sanar dúvidas persistentes e solidificar sua base de conhecimento. Para o docente, surge como uma alternativa metodológica eficaz e diferenciada para a revisão de conteúdos essenciais, que frequentemente carecem de novas abordagens.

Assim, os projetos apresentados a seguir foram pensados para iniciar a prática com *Python* de forma acessível, mas significativa, tanto para alunos quanto para professores em formação. Eles antecipam e preparam o terreno para a sequência didática proposta nos próximos capítulos, permitindo que o foco posterior seja na aprendizagem conceitual mais aprofundada, sem deixar de lado a familiaridade técnica já desenvolvida.

Tabela 8 – Tabuada Interativa

PROJETO 1 – TABUADA INTERATIVA
Conteúdo: Multiplicação e Estrutura de Repetição (laços).

Objetivo: Criar um programa que gere a tabuada de um número informado pelo usuário. Ideal para fixar o conceito de multiplicação e o uso do *for*.

Código:

```
# O usuário escolhe um número
numero = int(input("Digite um número para ver a tabuada: "))

# Laço de repetição para gerar a tabuada
for i in range(1, 11):
    resultado = numero * i
    print(f "{numero} x {i} = {resultado}")
```

Explicação detalhada:

- **int(input(...)):** a função *input()* coleta uma informação digitada. O int() transforma essa informação em número inteiro;
- **range(1, 11)**: gera a sequência de 1 até 10 (o 11 não é incluído);
- **for i in range(...):** para cada valor de i (de 1 a 10), o programa realiza a multiplicação;
- **print(f"..."):** o f"" permite montar a frase com variáveis embutidas. É chamado de *f-string*.

Fonte: Autoria própria.

Neste primeiro projeto, o professor pode trabalhar várias operações além da multiplicação, explorando o máximo da criatividade e conceitos da linguagem associados à Matemática. É notório que a programação se mostra como um forte aliado prático no ensino-aprendizagem de qualquer componente que necessite o uso de estruturas lógicas e raciocínio. Um bom exemplo é, se o professor sente que determinada turma sente dificuldade em divisão, ele pode criar um projeto piloto utilizando a linguagem *Python* que seja focado nessa operação, assim consegue fazer com que o discente desenvolva várias habilidades durante a execução da tarefa, sendo uma alternativa para sanar dúvidas e possivelmente superar dificuldades.

Tabela 9 – Calculadora de média escolar

PROJETO 2 – CALCULADORA DE MÉDIA ESCOLAR
Conteúdo: Aritmética, Condicionais (*if*) e Função *round*.

Objetivo: Desenvolver um programa que calcula a média de 3 notas e informa se o aluno está aprovado ou não.

Código:

```
# Coletando as três notas
nota1 = float(input("Digite a primeira nota: "))
nota2 = float(input("Digite a primeira nota: "))
nota3 = float(input("Digite a primeira nota: "))

# Calculando a média aritmética
media = (nota1 + nota2 + nota3) / 3

# Exibindo a média com duas casas decimais
print("Média final:", round(media, 2))

# Verificando se o aluno passou
if media >= 7:
   print("Aluno aprovado!")
elif media >= 5:
   print("Aluno em recuperação.")
else:
   print("Aluno reprovado.")
```

Explicação detalhada:

- **float(input(...)):** transforma a entrada do usuário em número com casas decimais;
- **(nota1 + nota2 + nota3) / 3**: soma as notas e divide por 3;
- **round(media, 2):** arredonda a média para 2 casas decimais;
- **if, elif, else:** estruturas de decisão condicional que avaliam o valor da média.

Fonte: Autoria própria.

O segundo projeto é uma boa iniciativa para inserir a Matemática em contextos escolares. Muitas vezes, os alunos se veem em situações em que precisam calcular a média de suas notas durante o ano, auxiliá-los na construção de um código que automatize esse processo é de grande valia, pois além de fortalecer os conhecimentos da linguagem de programação e da Matemática, o aluno consegue associá-las a contextos do seu cotidiano, seja ele escolar ou não, assim essas tarefas instigam o pensamento do aluno em automatizar outras do seu dia a dia. Por fim, o docente, fica encarregado de estruturar essa atividade, de acordo com o método de avaliação de notas da instituição de ensino.

Tabela 10 – Calculadora de área de triângulo

PROJETO 3 – CALCULADORA DE ÁREA DE TRIÂNGULO
Conteúdo: Fórmulas geométricas, Entrada de dados e Operações aritméticas.
Objetivo: Calcular a área de um triângulo com base e altura informadas pelo usuário. Ideal para conectar a programação à geometria.

Código:
 	
`# Solicita base e altura do triângulo`
`base = float(input("Digite a base do triângulo (em cm): "))`
`altura = float(input("Digite a base do triângulo (em cm): "))`
 	
`# Fórmula da área do triângulo: (base * altura) / 2`
`area = (base * altura) / 2`
 	
`# Mostrando o resultado formatado`
`print(f"A área do triângulo é {area} cm²")`

Explicação detalhada: ● **Fórmula usada:** área = (base × altura) ÷ 2 ● O programa usa *float*() porque base e altura podem ter decimais. ● *f-string* novamente usada para mostrar o resultado de forma clara.

Fonte: Autoria própria.

A proposta do terceiro projeto se encaixa como um aliado no ensino da geometria, em especial em fórmulas de área de diversas figuras geométricas. Assim, pode proporcionar uma memorização mais efetiva com a articulação de códigos. Visto que, por ser um conteúdo com muitas fórmulas, os alunos podem sentir uma dificuldade em "decorá-las" todas em uma única vez, utilizando um único método de ensino-aprendizagem. Lorenzato (2006, p.88) argumenta que:

> A dificuldade dos alunos em aprender geometria está frequentemente associada à ênfase na memorização de fórmulas e regras, descoladas de qualquer contexto real ou visual. Quando a geometria não é ensinada por meio da experimentação, da visualização e da construção, perde-se o sentido do conteúdo.

A introdução do *Python* no ensino de Matemática não deve ser encarada como uma "moda" tecnológica, mas como uma mudança epistemológica: sair de um ensino centrado

na repetição e memorização para um ensino que valoriza a construção, a autoria e o pensamento lógico. Com isso, o próximo capítulo propõe uma sequência didática completa, alinhada à BNCC e inspirada nos princípios da Aprendizagem Significativa e do Construcionismo, para que o professor possa aplicar o *Python* como ferramenta de transformação na sala de aula de Matemática.

5. METODOLOGIA

Este capítulo tem como objetivo apresentar os caminhos metodológicos escolhidos para a construção e aplicação deste trabalho. Com isso, a partir da proposta de integrar a linguagem *Python* ao ensino de Matemática no Ensino Médio, foram elaboradas quatro sequências didáticas, cada uma voltada a um conteúdo específico e recorrente nas dificuldades dos alunos do ensino básico: expressões algébricas, função do primeiro grau, função do segundo grau e matemática aplicada ao cotidiano. Essas quatro frentes estruturam o projeto de maneira gradual, permitindo que os alunos avancem no conteúdo ao mesmo tempo em que desenvolvem suas habilidades em programação.

A escolha dos temas não foi feita ao acaso, os autores desse projeto, por já terem experienciando o estágio em sua graduação, podem afirmar que, durante as práticas da docência, notou-se a dificuldade de boa parte dos alunos em avançar na aprendizagem destes conteúdos em específico. Assim, houve a sensibilidade em construir esse material pedagógico em apoio aos docentes visto que essa abordagem é valiosa tanto para o aluno, que tem a chance de esclarecer dúvidas e conectar ideias, quanto para o professor, que ganha uma ferramenta pedagógica para reforçar temas fundamentais de maneira mais eficiente e engajadora.

Expressões algébricas são, muitas vezes, a porta de entrada para o pensamento simbólico na Matemática e o estudo das funções, mas acabam sendo trabalhadas de forma mecânica e descontextualizada, os discentes ainda não "enxergam" onde elas seriam aplicadas no seu dia a dia. Com isso, as funções de primeiro e segundo grau que geralmente são ensinadas posteriormente perdem ainda o mais o "sentido" e contexto real na aplicação cotidiana, por serem consideradas "mais difíceis" não visão discente.

A partir desses pontos a Matemática aplicada ao cotidiano visa consolidar os saberes de forma contextualizada, valorizando a interdisciplinaridade e a capacidade de resolução de problemas reais. A utilização do *Python* como ferramenta de apoio oferece uma abordagem dinâmica e interativa, permitindo que o estudante "veja" a Matemática funcionando por meio de códigos simples, conectando teoria e prática de maneira significativa.

Neste capítulo, serão detalhados o tipo de pesquisa adotado, a metodologia de base escolhida — com foco na Aprendizagem Baseada em Problemas (ABP) —, os instrumentos e procedimentos utilizados, bem como a justificativa para as escolhas realizadas. A proposta da sequência didática será também explicada de forma detalhada,

e no capítulo seguinte, o detalhamento das sequências, como cada atividade está alinhada à BNCC e ao desenvolvimento de competências matemáticas e digitais, com ênfase na resolução de problemas, raciocínio lógico, comunicação e pensamento computacional.

A intenção é mostrar não apenas o "como", mas o "porquê" de cada etapa do trabalho, reforçando que este projeto não se trata apenas de ensinar *Python* ou Matemática, mas de repensar a prática pedagógica a partir de uma nova lógica educacional: mais ativa, contextualizada e significativa, onde essas ferramentas e essa metodologia podem ser alinhadas e trabalhadas em paralelo ao ensino-aprendizagem.

5.1. TIPO DE PESQUISA

Quanto a metodologia, é desenvolvida com abordagem qualitativa, de natureza aplicada, e com delineamento exploratório e descritivo. De acordo com Gil (2008, p. 42), a pesquisa aplicada "busca gerar conhecimentos para aplicação prática e dirigidos à solução de problemas específicos", sendo adequada ao objetivo deste estudo: a elaboração de uma sequência didática para o ensino de expressões algébricas e funções de primeiro grau, funções de segundo grau e Matemática aplicada ao cotidiano, utilizando a linguagem *Python* como recurso pedagógico.

A abordagem da pesquisa é de natureza qualitativa, pois busca compreender e propor soluções educacionais que promovam o uso de tecnologias digitais na prática docente da Matemática. Como discutem Bogdan e Biklen (1994, p. 49), esse tipo de pesquisa se preocupa com os significados atribuídos pelos sujeitos aos fenômenos, sendo ideal para investigar práticas educativas e construir materiais pedagógicos contextualizados.

É importante ressaltar que o trabalho foi dividido em duas etapas principais:

Pesquisa Exploratória – Fundamentação Teórica

Inicialmente, realiza-se uma pesquisa exploratória com base em levantamento bibliográfico. Essa etapa visa compreender os fundamentos do ensino de Matemática com tecnologias, a relação entre a linguagem *Python* e os conteúdos matemáticos, e os princípios da Aprendizagem Baseada em Problemas (ABP). A pesquisa exploratória, conforme Triviños (1987, p. 110), permite ampliar o entendimento de temas pouco abordados, como o uso de programação por professores de Matemática em formação ou em atuação, permitindo formular hipóteses no redirecionamento de outras pesquisas.

Pesquisa Descritiva – Elaboração do Guia e da Sequência Didática

A segunda etapa consiste em uma pesquisa descritiva, voltada à construção do material didático. Será elaborado:

- Um guia introdutório voltado a professores que possuem ou não familiaridade com programação – inserido dentro do capítulo 4;
- Quatro sequências didáticas, baseado na metodologia da Aprendizagem Baseada em Problemas (ABP), que articula os conteúdos de expressões algébricas, funções de primeiro grau, funções de segundo grau e Matemática aplicada ao cotidiano, com atividades práticas em *Python*.

Essa etapa tem como foco detalhar como a linguagem de programação pode ser aplicada ao ensino da Matemática, facilitando a mediação pedagógica e promovendo o uso de recursos tecnológicos em sala de aula. Segundo Gil (2008, p. 28), a pesquisa descritiva visa "descrever as características de determinada população ou fenômeno", o que neste caso se traduz na descrição e estruturação de uma proposta metodológica.

Para a delimitação da pesquisa é importante destacar que este trabalho não contempla a aplicação prática da sequência didática, limitando-se à proposição e construção do material que poderá ser utilizado posteriormente por professores em sua prática pedagógica.

Aos autores, a intenção é dar continuidade a esta investigação com a aplicação prática da sequência didática, seja em um programa de pós-graduação lato sensu ou, futuramente, no âmbito de um mestrado acadêmico ou profissional. Essa perspectiva amplia o horizonte da pesquisa, conferindo-lhe potencial de impacto real no campo educacional. Além disso, ao projetar a execução concreta da proposta em contextos formativos, o trabalho ultrapassa o plano hipotético e evita permanecer no campo das suposições, respondendo, de forma propositiva, à indagação comum que ronda muitas produções acadêmicas: “será que funciona na prática?”.

5.2. A APRENDIZAGEM BASEADA EM PROBLEMAS (ABP) NAS TECNOLOGIAS DE ENSINO

A Aprendizagem Baseada em Problemas (ABP) surgiu na década de 1960, idealizada pelo médico e educador Howard Barrows, da McMaster University, no Canadá. Inicialmente aplicada à formação de estudantes de medicina, essa metodologia

foi criada para transformar a forma como os futuros profissionais se relacionam com o conhecimento: saindo de uma postura passiva para uma atuação ativa, investigativa e crítica diante dos desafios do mundo real.

Desde então, a ABP se expandiu para diversas áreas do saber — incluindo a Educação Básica — e passou a representar um poderoso instrumento para desenvolver habilidades essenciais no século XXI, como a autonomia intelectual, o pensamento crítico, a capacidade de colaboração e a resolução criativa de problemas. A proposta central é simples e profunda: os alunos aprendem melhor quando partem de uma situação-problema real, relevante e instigante, que os desafia a buscar soluções de forma coletiva e reflexiva.

No contexto do ensino de Matemática, essa metodologia encontra um terreno fértil. Ao invés de simplesmente memorizar fórmulas ou seguir procedimentos repetitivos, os estudantes são convidados a investigar, formular hipóteses, testar ideias, comparar resultados e tomar decisões fundamentadas. É nesse movimento que o conhecimento matemático ganha vida, significado e propósito. Como destacam Savery e Duffy (1995), a ABP promove um ambiente no qual aprender é uma consequência natural do ato de resolver problemas com sentido.

A lógica da ABP também dialoga com os fundamentos da Teoria Construcionista de Papert (1980), segundo a qual os alunos aprendem com mais profundidade quando constroem algo significativo para si e para os outros. Ao programar em *Python*, por exemplo, os estudantes não apenas aplicam conteúdos matemáticos, mas desenvolvem raciocínio lógico, organização de pensamento e criatividade. Eles passam a manipular ideias matemáticas de forma concreta, atribuindo função e intenção ao que antes poderia parecer apenas um conteúdo escolar.

Hoje, a ABP se mostra especialmente relevante em tempos nos quais o protagonismo estudantil e o uso significativo da tecnologia são indispensáveis. Ela rompe com a lógica de ensino centrada no professor e oferece aos alunos um papel ativo no processo de aprendizagem, permitindo que eles aprendam fazendo, discutindo e criando.

A escolha pela Aprendizagem Baseada em Problemas (ABP) como metodologia central deste trabalho não é apenas uma estratégia didática, mas uma decisão coerente com os objetivos que norteiam a proposta: formar sujeitos capazes de aplicar a Matemática em situações reais, desenvolver o pensamento computacional e agir com autonomia no processo de aprendizagem. Essa metodologia permite transformar a sala de aula em um espaço de investigação ativa, no qual o erro é parte do caminho e não um fim.

Trata-se de uma aprendizagem com propósito, onde o conteúdo ganha sentido à medida que é mobilizado para resolver situações concretas, ligadas à vida dos próprios estudantes.

Ela convida o aluno a deixar o papel passivo e assumir, junto ao professor, a responsabilidade pela construção do conhecimento.

5.3. SEQUÊNCIA DIDÁTICA

A sequência didática é uma ferramenta metodológica essencial para a organização e planejamento do processo de ensino-aprendizagem. Criada e desenvolvida por Dolz, Noverraz e Schneuwly (2004), no contexto da didática do francês como língua materna, essa proposta se consolidou como uma estratégia estruturante no campo educacional, permitindo que o professor organize os conteúdos em etapas articuladas, coerentes e progressivas. Sua aplicação se estendeu rapidamente a outras áreas do conhecimento, inclusive à Matemática, dada sua eficácia na promoção de aprendizagens significativas.

No ensino da Matemática, a sequência didática se apresenta como uma poderosa aliada para enfrentar os desafios que historicamente permeiam essa disciplina. Isso porque sua estrutura propõe momentos distintos e integrados de exploração, sistematização e aplicação dos conteúdos, favorecendo a construção de saberes de forma mais concreta e contextualizada. Como defendem Libâneo (2013) e Moran (2021), a aprendizagem se torna mais significativa quando o aluno compreende o sentido do que estuda e consegue estabelecer relações com o mundo que o cerca — algo que a sequência didática bem elaborada possibilita ao proporcionar uma organização lógica e funcional dos conhecimentos.

A Base Nacional Comum Curricular (BNCC) reforça essa necessidade ao destacar a importância de práticas pedagógicas que articulem competências cognitivas, sociais e tecnológicas. A BNCC valoriza o protagonismo do aluno, a resolução de problemas, o pensamento crítico e a utilização de recursos diversos, como linguagens digitais. Nesse contexto, a sequência didática permite ao professor alinhar os objetivos de aprendizagem às habilidades propostas pela Base, promovendo uma prática docente intencional, planejada e avaliativa.

Além disso, a utilização de sequências didáticas deve ser vista como obrigatória e indispensável na formação dos professores, não apenas como um instrumento de organização pedagógica, mas como um dispositivo que sustenta a reflexão sobre a própria prática. Como afirma Nóvoa (2009), o professor precisa ser autor e pesquisador de sua

atuação, e para isso, instrumentos como a sequência didática são fundamentais para sistematizar experiências, reavaliar estratégias e construir intervenções mais eficazes. Ainda, como defendem Severino (2016) e Papert (1980), a aprendizagem só ocorre com significado quando o aluno participa ativamente do processo, manipulando, testando, errando e reconstruindo o conhecimento — ações que uma sequência bem estruturada permite com maior naturalidade.

No presente trabalho, o uso da sequência didática é o fio condutor da proposta metodológica, pois garante coerência entre os conteúdos matemáticos, a utilização da linguagem *Python* e o desenvolvimento de competências previstas na BNCC. Ela orienta cada etapa das atividades, desde o resgate conceitual até a aplicação prática por meio da programação, organizando os saberes de forma acessível e conectada à realidade digital dos estudantes.

Portanto, mais do que uma simples ferramenta, a sequência didática representa um compromisso com uma educação mais significativa, planejada e transformadora. Seu uso deve ser incentivado desde a formação inicial, pois prepara o futuro professor para pensar suas aulas de forma crítica, criativa e alinhada às demandas contemporâneas do ensino da Matemática.

No próximo capítulo, serão apresentadas as sequências didáticas construídas pelos autores do trabalho, atreladas à metodologia ABP e o uso da linguagem Python, com conteúdos selecionados com critérios pré-estabelecidos pelos autores e suas vivências docentes.

6. SEQUÊNCIAS DIDÁTICAS

Dando continuidade ao desenvolvimento desta pesquisa, este capítulo apresenta a estruturação das sequências didáticas propostas para o ensino de conteúdos da Matemática por meio da linguagem de programação *Python*. Tais sequências foram elaboradas com base na metodologia de Aprendizagem Baseada em Projetos (ABP), buscando integrar teoria e prática de maneira significativa, criativa e contextualizada. A proposta é que o aluno atue como protagonista do processo de aprendizagem, sendo incentivado a aplicar conceitos matemáticos em situações concretas e resolver desafios com o auxílio da tecnologia.

É importante ressaltar que, para a efetiva aplicação das sequências aqui descritas, recomenda-se que o professor inicie esse processo com uma aula introdutória, conforme descrito no Capítulo 4 deste trabalho. Esse momento inicial é fundamental para ambientar os estudantes no universo da linguagem *Python*, mesmo aqueles que possuem pouco ou nenhum contato com ferramentas computacionais. A aula introdutória propõe um conjunto de conceitos básicos — como variáveis, operadores, estruturas de controle e funções — acompanhados de três projetos-pilotos que funcionam como primeiros experimentos criativos e orientados no uso do Python aliado à matemática. Tais atividades despertam o interesse dos alunos, reforçam o raciocínio lógico e preparam o terreno para os conteúdos mais específicos que serão trabalhados nas sequências seguintes.

Com esse repertório inicial assimilado, o professor pode então dar continuidade ao plano didático, desenvolvendo os conteúdos matemáticos com mais profundidade. Neste capítulo, são apresentadas quatro sequências principais, com foco nos seguintes temas: Expressões Algébricas, Função do Primeiro Grau, Função do Segundo Grau e Matemática Aplicada ao Cotidiano. Cada uma delas foi cuidadosamente elaborada com base na BNCC, respeitando os objetivos de aprendizagem e as habilidades esperadas para o Ensino Médio. A estrutura adotada contempla os componentes curriculares, os objetivos, os recursos didáticos, o tempo estimado, os métodos de avaliação e os resultados esperados, sempre articulando o uso do *Python* como ferramenta pedagógica.

Ao alinhar o ensino da matemática aos elementos da cultura digital, espera-se ampliar as possibilidades de atuação docente e de aprendizagem discente, promovendo experiências mais dinâmicas, significativas e compatíveis com as demandas contemporâneas da educação.

6.1. EXPRESSÕES ALGÉBRICAS

Esta primeira sequência didática foi elaborada com o objetivo de ensinar o conteúdo de expressões algébricas a estudantes do Ensino Médio, utilizando como ferramenta de apoio a linguagem de programação Python. A proposta contempla três momentos distintos: a apresentação e prática do conteúdo matemático em sala de aula, a introdução das expressões em Python por meio de exemplos guiados e, por fim, a construção de um pequeno projeto com aplicação prática do conteúdo na linguagem.

A sequência está estruturada para ser desenvolvida ao longo de duas a três aulas e se apoia na metodologia da Aprendizagem Baseada em Projetos (ABP). Essa abordagem permite que o aluno compreenda as expressões não apenas como fórmulas manipuláveis, mas como elementos funcionais dentro de um sistema computacional lógico. Além disso contempla também os objetivos de aprendizagem da BNCC, respeita o tempo pedagógico e considera os conhecimentos prévios adquiridos na aula introdutória

É importante ressaltar que cada instituição de ensino segue um conteúdo programático com base na apostila adotada. Assim, fica em aberto a ordem e escolha dos principais focos do conteúdo em si que o professor pode trabalhar, podendo até mesmo dar ênfase em partes específicas de maior dificuldade dos alunos. A intenção é que a sequência seja um guia, não precisa ser executada igualmente em todos os contextos, visto que cada sala de aula, docente, discentes e instituição de ensino tem suas vivências e metodologias.

SEQUÊNCIA DIDÁTICA

Componente curricular	Matemática
Conteúdo	Expressões algébricas – aplicação em problemas; Introdução ao uso de variáveis e operadores matemáticos no Python.
Público-alvo	Séries iniciais do Ensino Fundamental II e Ensino Médio
Duração da aula	3 aulas de 50 minutos cada (total: 150 minutos)
Recursos didáticos	Quadro - pincel/lousa digital - apostila do conteúdo - slides - computadores - projetor - acesso à internet - caderno
Habilidades - BNCC: EM13MAT101 EM13MAT302	• Reconhecer e utilizar a linguagem algébrica para representar e resolver problemas; • Resolver e elaborar problemas que envolvam a modelagem e a interpretação de situações por meio de expressões algébricas; • Compreender o papel das tecnologias na resolução de problemas matemáticos do cotidiano.

Objetivos

- Compreender o conceito de expressões algébricas e suas partes;
- Identificar, montar e interpretar expressões em diferentes contextos;
- Aplicar expressões algébricas simples no ambiente de programação Python;
- Estimular o raciocínio lógico-matemático e o uso da tecnologia como ferramenta de apoio;
- Desenvolver um pequeno projeto prático, utilizando Python, que envolva expressões algébricas.

Metodologia - Sequência didática

Aula 1 - Relembrando o conceito e praticando expressões

1. Introdução teórica (20 min):

O professor inicia com uma retomada sobre o que são expressões algébricas, revendo os conceitos de termos, coeficientes, variáveis, operadores e graus de uma expressão. A explicação é contextualizada com situações do cotidiano, como gastos mensais ou velocidade média. Nessa etapa deve ser utilizada a apostila da instituição de ensino ou o material adotado pelo docente.

2. Prática tradicional (20 min):

Distribuição de uma lista com exercícios simples de construção e resolução de expressões. Os alunos resolvem individualmente e depois em duplas, à critério do docente.

3. Socialização e fechamento (10 min):

Discussão coletiva das respostas, destacando erros comuns e formas diferentes de resolver.

Metodologia - Sequência didática

Aula 2 - Expressões algébricas em Python

1. Revisão rápida da linguagem Python (10 min):

Retomada da aula piloto com perguntas:

- O que são variáveis?
- Como fazer contas no Python?

Nessa etapa fica como sugestão deixar o slide como suporte, espelhando os quadros com todos os informativos relevantes como tipo de dados, operadores, estruturas de controle e o que for necessário.

2. Transposição dos exemplos (15 min):

O professor apresenta no projetor exemplos simples do cotidiano e resolve com Python:

código Python:

```
# Exemplo 1: valor de um produto com desconto
preco = 100
desconto = 0.10
valor_final = preco - (preco * desconto)
print("Valor com desconto:", valor_final)
```

Explica que isso corresponde à expressão:

V = p - (p × d)

Com interpretação contextual: **p** = preço, **d** = desconto, **V** = valor final.

```
# Exemplo 2: cálculo do valor pago por corrida de táxi
km = 12
preco_por_km = 3
bandeirada = 5
total = bandeirada + (km * preco_por_km)
print("Valor da corrida:", total)
```

Aqui, a expressão é:

C = b + (k × v)

Com interpretação contextual: **b** = bandeirada, **k** = km, **v** = preço por km, **V** = Valor da corrida.

Observações: as atividades serão contextualizadas ou estruturadas durante a aula, de acordo com o critério do professor. Fica como orientação pedir que os alunos já testem os códigos no programa em paralelo as explicações.

Metodologia - Sequência didática

Aula 3 - Mini projeto com expressões algébricas

1. Proposta do desafio (10 min):

O professor propõe que os alunos, em duplas, escolham uma situação do cotidiano que envolva expressões algébricas e criem um pequeno programa em Python que a represente e execute o cálculo. Exemplos sugeridos:

- Simular o salário mensal de um trabalhador com hora extra;
- Calcular o valor de uma compra com parcelas e juros;
- Estimar o tempo de uma viagem com base na distância e velocidade.

Nessa etapa, o professor pode associar as expressões algébricas com algum outro conteúdo ou componente que o aluno esteja trabalhando, promovendo a interdisciplinaridade.

2. Desenvolvimento (30 min):

Com base nos exemplos anteriores, os alunos escrevem seus códigos com apoio do professor - essa tarefa pode ser orientada e iniciada em sala, com isso o professor pode estipular um prazo maior, com suporte assíncrono para conclusão do projeto.

3. Apresentação (10 min):

Algumas duplas compartilham o que criaram, explicando o raciocínio.

Avaliação

Alguns pontos devem ser considerados na hora da avaliação:

- Participação nas discussões e práticas;
- Entendimento demonstrado na resolução de problemas tradicionais;
- Clareza e coerência no código Python desenvolvido no projeto;
- Capacidade de explicar a transposição da expressão algébrica para o código.

É importante ressaltar que, cada professor deve avaliar de acordo com a realidade da sala e da instituição, tais pontos devem ser levados em consideração.

Resultados esperados

- Melhor compreensão das expressões algébricas e seus usos;
- Capacidade de representar situações reais em forma algébrica e programável;
- Estímulo ao uso de tecnologia de forma crítica, criativa e matemática;
- Maior engajamento por meio da interdisciplinaridade.

Referências

BRASIL. Base Nacional Comum Curricular. Brasília: MEC, 2018. Disponível em: Melhor compreensão das expressões algébricas e seus usos; Capacidade de representar situações reais em forma algébrica e programável; Estímulo ao uso de tecnologia de forma crítica, criativa e matemática; Maior engajamento por meio da interdisciplinaridade.. Acesso em: 7 jun. 2025.

MORAN, José Manuel. Metodologias ativas para uma educação inovadora: uma abordagem teórico-prática. In: BACICH, L.; MORAN, J. M. (Org.). Metodologias ativas para uma educação inovadora: uma abordagem teórico-prática. Porto Alegre: Penso, 2018.

PAPERT, Seymour. A máquina das crianças: repensando a escola na era da informática. Porto Alegre: Artmed, 1994.

VALENTE, José Armando. Tecnologia e formação de professores: o enfoque da informática educativa. In: LITWIN, E. (Org.). Tecnologia Educativa: política, histórias e propostas. Porto Alegre: Artmed, 2003.

6.2. FUNÇÃO DE PRIMEIRO GRAU – AFIM

Após o trabalho com expressões algébricas e a consolidação dos fundamentos da linguagem *Python*, esta sequência didática tem como foco o ensino da função de primeiro grau, explorando sua representação algébrica, interpretação gráfica e aplicação em contextos reais. O objetivo é aliar a compreensão conceitual do conteúdo matemático com a utilização prática da linguagem de programação, favorecendo uma aprendizagem ativa, contextualizada e significativa.

A proposta inicia com uma revisão dos principais conceitos da função afim, como coeficiente angular, coeficiente linear e o comportamento gráfico da reta. A partir disso, os alunos são conduzidos a codificar esses elementos no *Python*, atribuindo valores às variáveis a e b e testando diferentes combinações para observar o impacto nos resultados.

O diferencial desta sequência está na aplicação do conteúdo por meio da criação de um programa interativo, em que os alunos utilizam os comandos *input()*, *print()* e operadores matemáticos para desenvolver uma calculadora de função do primeiro grau. O projeto final pode ser aprimorado com condicionais simples (*if*, *else*), permitindo a inclusão de verificações e respostas personalizadas com base nos valores inseridos.

Mais do que resolver equações, a proposta convida os estudantes a investigar padrões, modelar situações reais e compreender a função como uma ferramenta matemática que pode ser programada, testada e adaptada. Ao final da sequência, espera-se que os alunos reconheçam a utilidade da programação como aliada da matemática na resolução de problemas cotidianos e no desenvolvimento do raciocínio lógico.

SEQUÊNCIA DIDÁTICA

Componente curricular	Matemática
Conteúdo	Função do 1º grau (afim): interpretação, representação algébrica e construção de algoritmos em Python
Público-alvo	Séries iniciais do Ensino Fundamental II e Ensino Médio
Duração da aula	3 aulas de 50 minutos cada (total: 150 minutos)
Recursos didáticos	Quadro - pincel/lousa digital - apostila do conteúdo - slides - computadores - projetor - acesso à internet - caderno
Habilidades - BNCC: EM13MAT104 EM13MAT401	• Resolver e elaborar problemas envolvendo funções a partir de diferentes estratégias (tabelas, gráficos, expressões algébricas, planilhas e ferramentas digitais); • Utilizar tecnologias digitais para representar e resolver problemas por meio de linguagens matemáticas e computacionais.

Objetivos

- Revisar os principais conceitos da função de 1º grau e sua aplicação em problemas reais;
- Introduzir o uso da linguagem Python como recurso para testar valores e visualizar o comportamento da função afim;
- Estimular a lógica algorítmica por meio de comandos básicos (*input*, *print*, operadores, *if*, *else*);
- Desenvolver um projeto simples que integre a Matemática com programação, promovendo a aprendizagem ativa e contextualizada.

Metodologia - Sequência didática

Aula 1 - Relembrando o conceito e praticando expressões

1. Revisão do conteúdo matemático (20 min):

O professor revisa com a turma os elementos fundamentais da função afim:

- Forma geral: f(x) = ax + b;
- Coeficiente angular a e coeficiente linear b;
- Crescimento ou decrescimento da função;
- Ponto de interseção com o eixo y;
- Raiz da função (quando f(x) = 0).

Com base em exemplos numéricos simples, os alunos realizam pequenos exercícios manuais para reforçar a intuição sobre o comportamento da função.

Metodologia - Sequência didática

2. Introdução a linguagem Python (30 min):

O professor propõe um código inicial em Python para ilustrar o funcionamento da função afim com valores definidos manualmente, entende-se que nessa etapa o aluno já tem um conhecimento prévio da linguagem com base nos projetos iniciais. Veja o exemplo:

código Python:

```
a = 2
b = 3
x = 4
y = a * x + b
print(f"O valor de y para x = {x} é {y}")
```

Em seguida, os alunos são desafiados a testar diferentes valores para a, b e x, observando como as mudanças afetam o resultado da função. O objetivo aqui é estabelecer relação entre os parâmetros matemáticos e a execução do código.

Aula 2 - Aprimoramento e aplicação do código (50 min):

Nessa etapa o professor o professor libera a aula para os alunos melhorarem o código com alguns textos, operadores e se possível já aplicar os conceitos aprendidos em códigos práticos, como calcular lucro de vendas, variação de preços de um produto e etc. Nessa etapa é importante que o aluno tenha a liberdade e criatividade orientada pelo docente. Permita que o aluno manipule o código. Se preferir, utilize o código da aula passada como exemplo para ser aprimorado/aplicado à algum contexto. Assim, tomando esse exemplo, espera-se que o aluno siga por algo nesse caminho:

código Python:

```
a = float(input("Digite o valor de a: " ))
b = float(input("Digite o valor de b: " ))
x = float(input("Digite o valor de x: " ))
y = a * x + b
print(f"Para f(x) = {a}x + {b}, o valor de f({x}) é {y}")
```

A proposta é que os alunos explorem diferentes cenários e compreendam, com suporte visual e algorítmico, o comportamento da função a partir dos dados inseridos. Lembrando que esse é um exemplo de aprimoramento simples, o aluno terá a criatividade de criar e recriar o código no contexto que preferir.

Metodologia - Sequência didática

Aula 3 - Projeto Final - Calculadora de Função Afim (50 min)

O desafio final é criar um programa mais completo, com personalização e lógica condicional (*if*, *else*) que analise a função digitada:

código Python:

```
a = float(input("Digite o coeficiente a: "))
b = float(input("Digite o coeficiente b: "))
if a > 0:
    print("A função é crescente.")
elif a < 0:
    print("A função é decrescente.")
else:
    print("A função é constante.")
x = float(input("Digite um valor para x: "))
y = a * x + b
print(f"Para x = {x}, temos f(x) = {y}")
```

Sugestões para os alunos personalizarem o código:

- Exibir a raiz da função (valor de x quando y = 0);
- Criar mensagens visuais com *print( )* estilizadas;
- Utilizar mais condições para deixar o código mais interativo;
- Simular situações reais (ex: preço total em função da quantidade comprada).

Avaliação

A avaliação será contínua e formativa, considerando:

- Participação nas atividades práticas;
- Clareza e organização do código;
- Capacidade de relacionar a função matemática com o código;
- Entrega e explicação do projeto final simples.

Resultados esperados

Ao final da sequência, espera-se que os alunos:

- Compreendam a estrutura e o comportamento da função do 1º grau;
- Identifiquem os papéis dos coeficientes a e b em diferentes contextos;
- Usem a linguagem Python para simular, calcular e interpretar funções afins;
- Desenvolvam raciocínio lógico e autonomia na resolução de problemas.

Referências

BRASIL. Base Nacional Comum Curricular. Brasília: MEC, 2018.
GIL, Antonio Carlos. Didática do Ensino Superior. São Paulo: Atlas, 2008.
KENSKI, Vani Moreira. Tecnologias e Ensino Presencial e a Distância. Campinas: Papirus, 2012.
PAPERT, Seymour. A Máquina das Crianças: repensando a escola na era da informática. Porto Alegre: Artmed, 1980.
SEVERINO, Antônio Joaquim. Metodologia do Trabalho Científico. São Paulo: Cortez, 2007.

6.3. FUNÇÃO DE SEGUNDO GRAU – QUADRÁTICA

Dando continuidade ao desenvolvimento das sequências didáticas que integram conteúdos matemáticos ao uso da linguagem de programação Python, a próxima sequência, tem como proposta o ensino da função do segundo grau. Após já terem sido introduzidos à linguagem por meio da aula introdutória descrita no Capítulo 4 e da realização dos projetos pilotos, os alunos estarão mais familiarizados com comandos básicos como *print()*, *input(),* variáveis e operadores matemáticos, o que possibilita o avanço para construções mais elaboradas.

Nesta etapa, a proposta inicia com uma revisão conceitual da função quadrática, retomando elementos fundamentais como coeficientes, discriminantes e raízes. Em seguida, os alunos são conduzidos a inserir os coeficientes da equação no *Python*, compreendendo como representar expressões matemáticas com variáveis e realizar cálculos a partir delas. O objetivo é que a matemática não permaneça apenas no papel, mas ganhe vida com a execução de códigos simples e funcionais.

Por fim, propõe-se a criação de um projeto final prático, no qual os estudantes aprimoram o código da calculadora de função do 2º grau, utilizando recursos como *if*, *else* e *elif* para tratar diferentes situações (como raízes reais ou inexistentes). A proposta permite o desenvolvimento do pensamento computacional aliado à interpretação matemática, além de estimular a personalização e a criatividade na apresentação das soluções. Essa experiência contribui para que os alunos percebam como os conteúdos algébricos podem ser aplicados de forma concreta, funcional e próxima da realidade tecnológica que vivenciam.

SEQUÊNCIA DIDÁTICA

Componente curricular	Matemática
Conteúdo	Equações do segundo grau com aplicação prática em linguagem Python
Público-alvo	Séries finais do Ensino Fundamental II ou Ensino Médio.
Duração da aula	4 aulas de 50 minutos cada (total: 200 min)
Recursos didáticos	Quadro - pincel/lousa digital - apostila do conteúdo - slides - computadores - projetor - acesso à internet - caderno
Habilidades - BNCC EF08MA09 EM13MAT302	• Resolver e elaborar, com e sem uso de tecnologias, problemas que possam ser representados por equações polinomiais de 2º grau do tipo ax2 = b; • Construir modelos empregando as funções polinomiais de 1º ou 2º graus, para resolver problemas em contextos diversos, com ou sem apoio de tecnologias digitais

Objetivos

- Compreender a estrutura geral da equação do segundo grau;
- Resolver equações utilizando métodos tradicionais (fórmula de Bhaskara, fatoração, soma e produto);
- Interpretar o discriminante delta (Δ) e suas implicações no número de raízes reais;
- Aplicar a linguagem Python como ferramenta para automatizar a resolução dessas equações;
- Criar um projeto computacional simples para resolver equações do 2º grau.

Metodologia - Sequência didática

Aula 1 - Relembrando os conceitos

1. Introdução teórica (15 min):

O professor inicia com uma retomada do que é uma equação do segundo grau, destacando sua forma geral $ax^2 + bx + c = 0$, explicando o papel dos coeficientes a, b e c, e reforçando que a nunca pode ser zero. Em seguida, apresenta a fórmula de *Bhaskara* como método para resolver esse tipo de equação, explicando cada parte da fórmula e o significado do discriminante $\Delta = b^2 - 4ac$. A explicação é contextualizada com situações do cotidiano que envolvem esse tipo de equação, como cálculos de altura de um objeto lançado ao ar ou problemas com áreas. O professor pode utilizar quadro, slides ou apostila como apoio visual.

Metodologia - Sequência didática

2. Prática (25 min):
Distribuição de uma lista de exercícios simples com equações do segundo grau para que os alunos resolvam utilizando a fórmula de *Bhaskara*. Os exercícios devem contemplar casos com duas raízes reais distintas, uma raiz real (raiz dupla) e nenhum valor real. Os alunos resolvem individualmente ou em duplas, conforme orientação do professor.

3. Socialização e fechamento (10 min):
Discussão coletiva das respostas, destacando os erros mais comuns (como troca de sinais ou erros no cálculo do discriminante) e reforçando a interpretação dos resultados com base no valor de delta (Δ). O professor pode finalizar com um exemplo extra, resolvido passo a passo com a turma.

Aula 2 - Equação do segundo grau com Python

1. Retomada da aula anterior e introdução à linguagem Python (15 min):
Retomada rápida dos conceitos principais trabalhados anteriormente, especialmente a forma geral da equação do segundo grau e a fórmula de *Bhaskara*. O professor introduz a linguagem Python com perguntas norteadoras como:

- Como podemos automatizar esse cálculo?
- O que seria uma forma de resolver isso com o computador?

Nesta etapa, é sugerido utilizar slides ou o quadro digital para apresentar os principais elementos da linguagem Python, como variáveis, operadores matemáticos e a função print(), preparando o terreno para a transposição do conteúdo matemático para o ambiente de programação.

2. Demonstração prática com código Python (20 min):
O professor projeta um código simples que resolve uma equação do segundo grau usando a fórmula de *Bhaskara* em Python. O código deve ser explicado linha por linha, destacando os pontos em comum com a expressão matemática tradicional.

código Python

```
# Exemplo 1: resolvendo uma equação de segundo grau
a = 1
b = -3
c = -4
delta = b**2 - 4*a*c
x1 = (-b + delta**0.5) / (2*a)
x2 = (-b - delta**0.5) / (2*a)
print("As raízes da equação são:", x1, "e", x2)
```

Metodologia - Sequência didática

3. Interpretação do código (15 min):

O docente interpreta o código junto aos alunos, explicando cada linha e relacionando com a equação $ax^2 + bx + c = 0$ e a fórmula de *Bhaskara*. Enfatiza que:

- a, b e c são os coeficientes;
- delta é o discriminante (Δ);
- x1 e x2 são as soluções da equação;
- O operador **0.5 representa a raiz quadrada.

A aula finaliza com perguntas reflexivas e sugestões para que os alunos, em aula futura, experimentem outros valores para os coeficientes, observando o comportamento das raízes conforme o valor de Δ.

Aula 3 - Analisando o discriminante

1. Aplicação prática do código (20 min):

Nesta aula, os alunos irão utilizar computadores ou celulares (com ambiente Python disponível, como o *Google Colab*/*Vscode* ou o aplicativo *Pydroid*) para testar o código apresentado anteriormente. Cada aluno ou dupla digita o código base da fórmula de *Bhaskara* e insere diferentes valores para os coeficientes a, b e c, observando como os resultados mudam conforme o valor do discriminante (Δ). O objetivo é fixar a lógica da fórmula através da experimentação.

2. Variações orientadas pelo professor e atividade (20 min):

O professor propõe pequenas modificações no código original, como:

- Inserir mensagens personalizadas nos resultados (print());
- Exibir os resultados com duas casas decimais (round(x1, 2));
- Permitir que o usuário digite os valores de a, b e c com input();
- Incluir condicionais da função.

Atividade prática – Desafio no Python:

Os alunos devem adaptar o código para que, além de calcular as raízes, o programa também informe a natureza da equação com base no valor de delta (Δ), utilizando condicionais (*if*, *elif*, *else*). Exemplo do comportamento esperado:

código Python

```
a = float(input("Digite o valor de a: " ))
b = float(input("Digite o valor de b: " ))
c = float(input("Digite o valor de c: " ))
delta = (b**2)-4*a*c
if a == 0:
  print("O valor de a, deve ser diferente de 0")
 elif delta < 0
    print("Sem raízes reais")
else:
    x1 = (-b + delta**0.5)/(2*a)
    x2 = (-b - delta**0.5)/(2*a)
print(f"As raízes são {x1} e {x2}.")
```

6.4. A MATEMÁTICA APLICADA AO COTIDIANO

O ensino da Matemática ganha maior relevância e engajamento quando os estudantes conseguem perceber a presença dos conceitos matemáticos em situações reais do seu dia a dia. A Base Nacional Comum Curricular (BNCC) destaca a importância de promover competências que articulem a Matemática ao exercício da cidadania, ao raciocínio lógico e ao pensamento crítico (Brasil, 2018). Nesse contexto, a abordagem de conteúdos por meio de problemas práticos e contextualizados torna-se fundamental para o desenvolvimento de aprendizagens significativas.

Um dos desafios frequentes enfrentados por docentes é justamente tornar o componente mais próximo da realidade dos alunos, principalmente em um cenário marcado pela intensa presença das tecnologias e pela necessidade de desenvolver habilidades ligadas à autonomia, tomada de decisões e planejamento. Ao integrar a linguagem *Python* ao ensino de Matemática, abre-se a possibilidade de explorar essas competências de forma dinâmica, interativa e com forte apelo para os jovens, que veem na programação uma linguagem próxima de sua realidade digital.

A sequência didática apresentada a seguir parte dessa perspectiva e propõe uma atividade voltada à Matemática aplicada ao cotidiano, com foco na organização financeira pessoal. Trata-se de um tema relevante para adolescentes, pois envolve reflexões sobre consumo, economia, responsabilidade e projeção de gastos — aspectos essenciais para a vida adulta. Além disso, a atividade utiliza estruturas básicas de programação para representar expressões matemáticas, porcentagens e organização de dados, estimulando o raciocínio lógico-matemático em um contexto realista.

Ao empregar o *Python* como ferramenta pedagógica, pretende-se também contribuir para o desenvolvimento do pensamento computacional, conforme orienta a BNCC, favorecendo a construção de conhecimentos por meio da manipulação de informações, da resolução de problemas e da criação de soluções úteis no cotidiano escolar e pessoal dos estudantes.

SEQUÊNCIA DIDÁTICA

Componente curricular	Matemática
Conteúdo	Educação financeira básica, porcentagem, organização de dados, expressões numéricas.
Público-alvo	Ensino Médio
Duração da aula	3 aulas de 50 minutos cada (total: 150 minutos)
Recursos didáticos	Quadro - pincel/lousa digital - material orientados - slides - computadores - projetor - acesso à internet - caderno
Habilidades - BNCC: EM13MAT101 EM13MAT401 EM13COMP05	• Resolver e elaborar problemas com números racionais em diferentes contextos, incluindo financeiros • Utilizar e interpretar diferentes linguagens de programação como ferramenta de modelagem matemática; • Compreender e usar tecnologias digitais de forma crítica, significativa e ética.

Objetivos

- Relacionar conceitos matemáticos a situações reais e cotidianas;
- Desenvolver noções de planejamento financeiro pessoal;
- Aplicar expressões matemáticas com variáveis e porcentagens em linguagem Python;
- Estimular a autonomia e o raciocínio lógico por meio de pequenos projetos computacionais;

Metodologia - Sequência didática

Aula 1 - Apresentação do tema: organização financeira

1. Introdução teórica (20 min):

Inicie a aula perguntando ao aluno: "Se você recebesse R$800,00 por mês, conseguiria organizá-lo bem? O que você compraria? Quanto economizaria?" Proponha um bate-papo rápido para introduzir o tema organização financeira e como a Matemática nos ajuda a tomar boas decisões.

O professor pode trazer algum material teórico para discutir com os alunos ou abrir um momento para um bate-papo voltado a realidade de cada um deles e sua organização financeira.

2. Revisão e retomada (30 min):

Relembre brevemente conceitos de:

- Porcentagem.
- Soma e subtração com números decimais.
- Expressões matemáticas simples com variáveis.

Metodologia - Sequência didática

Aula 2 - Apresentação do código e testes

1. Introdução ao Código (30 min):

Apresente o código com calma, linha por linha, recapitule todos os conceitos aprendidos nas aulas anteriores, volte os slides propostos se necessário:

Código Python:

```
# Simulador de orçamento mensal

# Receita mensal
salario = float(input("Digite seu salário mensal: R$ "))

# Gastos fixos
aluguel = float(input("Quanto você gasta com moradia? R$ "))
transporte = float(input("Gastos com transporte? R$ "))
alimentacao = float(input("Gastos com alimentação? R$ "))
outros = float(input("Outros gastos fixos? R$ "))

# Total de gastos
total_gastos = aluguel + transporte + alimentacao + outros

# Economia sugerida (20% do salário)
economia_ideal = salario * 0.20

# Saldo restante
saldo = salario - total_gastos

# Resultados
print("\nResumo Financeiro:")
print(f"Gastos totais: R$ {total_gastos:.2f}")
print(f"Saldo restante: R$ {saldo:.2f}")
print(f"Economia ideal (20%): R$ {economia_ideal:.2f}")

if saldo >= economia_ideal:
    print("Parabéns! Você está economizando bem!")
elif saldo > 0:
    print("Você tem saldo, mas poderia economizar um pouco mais.")
else:
    print("Atenção! Seus gastos estão maiores que sua receita.")
```

Metodologia - Sequência didática

Aula 2 - Apresentação do código e testes

2. Exploração e início de atividade prática (20 min)

Oriente os alunos a testarem o código, mudando os valores e observando os resultados. Incentive-os a simular diferentes cenários (ganhando pouco, ganhando muito, gastando mal, etc.).

Aula 3 - Desafio final - Projeto individual (50 min)

Peça aos alunos que criem uma nova versão do simulador, com mais categorias (como lazer, compras, celular etc.) ou que permita planejar dois meses seguidos. Estimule a criatividade e a personalização e por fim peça que eles apresentem a turma.

Avaliação

- Participação nas discussões e testes com o código;
- Capacidade de adaptar o código com novos dados;
- Clareza ao interpretar os resultados gerados pelo programa;
- Entrega do mini projeto final com explicações sobre como construíram e usaram os cálculos.

É importante ressaltar que, cada professor deve avaliar de acordo com a realidade da sala e da instituição, tais pontos devem ser levados em consideração.

Resultados esperados

- Compreensão dos conceitos matemáticos aplicados ao cotidiano;
- Capacidade de uso básico da linguagem Python como ferramenta de modelagem matemática;
- Apropriação de noções de planejamento financeiro;
- Aumento da motivação dos estudantes ao verem utilidade real da Matemática.

Referências

BRASIL. Base Nacional Comum Curricular (BNCC). Brasília: MEC, 2018.
GIL, Antonio Carlos. Métodos e Técnicas de Pesquisa Social. 6. ed. São Paulo: Atlas, 2008.
KENSKI, Vani Moreira. Tecnologias e Ensino Presencial e a Distância. Campinas: Papirus, 2012.
PAPERT, Seymour. A Máquina das Crianças: repensando a escola na era da informática. Porto Alegre: Artmed, 1980.
SEVERINO, Antônio Joaquim. Metodologia do trabalho científico. 23. ed. São Paulo: Cortez, 2007.

7. CONSIDERAÇÕES FINAIS

Este trabalho buscou apresentar uma proposta de sequência didática voltada ao ensino de conteúdos matemáticos comumente identificados como desafiadores pelos alunos — como multiplicação, aritmética e geometria — aliando-os à linguagem de programação *Python* e à metodologia da Aprendizagem Baseada em Problemas (ABP). A escolha desses conteúdos não foi aleatória: ela emergiu das vivências dos autores em estágios supervisionados e monitorias, em que se evidenciaram lacunas significativas na aprendizagem matemática dos estudantes. Ao propor uma abordagem inovadora, o presente estudo se fundamenta em referenciais teóricos sólidos, como Papert (1980), Libâneo (2013), Moran (2021), Severino (2016), entre outros, que defendem uma educação ativa, crítica, reflexiva e em sintonia com as demandas contemporâneas.

A construção da sequência didática aqui apresentada é mais do que uma organização de etapas: é uma estratégia pedagógica estruturada com base na BNCC, que visa tornar o ensino mais significativo, contextualizado e conectado às reais necessidades dos alunos. A utilização do *Python* nesse processo se mostra como uma potente ferramenta de mediação entre o pensamento matemático e a lógica computacional, permitindo ao aluno desenvolver não apenas conteúdos específicos, mas também habilidades como raciocínio lógico, resolução de problemas e autonomia intelectual. Como defende Papert, quando o aluno programa, ele aprende sobre o pensamento, sobre a linguagem e sobre o mundo — um aprendizado ativo e transformador.

A metodologia da ABP, por sua vez, possibilita a construção do conhecimento de forma colaborativa e centrada em situações-problema, promovendo o protagonismo estudantil e aproximando o processo de aprendizagem da realidade. Essa escolha dialoga com os pressupostos de autores como Severino (2016) e Moran (2021), que defendem uma educação centrada na investigação, na experiência e na ação. Os alunos, imersos em um mundo digital, demandam práticas educativas que façam sentido em sua rotina, que considerem suas formas de pensar, comunicar e interagir com o conhecimento. Ignorar essa realidade é perpetuar um modelo de ensino desalinhado com o presente e incapaz de preparar para o futuro.

Embora as sequências propostas ainda não tenham sido aplicadas na prática, elas se apresentam como sugestões valiosas para professores em formação e em exercício, servindo de inspiração para ações em sala de aula e, futuramente, como base para um projeto de pesquisa em nível de pós-graduação ou mestrado. Nessas etapas acadêmicas,

será possível aprofundar os aspectos quantitativos e qualitativos, aplicar a sequência com turmas reais, mensurar os resultados de aprendizagem e refinar a proposta com base nas evidências colhidas.

É importante destacar que este trabalho não se encerra em si mesmo. Ele é um ponto de partida. A partir dessa experiência piloto, outras sequências podem — e devem — ser criadas, explorando diferentes conteúdos matemáticos, outras linguagens computacionais, novos contextos de ensino. O professor que domina a estrutura de uma sequência didática tem em mãos uma ferramenta poderosa para planejar, inovar e transformar suas práticas pedagógicas. E isso é urgente. Como diz Moran (2021), não basta ensinar com tecnologia, é preciso ensinar para a vida em uma sociedade tecnológica.

Ao resgatar o construcionismo de Papert, reafirma-se aqui a ideia de que os alunos constroem melhor seu conhecimento quando estão engajados em construir algo significativo para eles — seja um código, uma solução ou uma ideia. A escola precisa deixar de ser um espaço de transmissão e passar a ser um espaço de criação.

Educar com sentido é ensinar o aluno a pensar. E programar é pensar com profundidade. Se queremos uma educação transformadora, precisamos ousar. A tecnologia, quando bem utilizada, não é distração — é revolução. E toda revolução começa com um passo: uma boa sequência.

REFERÊNCIAS BIBLIOGRÁFICAS

BARDIN, Laurence. *Análise de conteúdo*. São Paulo: Edições 70, 2011.

BASE NACIONAL COMUM CURRICULAR. *Educação é a base*. Brasília: MEC, 2018. Disponível em: https://basenacionalcomum.mec.gov.br/. Acesso em: 04 abr. 2025.

BOGDAN, Robert; BIKLEN, Sari. *Investigação qualitativa em educação: uma introdução à teoria e aos métodos*. Porto: Porto Editora, 1994.

BRASIL. *Lei nº 9.394, de 20 de dezembro de 1996*. Estabelece as diretrizes e bases da educação nacional. *Diário Oficial da União*: seção 1, Brasília, DF, 23 dez. 1996. Disponível em: http://www.planalto.gov.br/ccivil_03/leis/l9394.htm. Acesso em: 17 maio 2025.

CORMEN, Thomas H. et al. *Algoritmos: teoria e prática*. 3. ed. Rio de Janeiro: Elsevier, 2012.

D'AMBROSIO, Ubiratan. *Educação matemática: da teoria à prática*. 4. ed. Campinas: Papirus, 1996.

DOLZ, J.; NOVERRAZ, M.; SCHNEUWLY, B. *Sequências didáticas para o oral e a escrita: apresentação de uma ferramenta para a didática do francês*. Campinas, SP: Mercado das Letras, 2004.

FIORENTINI, Dario; LORENZATO, Sergio. *Formação de professores de matemática: explorando novos caminhos com outros olhares*. Campinas: Autores Associados, 2009.

FRITZ, Christopher; KELLEY, Todd. Teaching programming environments: IDEs and online platforms. *Journal of Computing in Higher Education*, v. 30, n. 1, p. 1–22, 2018.

GARCIA, Lívia T. et al. A utilização de ambientes integrados de desenvolvimento no ensino de programação. *Revista Brasileira de Informática na Educação*, v. 26, n. 2, p. 183–202, 2018.

GIL, Antonio Carlos. *Como elaborar projetos de pesquisa*. 6. ed. São Paulo: Atlas, 2023.

GIL, Antonio Carlos. *Métodos e técnicas de pesquisa social*. 7. ed. São Paulo: Atlas, 2023.

KENSKI, Vani Moreira. *Educação e tecnologias: o novo ritmo da informação*. 7. ed. Campinas: Papirus, 2012.

LÉVY, Pierre. *Cibercultura*. Tradução de Carlos Irineu da Costa. São Paulo: Editora 34, 1999.

LIBÂNEO, José Carlos. *Didática*. 29. ed. São Paulo: Cortez, 2021.

RAMALHO, Luciano. *Python fluente: programação clara, concisa e eficaz*. 2. ed. São Paulo: O'Reilly Media; Novatec Editora, 2022.

MORAN, José Manuel. *A educação que desejamos: novos desafios e como chegar lá*. 2. ed. Campinas: Papirus, 2015.

MORAN, José Manuel. *A reinvenção das práticas pedagógicas na era digital*. 2. ed. São Paulo: Loyola, 2021.

MORAN, José Manuel. *Metodologias ativas para uma aprendizagem mais significativa*. Campinas, SP: Papirus, 2021.

MORAN, José Manuel. *Metodologias ativas para uma educação inovadora: uma abordagem teórico-prática*. 2. ed. Campinas: Papirus, 2021.

NÓVOA, António. *Professores: imagens do futuro presente*. Lisboa: Educa, 2009.

PAPERT, Seymour. *A máquina das crianças: repensando a escola na era da informática*. Tradução de Sandra Costa. Porto Alegre: Artmed, 1994.

PAPERT, Seymour. *Desafios à mente: computadores, crianças e o uso criativo da tecnologia*. Tradução de Sandra Costa. Brasília: Editora da UnB, 1980.

PAPERT, Seymour. *Mindstorms: children, computers, and powerful ideas*. 2. ed. New York: Basic Books, 1993.

SAAVEDRA, O.; ALONSO, F. Plataformas colaborativas e aprendizagem online: o Google Colaboratory. *Revista Educação e Tecnologia*, v. 16, n. 1, p. 95–108, 2020.

SANTOS, Lulu. *Tempos Modernos*. In: [álbum homônimo]. Warner Music Brasil, 1982. Música.

SANTOS, Maria F. dos; LIMA, João C. de. Uso de plataformas digitais para o ensino de programação: análise comparativa e aplicação. *Revista de Ensino de Ciências Exatas e Tecnológicas*, v. 5, n. 2, p. 121–139, 2021.

SEVERINO, Antônio Joaquim. *Metodologia do trabalho científico*. 27. ed. rev. e ampl. São Paulo: Cortez, 2021.

TRIVIÑOS, Augusto Nibaldo Silva. *Introdução à pesquisa em ciências sociais: a pesquisa qualitativa em educação*. São Paulo: Atlas, 1987.

VALENTE, José Armando. O computador na sociedade do conhecimento. In: VALENTE, J. A.; ALMEIDA, M. E. B. de. *Tecnologias na educação: ensinando e aprendendo com as tecnologias*. São Paulo: Unicamp/Nied, 2009. p. 15–38.

VASCONCELOS, E. A.; DIAS, M. T. O uso de IDEs no processo de ensino-aprendizagem da programação. *Educação & Tecnologia*, v. 12, n. 1, p. 45–60, 2019.

WING, Jeannette M. Computational thinking. *Communications of the ACM*, v. 49, n. 3, p. 33–35, 2006. Disponível em: https://doi.org/10.1145/1118178.1118215. Acesso em: 04 abr. 2025.