UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
Programa de Pós-Graduação em Matemática
Mestrado Profissional - PROFMAT/CCT/UFCG

João Evayr de Souza

O Uso da Linguagem de Programação Python na Resolução de Problemas Matemáticos do Ensino Médio

Campina Grande - PB
Março/2023

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
Programa de Pós-Graduação em Matemática
Mestrado Profissional - PROFMAT/CCT/UFCG

João Evayr de Souza

O Uso da Linguagem de Programação Python na Resolução de Problemas Matemáticos do Ensino Médio

Trabalho de Conclusão de Curso apresentado ao Corpo Docente do Programa de Pós-Graduação em Matemática - CCT - UFCG, na modalidade Mestrado Profissional, como requisito parcial para obtenção do título de Mestre.

Orientador: Dr Rodrigo Cohen Mota Nemer

Campina Grande - PB
Março/2023

S729u Souza, João Evayr de.
O uso da linguagem de programação Python na resolução de problemas matemáticos do ensino médio / João Evayr de Souza. – Campina Grande, 2023.
124 f. : il. color.

Dissertação (Mestrado Profissional em Matemática) – Universidade Federal de Campina Grande, Centro de Ciências e Tecnologia, 2023.
"Orientação: Prof. Dr. Rodrigo Cohen Mota Nemer".
Referências.

1. Linguagem de Programação Python. 2. Resolução de Problemas Matemáticos. 3. Ensino Médio. I. Nemer, Rodrigo Cohen Mota. II. Título.

CDU 512(043)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA MARIA ANTONIA DE SOUSA CRB 15/398

João Evayr de Souza

O Uso da Linguagem de Programação Python na Resolução de Problemas Matemáticos do Ensino Médio

Trabalho de Conclusão de Curso apresentado ao Corpo Docente do Programa de Pós-Graduação em Matemática - CCT - UFCG, na modalidade Mestrado Profissional, como requisito parcial para obtenção do título de Mestre.

Trabalho aprovado. Campina Grande - PB, 03 de março de 2023:

Rodrigo Cohen Nemer

Dr Rodrigo Cohen Mota Nemer
Orientador

Maria Isabelle Silva Dias Yanes

Dra Maria Isabelle Silva Dias Yanes
Examinadora externa - UEPB

Deise Mara B. de Almeida

Dra Deise Mara Barbosa de Almeida
Examinadora interna - UFCG

Campina Grande - PB
Março/2023

Dedico a todos os professores da rede pública de ensino, que enfrentam muitos desafios para honrarem com dignidade a nossa profissão.

Agradecimentos

A princípio, rendo graças a Deus Pai Eterno que, por meio de Cristo Jesus, Seu Filho, no Santo Espírito, tem abençoado a minha vida com inúmeras bênçãos espirituais.

Na caminhada para a obtenção do título de Mestre em Matemática, obtive apoio, incentivo e torcida, tanto de pessoas próximas e importantes em minha vida, como também de pessoas que conheci ao longo do caminho e conquistaram o meu apreço e minha consideração. Cada uma delas tem uma importância singular e, agora, agradeço de modo particular e especial.

Agradeço aos meus pais, José Moreno e Maria do Carmo, inclusive aos meus irmãos Edivânio, Edivaldo, Everaldo, Elenice e Egídio, por todo afeto, amor e cordialidade. Obrigado pelo apoio e incetivo ofertado a mim nessa trajetória acadêmica, pelas mensagens que expressavam preocupações com as minhas viagens rotineiras à Campina Grande, por compreenderem as minhas ausências nos eventos familiares e por acreditarem no meu potencial mais do que eu mesmo acredito.

Agradeço pela existência do afeto dos meus sobrinhos que muitas vezes salvaram os meus dias, ofertando leveza em dias de estresse e sobrecarga. Obrigado a Carlos, Pedro, Emilly, Bianca, Eloísa, Lorenzo, Érica, Elisa e Lavínia.

Quando iniciei a preparação para a caminhada desse mestrado, deparei-me com os dias finais da vida do meu avô e, junto a eles, com um dos maiores ensinamentos que a vida e o meu avô poderiam me dar. Gratidão ao dono do maior senso de humor, que via graça na desgraça. Agradeço ao meu avô Joaquim Padre (*in memoriam*) por ter transfigurado o meu *mindset* com seu exemplo de serenidade, autoconhecimento e sabedoria, sobretudo no momento de sua partida. Obrigado, Vovô!

Manifesto minha gratidão aos meus primos, irmãos e amigos de infância por sempre se mostrarem solícitos para ouvir meus desabafos e sempre perguntarem como estavam os progressos e ânimos nessa trajetória acadêmica. Obrigado a Larissa Souza, Miriellem Souza, Lucas Souza, David Dino, Estefany Freires, Sérgio Silva, Ismael Diniz, Janaide Araújo, Felipe Silva e Fabrício Santos.

Sou grato a toda a equipe da Escola de Referência em Ensino Médio Irnero Ignacio (EREMII). Aos colegas de trabalho que sempre se mostraram solícitos e dispostos a ajudar. Obrigado pelo vínculo fraterno construído, por todo apoio, e toda amizade ofertada ao dividirmos experiências profissionais. Caminhamos juntos tentando evoluir e promover uma educação de qualidade. De modo especial, agradeço a Alda Príncipe, Clebber Max, Dênio Melo, Lúcia Helena, Aline Santos, Murilo Gomes, Nathalia Queiroz, Edlene Moura, Vilma Queiroz e José Nunes.

Não poderia deixar de agradecer à UDP da GRE Sertão do Alto Pajeú por auxiliar-me na solicitação de afastamento para curso e fazer a interlocução com a SAD-PE.

Agradeço imensamente a toda equipe da Autarquia Educacional de Serra Talhada (AESET), inclusive aos meus alunos e residentes do curso de Licenciatura em Matemática da Faculdade de Formação de Professores e Serra Talhada (FAFOPST), por todas as trocas de experiências profissionais e cordialidades ofertadas, o que apoiou a minha evolução profissional.

Ao percorrer essa jornada do Mestrado Profissional em Matemática, tive o amparo de um grupo especial de Amigos Matemáticos que carrego desde a graduação, onde alguns egressos do PROFMAT e companheiros de profissão promoviam debates enriquecedores e instigantes que contribuíram para o meu crescimento profissional. Manifesto a minha gratidão a Airton Magalhães, Alex Magalhães, Fagner Magalhães, Isaías Lima, Matheus Queiroz, Ricardo Eryton, Tiago Melo e William Santana.

Agradeço, de modo especial, aos meus alunos que constantemente demonstraram sua admiração e afeto e que, ao debatermos sobre conceitos científicos, sempre existiram conversas sobre as alegrias e tristezas da vida, o que possibilitou a construção de amizades que levaremos para a vida toda. Em nome de todos os meus alunos, agradeço de modo especial à Viviany Brazil, Francimagno Lacerda, Cecília Alves e Magda Pereira, a este pequeno grupo, titulado por "As Corra Linda de Pai", que me promoveu apoio em dias pesados dessa rotina acadêmica.

Tenho uma imensa gratidão por uma pessoa que o Espírito Santo me apresentou: Tina Lima. A minha parceira de missões cristãs que dividiu comigo os ânimos e as cargas emocionais dessa vida de mestrando, do início ao fim, pois demos uma pausa nas missões e começamos o mestrado juntos, cada qual em sua área. Com esta licenciada em química, dividi experiências que fortaleciam as energias nas vivências acadêmicas.

Outro amigo que a vivência cristã me presenteou foi Marcus Vinícius Souza. Um excelente ouvinte e companheiro de boas práticas. Expresso o meu singelo obrigado a este amigo, pelos debates pertinentes que sempre me promoveram apoio e encorajamento.

Tenho uma imensa gratidão à Paróquia Nossa Senhora da Penha da cidade de Serra Talhada-PE. Diante das memórias das positivas conversas vespertinas com Padre Josenildo, agradeço pelas reflexões instigantes e pelos exemplos de amor cristão obtidos nessa paróquia. Agradeço aos grupos da Pastoral Catequética, do Ministério da Eucaristia e do EJC. Estar inserido em algumas ações da paróquia, mesmo com limitações de tempo, promoveu-me energia e ânimo para trilhar essa jornada acadêmica.

Para deslocar-me até a universidade, percorrendo 680km todas as sextas-feiras, contei com a ajuda dos amigos do curso, com os quais dividia não só o transporte, mas também os desafios e as felicidades que a matemática nos proporcionava. Agradeço à galera do "Wira-móvel", onde Carlos Gonzaga, Wellington Rodrigues e Wirander

Rosa dividiram comigo muitas risadas e aflições no trânsito percorrido no primeiro ano do curso. Já no segundo ano, fui agraciado com a companhia de Tiago Melo, Ruth Micaely e Lucivaldo Pereira, onde compartilhávamos os anseios, os ânimos, os cafés nos acostamentos das estradas e as cargas do trânsito noturno semanal. A estes, expresso o meu singelo obrigado.

Aos meus colegas de sala, manifesto imenso apreço e gratidão, pois construímos uma irmandade, onde a prestatividade e a cordialidade sempre existiram em nossas relações. Posso garantir que a união da nossa turma foi um fator determinante para o nosso progresso e evolução, pois a ajuda mútua ao estudarmos e compartilharmos experiências acadêmicas e profissionais fortaleceram não só a nossa aprendizagem, mas também a nossa vontade de vencer juntos. Expresso a minha imensa gratidão aos amigos André Macedo, Andresson Alquino, Benildo Virgínio, Carlos Gonzaga, Eli Azevedo, Érico Andrade, Erivan Barbosa, Gilmar Veríssimo, Gilvandro Melo, Idalice Santiago, Rafael Macedo e Wirander Rosa. Particularmente, agradeço a Cláudio Teodista por sempre se prontificar para conduzir e organizar estudos coletivos por meio de webconferências noturnas. De modo especial, agradeço, também, a Wellington Rodrigues por acolher-me em sua residência por uma semana, na prévia da qualificação, para estudarmos e revisarmos juntos alguns tópicos importantes para o Exame Nacional de Qualificação.

Sou extremamente grato aos docentes do Departamento de Matemática da UFCG, principalmente aos que estão vinculados ao PROFMAT, estimados Doutores José de Arimateia Fernades, Marcelo Carvalho, Fernando Aires, Jaime Alves, Daniel Cordeiro e Deise Mara. De maneira muito especial sou grato ao coordenador do curso Dr. Romildo Nascimento e ao orientador de pesquisa da minha dissertação Dr. Rodrigo Cohen Mota Nemer. Quero agradecer também à secretária da UAMat, Isabela Souza, por toda prestatividade e cordialidade.

Um brado de gratidão a professora da UEPB, Dra. Maria Isabelle Silva, que foi examinadora externa da pesquisa da dissertação e que muito contribuiu com suas ponderações e colocações.

Expresso minha gratidão à Universidade Federal de Campina Grande que, junto à Unidade Acadêmica de Matemática, oferta com imenso compromisso o curso de Mestrado Profissional em Matemática - PROFMAT.

Por fim, agradeço à Sociedade Brasileira da Matemática - SBM pelo oferecimento deste Curso em Rede Nacional.

"Bonito é melhor que feio.
Explícito é melhor que implícito.
Simples é melhor que complexo.
Complexo é melhor que complicado.
Erros nunca devem passar silenciosamente.
Agora é melhor que nunca."
(The Zen of Python, by Tim Peters)

Resumo

Este trabalho aborda a Linguagem de Programação Python associada à Linguagem Matemática com a finalidade de resolver problemas matemáticos específicos do Ensino Médio. Esta pesquisa fundamenta-se numa habilidade presente na Base Nacional Comum Curricular que direciona para o uso e implementação da Linguagem de Programação como um recurso digital e tecnológico nas aulas de Matemática. Para essa abordagem, promove-se um tutorial básico acerca dos comandos e funções essenciais para o manuseio do Python no contexto da solução dos problemas sugeridos. São propostos quinze problemas com sugestões de codificação e execução de programas que abordam vários tópicos estudados no Ensino Médio. Além disso, é proposta uma Sequência Didática que explora outros problemas de programação sobre Funções Polinomiais definidas por mais de uma sentença.

Palavras-chave: Linguagem de Programação Python. Resolução de Problemas Matemáticos. Ensino Médio.

Abstract

This work addresses the Python Programming Language associated with the Mathematical Language in order to solve specific high school mathematical problems. This research is based on an ability present in the National Common Curricular Base that directs to the use and implementation of the Programming Language as a digital and technological resource in Mathematics classes. For this approach, a basic tutorial is promoted about the essential commands and functions for handling Python in the context of solving the suggested problems. Fifteen problems are proposed with suggestions for coding and running programs that address various topics studied in high school. In addition, a Didactic Sequence is proposed that explores other programming problems about Polynomial Functions defined by more than one sentence.

Keywords: Python Programming Language. Solving Mathematical Problems. High School.

Lista de ilustrações

Lista de tabelas

Lista de códigos-fonte: Linguagem de Programação Python

Sumário

1 Introdução

O presente trabalho tem a intenção de explorar o uso da Linguagem de Programação Python associada à Linguagem Matemática, com a finalidade de discutir a abordagem de conceitos e a elaboração de estratégias para a solução de problemas matemáticos específicos do Ensino Médio. Anseia-se apontar a Linguagem de Programação como um recurso tecnológico digital que pode favorecer a formação educacional das novas gerações, contribuindo com mudanças e formulações de novas estratégias pedagógicas, onde, de modo particular, analisa-se táticas voltadas para a área de Matemática.

As tecnologias digitais transformam constantemente a sociedade, contribuindo com a globalização, favorecendo a forma como as pessoas se comunicam, promovendo uma fluidez nas relações sociais e influenciando o mundo do trabalho. Essa dinâmica impacta a formulação de estratégias pedagógicas para a formação das novas gerações, pois, para atuar em uma sociedade em constante mudança, é necessário ter uma preparação para lidar frequentemente com novidades e inovações que exigem aptidão ao reinventar e aprimorar as habilidades cognitivas e profissionais. Com certeza, as futuras profissões estarão envolvidas com o uso, direto ou indireto, das tecnologias digitais.

Defende-se que a Linguagem de Programação é um poderoso recurso digital tecnológico que pode favorecer o desenvolvimento do letramento matemático dos estudantes e promover aptidões para o futuro mercado de trabalho, por isso esta dissertação pretende orientar professores sobre o seu uso em sala de aula.

A linguagem de programação possibilita a comunicação entre uma máquina e um programador. Esta se trata de uma linguagem formal que permite que um desenvolvedor crie programas a partir de um conjunto de ordens, ações consecutivas, dados e algoritmos, onde toda a sua formulação ocorre por meio de uma série de instruções, símbolos, palavras-chave, regras semânticas e sintáticas. A linguagem de programação faz o controle do comportamento lógico de uma máquina.

Diante disso, argumenta-se que há uma forte ligação entre a Linguagem de Programação e a Linguagem Matemática, o que permite a exploração e a aplicação de vários conceitos que unem estas áreas.

1.1 Objetivos

1.1.1 Objetivo Geral

Orientar o uso da Linguagem de Programação Python como um recurso didático e tecnológico que, em conformidade com as mudanças propostas pela BNCC, seja as-

sociada à linguagem matemática para o trato de problemas específicos do Ensino Médio.

1.1.2 Objetivos Específicos

- Analisar uma habilidade específica da área Matemática e Suas Tecnologias presente na BNCC referente ao uso da linguagem de programação na linguagem matemática;
- Apontar as vantagens pedagógicas de se usar a Linguagem de Programação Python;
- Promover um tutorial sucinto sobre a utilização da linguagem de Programação Python;
- Sugerir problemas que utilizem a linguagem de programação para codificar programas que usem conceitos sobre tópicos específicos de Matemática do Ensino Médio.
- Propor uma Sequência Didática aplicável ao Ensino Médio que utilize a linguagem de programação Python associada à linguagem matemática na abordagem de Funções Polinomiais definidas por mais de uma sentença.

1.2 Organização

Este trabalho está estruturado em 6 capítulos.

O Capítulo 1 se encarrega de introduzir esta dissertação, expondo os seus objetivos, sua organização e estruturação.

O Capítulo 2 trata da fundamentação teórica, onde é analisada uma habilidade específica presente na BNCC que sugere o uso da Linguagem de Programação associada à Linguagem Matemática.

O Capítulo 3, por sua vez, promove um tutorial sobre o manuseio da Linguagem de Programação abordada e sugerida nesse trabalho: Python. Além disso, são indicados Ambientes de Desenvolvimento Integrados que possibilitam a codificação e execução de programas.

Já o Capítulo 4 expõe um total de 15 problemas de programação com aplicações de conceitos matemáticos específicos.

O Capítulo 5 propõe uma sequência didática que orienta sobre o uso da Linguagem de Programação Python na abordagem de Funções Polinomiais definidas por mais de uma sentença.

Por fim, no Capítulo 6 são descritas as conclusões e considerações finais acerca da produção deste trabalho.

Na sequência, são expostas as Referências Bibliográficas e um Apêndice.

2 O uso da Linguagem de Programação na Linguagem Matemática: uma habilidade presente na BNCC

Neste capítulo, pretende-se fundamentar a importância da BNCC na estruturação e no dinamismo da Educação Básica, sobretudo na etapa Ensino Médio, ao discorrer sobre as mudanças no currículo e os Referencias Curriculares que norteiam a elaboração dos Itinerários Formativos para essa etapa.

Além disso, pretende-se analisar competências e habilidades específicas para a área de Matemática e Suas Tecnologias ao destacar a presença, nesses documentos, da sugestão do uso das tecnologias digitais e da computação em sala de aula. Com isso, pretende-se evidenciar uma habilidade específica que sugere a utilização da Linguagem de Programação associada à Linguagem Matemática.

Ademais, destaca-se, na sequência, as vantagens dessa utilização no apoio didático para o aperfeiçoamento do letramento matemático e para a solução de problemas matemáticos específicos, justificando, assim, a importância deste trabalho ao propor a utilização da Linguagem de Programação para tratar de conceitos e problemas matemáticos no Ensino Médio.

Inclusive, menciona-se, também, que a Linguagem de Programação indicada e estudada neste trabalho (Python) é apontada no atual currículo de Pernambuco, após as mudanças curriculares ocorridas no Ensino Médio.

2.1 Sobre a BNCC e os Referenciais Curriculares para a elaboração dos Itinerários Formativos

A Base Nacional Comum Curricular (BNCC), homologada pelo MEC em 2018, define as aprendizagens necessárias para os estudantes brasileiros da Educação Básica. Trata-se de "um documento de caráter normativo que define o conjunto orgânico e progressivo de **aprendizagens essenciais** que todos os alunos devem desenvolver ao longo das etapas e modalidades da Educação Básica, de modo a que tenham assegurados seus direitos de aprendizagem e desenvolvimento",(BRASIL, 2018, p.7).

A BNCC é uma referência nacional para a formulação dos currículos dos sistemas e das redes escolares dos Estados, do Distrito Federal e dos Municípios, inclusive das propostas pedagógicas das instituições escolares, onde sua estrutura se apresenta em

conformidade com a Constituição de 1988, com a Lei de Diretrizes e Bases da Educação Nacional (LDB, Lei nº 9.394/1996) e com o Plano Nacional de Educação (PNE, Lei nº 13.005/2014).

Dessa forma, a BNCC tem o objetivo de servir de referência obrigatória para a formulação dos currículos de todas as redes de ensino e escolas públicas e privadas do país ao apontar quais (e quando) as aprendizagens devem ser desenvolvidas. Ao ser promulgada pelo MEC em 2018, a BNCC apresentou várias modificações estruturais no desenvolvimento curricular das disciplinas, sobretudo do Ensino Médio, onde propõe uma reforma na matriz de referência curricular para os alunos do 1º, 2º e 3º anos dessa etapa escolar. As aprendizagens são expressas sob a forma de competências, as quais representam as capacidades dos estudantes de mobilizar, articular e integrar conhecimentos, habilidades, atitudes e valores. A Lei nº 13.415/2017, que institui as alterações que definem as mudanças no Ensino Médio, estabelece maior integração e flexibilidade curricular e a oferta de Itinerários Formativos.

> As recentes mudanças na LDB, em função da Lei nº 13.415/2017, substituem o modelo único de currículo do Ensino Médio por um modelo diversificado e flexível: O currículo do ensino médio será composto pela **Base Nacional Comum Curricular** e por **Itinerários Formativos**, que deverão ser organizados por meio da oferta de diferentes arranjos curriculares, conforme a relevância para o contexto local e a possibilidade dos sistemas de ensino, a saber: I – linguagens e suas tecnologias; II – matemática e suas tecnologias; III – ciências da natureza e suas tecnologias; IV – ciências humanas e sociais aplicadas; V – formação técnica e profissional.
> (BRASIL, 2018, p.475).

As alterações no Ensino Médio trazem como uma mudança essencial o aumento da carga horária que propõem uma separação entre **Formação Geral Básica** e **Itinerário Formativo**. O currículo que todos os estudantes devem cursar se refere à Formação Geral Básica que, na verdade, se trata do que respalda a BNCC e corresponde a, no máximo, 1800 horas. Já os Itinerários Formativos referem-se ao currículo flexível, ou seja, àquele em que o estudante poderá fazer suas escolhas, para aprofundar e ampliar aprendizagens em uma ou mais Áreas de Conhecimento e/ou na Formação Técnica e Profissional, com carga horária total mínima de 1200 horas. Os Itinerários Formativos se apresentam em três tipos: o itinerário por área, o itinerário integrado e o itinerário que envolve a formação técnica e profissional. O primeiro se refere ao aprofundamento, ampliação e aplicação de uma área específica dentre as quatro grandes áreas do conhecimento (Linguagens e Suas Tecnologias, Matemática e Suas Tecnologias, Ciências da Natureza e Suas Tecnologias e Ciências Humanas e Sociais Aplicadas); o segundo trata do aprofundamento em duas ou mais áreas dentre as quatro grandes áreas do conhecimento; e, por fim, o terceiro orienta sobre a promoção de uma qualificação dos estudantes para o mercado de trabalho.

No tocante à Formação Geral Básica, a BNCC traz um conjunto de **competências gerais** para toda a Educação Básica e outros conjuntos de **competências específicas** para cada área do conhecimento e, para cada uma dessas áreas, há várias **habilidades** a serem desenvolvidas. No que diz respeito aos Itinerários Formativos, é proposto que sejam construídos conforme a relevância para o contexto local dos estudantes e a possibilidade dos sistemas de ensino e, ainda, que sejam executados de forma bastante aplicada, desenvolvendo habilidades que estão colocadas nos Referenciais Curriculares para a sua elaboração. Destaca-se, ainda, que as habilidades dos Itinerários Formativos são distintas das habilidades da BNCC.

Procura-se, logo mais, destacar elementos pertinentes do conjunto das competências gerais, das competências específicas de área e das habilidades. Evidencia-se, pretensiosamente, competências e habilidades que são coniventes com a proposta deste trabalho.

No que se refere às competências gerais da educação básica, destaca-se que a BNCC apresenta um total de 10 competências numeradas que se inter-relacionam e se distribuem na abordagem didática proposta para as três etapas da Educação Básica (Educação Infantil, Ensino Fundamental e Ensino Médio). Dentre elas, destaca-se aqui a competência de número 5:

> Compreender, utilizar e criar tecnologias digitais de informação e comunicação de forma crítica, significativa, reflexiva e ética nas diversas práticas sociais (incluindo as escolares) para se comunicar, acessar e disseminar informações, produzir conhecimentos, resolver problemas e exercer protagonismo e autoria na vida pessoal e coletiva.
> (BRASIL, 2018, p.9)

Ao mencionar essa competência, evidencia-se a pretensão deste trabalho de utilizar a linguagem de programação para criar tecnologias digitais de informação e comunicação para a resolução de problemas matemáticos específicos em sala de aula.

A linguagem de programação pode ajudar os estudantes do Ensino Médio a abrir caminhos e oportunidades. Aprender a programar pode possibilitar aos estudantes o desenvolvimento das habilidades de raciocínio, organização do pensamento e de comunicação ao elaborar estratégias na produção de caminhos para a resolução de problemas, manifestando e desenvolvendo o seu protagonismo. Uma vez que na linguagem de programação utiliza-se de códigos escritos para se comunicar com um computador e instruí-lo a executar determinadas ações e resolver problemas específicos, sobretudo problemas matemáticos. Ressalta-se que a maior razão para se ensinar e aprender matemática é treinar e encorpar a habilidade de resolver problemas. Dessa forma, esse recurso digital e tecnológico pode ser usado tanto em um momento de reforço e solidificação de conhecimentos matemáticos básicos e de exercício de abstração e raciocínio lógico, quanto em um momento de aplicação desses conceitos para resolver problemas. Além disso, o seu uso pode promover uma mudança de ambiente dos conhecimentos matemáticos trabalhados ao sair do papel/caneta e perceber que esse conjunto de

instruções recebidas no contexto de sala de aula pode ser, sim, utilizado em outros lugares.

Para a área de Matemática, a BNCC apresenta 8 competências específicas para o Ensino Fundamental e 5 para o Ensino Médio. Menciona-se, novamente, a preocupação da BNCC com os impactos tecnológicos na transformação da sociedade e a busca para solução de problemas ao evidenciar a competência específica de matemática de número 5 para o Ensino Fundamental, bem como a competência específica de matemática de número 4 para o Ensino Médio:

- Competência 5 – Ensino Fundamental: "Utilizar processos e ferramentas matemáticas, inclusive tecnologias digitais disponíveis, para modelar e resolver problemas cotidianos, sociais e de outras áreas de conhecimento, validando estratégias e resultados." (BRASIL, 2018, p.267)

- Competência 4 – Ensino Médio: "Compreender e utilizar, com flexibilidade e precisão, diferentes registros de representação matemáticos (algébrico, geométrico, estatístico, computacional etc.), na busca de solução e comunicação de resultados de problemas."(BRASIL, 2018, p.531)

Ao analisar essas competências, fica evidente a intenção da BNCC em tratar das tecnologias digitais e explicitar o seu potencial na solução de problemas e na transformação da sociedade. Portanto, é interessante considerar e analisar a integração ao currículo escolar o estudo sobre as linguagens de programação na execução de cálculos representados na linguagem matemática, sobretudo, como propõe uma habilidade específica da BNCC detalhada mais adiante.

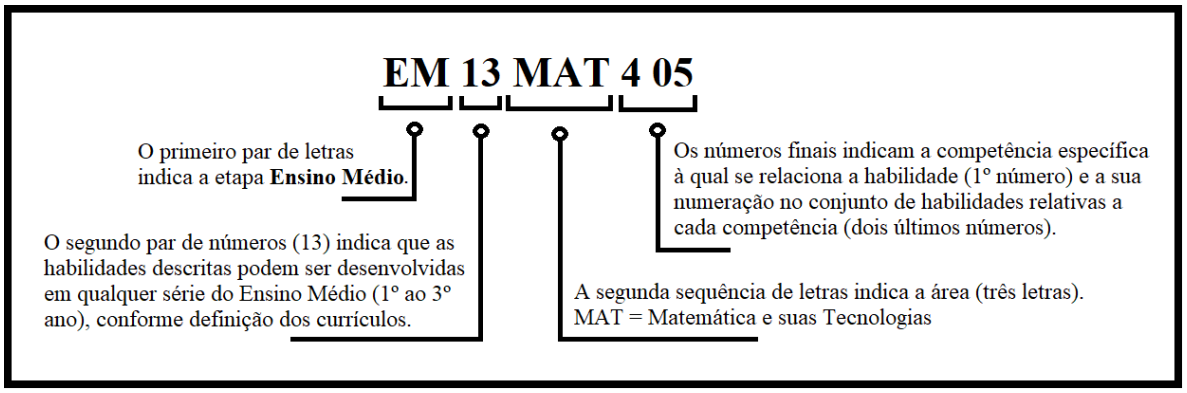
A BNCC estabelece, para cada área do conhecimento, competências específicas de área, que devem ser desenvolvidas ao longo das etapas da Educação Básica de forma particular a cada etapa. No Ensino Médio, por exemplo, o desenvolvimento dessas competências específicas de área deve ser promovido tanto no âmbito da BNCC como nos Itinerários Formativos das diferentes áreas. Essas intenções explicitam como as competências gerais da Educação Básica se expressam nas áreas do conhecimento e como estão articuladas com as competências específicas de área para o Ensino Fundamental e o Ensino Médio.

Para assegurar o desenvolvimento das competências específicas de área, a cada uma delas é relacionado um conjunto de habilidades, que representa as aprendizagens essenciais a serem garantidas no âmbito da BNCC a todos os estudantes. A organização dessas intenções são expressas no documento separadas por área e seguem uma mesma estrutura: definição de competências específicas de área e habilidades que lhes correspondem, onde cada competência é numerada em ordem crescente e cada habilidade é identificada por um código alfanumérico.

Dito isso, destaca-se aqui uma habilidade em particular, da área de Matemática, que advém da competência específica de número 4 para o Ensino Médio e, posteriormente, é relacionada pelo documento da BNCC ao eixo Números e Álgebra:

Matemática e Suas Tecnologias no Ensino Médio: Competências e Habilidades. Competência Específica 4.
Habilidade: "**(EM13MAT405) Utilizar conceitos iniciais de uma linguagem de programação na implementação de algoritmos escritos em linguagem corrente e/ou matemática.**" (BRASIL, 2018, p.543)

Figura 1 – Habilidade EM13MAT405.



Fonte: Elaborada pelo autor.

Conforme a Figura 1, o código alfanumérico EM13MAT405, refere-se à quinta habilidade proposta na área de Matemática e Suas Tecnologias relacionada à competência específica 4, que pode ser desenvolvida em qualquer série do Ensino Médio, conforme definições curriculares.

Tal habilidade propõe o uso em sala de aula de uma Linguagem e Programação alinhada à Linguagem Matemática, com o intuito de utilizar algorítimos para representar situações matemáticas e resolver problemas. Com isso, justifica-se a motivação desse trabalho de discutir formas de trabalhar conceitos e problemas matemáticos específicos no Ensino Médio ao se utilizar da linguagem de programação.

Portanto, pretende-se discutir, neste trabalho, caminhos que facilitem a utilização da linguagem de programação em sala de aula, apontando instruções, fornecendo sugestões e estabelecendo uma associação com a linguagem matemática.

O documento da BNCC assume de maneira explícita o seu compromisso com a educação integral, o que direciona as escolas e redes de ensino a estabelecerem, intencionalmente, processos educativos que promovam o desenvolvimento intelectual, físico, social, emocional e cultural dos estudantes e também com os desafios da sociedade contemporânea. É notório que a BNCC aponta o que os estudantes devem aprender

e, sobretudo, o que devem fazer com o que aprenderam, levando em consideração que a educação tem um compromisso com a formação e o desenvolvimento humano global, em suas dimensões intelectual, física, afetiva, social, ética e moral.

Dessa forma, a BNCC norteia a construção de currículos que levem em consideração o local e as possibilidades dos sistemas de ensino onde esses alunos estão inseridos, dando autonomia aos sistemas, às redes de ensino e às instituições escolares para tomar decisões que resultam de um processo de envolvimento e participação das famílias e da comunidade. A BNCC destaca, ainda, que essas decisões nas elaborações dos currículos refiram-se a algumas ações específicas, sendo essas apresentadas pela própria BNCC e dispostas em oito tópicos, dos quais destaca-se aqui a quinta ação (quinto tópico), que se refere à ação de "selecionar, produzir, aplicar e avaliar recursos didáticos e tecnológicos para apoiar o processo de ensinar e aprender", (BRASIL, 2018, p.17).

Com base nisso, enfatiza-se que a intenção deste trabalho corrobora com essa ideia, onde anseia-se demonstrar as vantagens de se incorporar a linguagem de programação no ensino e aprendizagem de matemática, onde exibe-se a incorporação do Python no processo de ensino e aprendizagem de matemática, expondo a aplicação e utilização desse recurso didático e tecnológico na resolução de problemas matemáticos específicos. Logo mais adiante, nos Capítulos 4 e 5, trata-se dessas questões com mais ênfase.

A preocupação com os impactos tecnológicos na transformação da sociedade está expressa na BNCC e se apresenta já nas competências gerais para a Educação Básica. Ainda assim, no que se refere **às tecnologias digitais e à computação**, a Base também afirma que

> Essa constante transformação ocasionada pelas tecnologias, bem como sua repercussão na forma como as pessoas se comunicam, impacta diretamente no funcionamento da sociedade e, portanto, no mundo do trabalho. A dinamicidade e a fluidez das relações sociais – seja em nível interpessoal, seja em nível planetário – têm impactos na formação das novas gerações. É preciso garantir aos jovens aprendizagens para atuar em uma sociedade em constante mudança, prepará-los para profissões que ainda não existem, para usar tecnologias que ainda não foram inventadas e para resolver problemas que ainda não conhecemos. Certamente, grande parte das futuras profissões envolverá, direta ou indiretamente, computação e tecnologias digitais. (BRASIL, 2018, p.473)

Sendo assim, torna-se bastante pertinente a intenção deste trabalho de evidenciar o uso da linguagem de programação associada à linguagem matemática, uma vez que a contemporaneidade é fortemente marcada pelo desenvolvimento tecnológico e o cotidiano está constantemente sendo movido pelas tecnologias digitais, tornando o mundo produtivo.

O uso da linguagem de programação pode ser aplicado no tratamento de dados científicos que se relacionam fortemente com a matemática. Considera-se que aplicar

a linguagem de programação como recurso didático tecnológico para apoiar o processo de ensinar e aprender matemática pode favorecer e ampliar a visão de mundo dos estudantes, que, em sua maioria, estão imersos em um mundo digital. O ato de programar promove nos discentes a habilidade de solucionar problemas matemáticos por meio da programação, desde os simples e diretos até os mais complexos e elaborados.

Ademais, o documento que expõe os Referenciais Curriculares para a elaboração de Itinerários Formativos expressa que a mudança no Ensino Médio "busca assegurar o desenvolvimento de conhecimentos, habilidades, atitudes e valores capazes de formar as novas gerações para lidar com desafios pessoais, profissionais, sociais, culturais e ambientais do presente e do futuro, considerando a intensidade e velocidade das transformações que marcam as sociedades na contemporaneidade."(BRASIL, 2020, p.4). Estes Referenciais Curriculares articulam que os Itinerários Formativos organizam-se a partir de quatro eixos estruturantes: Investigação Científica, Mediação e Intervenção Sociocultural, Processos Criativos e Empreendedorismo. "Tais eixos estruturantes visam integrar e integralizar os diferentes arranjos de Itinerários Formativos, bem como criar oportunidades para que os estudantes vivenciem experiências educativas profundamente associadas à realidade contemporânea, que promovam a sua formação pessoal, profissional e cidadã."(BRASIL, 2020, p.7).

Dentre esses eixos estruturantes, enfatiza-se aqui que o eixo Processos Criativos tem como foco pedagógico a participação dos estudantes na realização de projetos criativos, por meio da utilização e integração de diferentes linguagens, manifestações sensoriais, vivências artísticas, culturais, midiáticas e científicas aplicadas. Além disso, os Referenciais Curriculares para a elaboração de Itinerários Formativos apresentam várias habilidades específicas associadas aos eixos estruturantes, donde destaca-se, aqui, duas habilidades em particular:

Habilidades Específicas dos Itinerários Formativos associadas ao Eixo Estruturante - Processo Criativo.

- Área de Ciências da Natureza e suas Tecnologias.
  Habilidade: "(EMIFCNT06) Propor e testar soluções éticas, estéticas, criativas e inovadoras para problemas reais, considerando a aplicação de design de soluções e o uso de tecnologias digitais, programação e/ou pensamento computacional que apoiem a construção de protótipos, dispositivos e/ou equipamentos, com o intuito de melhorar a qualidade de vida e/ou os processos produtivos." (BRASIL, 2020, p.12).

- Área de Matemática e suas Tecnologias
  Habilidade: "(EMIFMAT04) Reconhecer produtos e/ou processos criativos por

> meio de fruição, vivências e reflexão crítica na produção do conhecimento matemático e sua aplicação no desenvolvimento de processos tecnológicos diversos." (BRASIL, 2020, p.12).

O código alfanumérico EMIFCNT06 refere-se à sexta habilidade proposta para a área de Ciências da Natureza e suas Tecnologias relacionada aos Itinerários Formativos a serem desenvolvidos no Ensino Médio. De maneira similar, o código alfanumérico EMIFMAT04 refere-se à quarta habilidade proposta para a área de Matemática e suas Tecnologias, também relacionada aos Itinerários Formativos a serem desenvolvidos no Ensino Médio.

A partir dessas habilidades, percebe-se que, assim como a BNCC, os Referenciais Curriculares para a elaboração de Itinerários Formativos expressam a sua intenção de utilizar as tecnologias digitais e a computação para tratar de problemas que passeiam por várias áreas do conhecimento, com o intuito de que os alunos possam solucioná-los de forma criativa. No que se refere à área de Matemática, o uso das tecnologias digitais, como a linguagem de programação, pode auxiliar no processo criativo dos estudantes de levantar e testar hipóteses acerca de variáveis que interferem na explicação de um conceito matemático ou na resolução de um problema. A linguagem de programação é uma ferramenta poderosa na elaboração de modelos com a linguagem matemática para o trato de dados científicos e pode promover nos alunos a habilidade de resolver problemas de forma criativa.

2.2 A Linguagem de Programação e a Linguagem Matemática

Ao estudar e aprender matemática, os alunos se deparam com uma vastidão de conceitos que envolvem símbolos, onde se faz necessário ter uma aptidão na leitura e compreensão desses, estabelecendo, assim, uma relação entre as simbologias e a linguagem natural. Dessa forma, o desenvolvimento do conhecimento matemático é, intrinsecamente, agregado ao domínio da interpretação dessa linguagem matemática.

De acordo com (LORENSATTI, 2009, p.90)

> A linguagem matemática pode ser definida como um sistema simbólico, com símbolos próprios que se relacionam segundo determinadas regras. Esse conjunto de símbolos e regras deve ser entendido pela comunidade que o utiliza. A apropriação desse conhecimento é indissociável do processo de construção do conhecimento matemático.

De maneira similar, ao estudar, aprender e desenvolver programação, o programador se depara com uma linguagem computacional própria ao escrever códigos que estruturam seus elementos e suas regras para o desenvolvimento de um *software*. Pode-se dizer que programar é o ato de construir um programa com um conjunto de instruções

precisas, dizendo a um computador o que fazer; dessa forma, pode-se dizer, também, que uma linguagem de programação é uma linguagem construída artificialmente, usada para instruir computadores (HAVERBEKE, 2018). A Linguagem de Programação é, portanto, uma linguagem escrita e formal que especifica um conjunto de instruções e regras usadas para gerar programas e *softwares*.

Reforçando esse conceito, (CARNEIRO et al., 2022, p.6) define que

> Entende-se por linguagem de programação ao conjunto de regras sintáticas definidas pelo usuário e que devem ser executadas pelo computador para a realização de tarefas, por exemplo, a solução numérica de problemas matemáticos ou aquisição de medidas de experimentos em laboratório.

Dessa maneira, como a programação está diretamente ligada com uma sequência lógica de símbolos e regras que tem o intuito de executar uma ação de forma objetiva, pode-se então perceber sua correlação com a matemática. Pois, o ato de programar se baseia na seleção e execução de um conjunto de ações que são formuladas previamente para atingir um determinado objetivo, o que se alinha com o trato da linguagem matemática de usar conceitos lógicos para selecionar caminhos e tomar decisões para resolução de problemas. Ao discorrer sobre a origem das Linguagens de Programação, (BERTOLINI et al., 2019, p.13) assegura que "as linguagens de programação surgiram da evolução da lógica matemática, no qual abstrai conceitos complexos da matemática e podia ser utilizada para resolver problemas específicos."

Ao discorrer sobre recursos computacionais no ensino da matemática, (GIRALDO; CAETANO; MATTOS, 2012, p.127, grifo do autor) afirmam que

> **A própria aprendizagem de uma sintaxe de programação já constitui, por si só, exercícios de simbolização matemática de natureza diferente daqueles que fazemos com papel e lápis**. De fato, quando aprendemos certa simbologia matemática, devemos nos familiarizar com a consistência lógica de suas regras para expressar ideias e procedimentos matemáticos adequadamente.

Além disso, tanto na programação quanto na resolução de questões matemáticas, o enfrentamento de erros e suas devidas correções podem ser construtivas e significativas para a efetivação do aprendizado e da construção de uma habilidade. Sobre o enfrentamento de erros, (GIRALDO; CAETANO; MATTOS, 2012, p.127) ainda reforçam que

> Quando aprendemos as regras sintáticas de uma linguagem de programação computacional, as eventuais inconsistências lógicas cometidas já são indicadas pelo próprio software, na forma de mensagens de erro. Isto é, de certa forma, o software responde as tentativas (corretas ou incorretas) do usuário para expressar procedimentos matemáticos.

Ademais, corroborando com (GIRALDO; CAETANO; MATTOS, 2012), no que diz respeito à interpretação de resultados produzidos por ferramentas computacionais

simbólicas, mesmo nos sistemas de computação e programação poderosos, a análise dos erros e suas devidas correções não dispensa ou substitui o conhecimento dos conceitos matemáticos envolvidos.

Acredita-se que a junção, em sala de aula, da linguagem de programação com a linguagem matemática pode contribuir consideravelmente para o desenvolvimento do letramento matemático dos alunos, pois o domínio da habilidade de produzir uma codificação para a execução de um programa que resolva problemas matemáticos específicos só é efetivo quando se domina a linguagem matemática necessária para se resolver tal problema. Sendo assim, saber codificar, com o intuito de gerar um programa que resolva um problema matemático, é também saber os conceitos matemáticos que conduzem o caminho para a sua solução. Isso é tratado com mais detalhes na Subseção 3.2.4.1 do Capítulo 3, onde se promove um tutorial sobre a utilização de operações aritméticas ao definir expressões matemáticas por meio da linguagem de programação Python. Inclusive, isso também fica evidente no Capitulo 4, onde se expõe problemas e aplicações matemáticas moldadas por meio da linguagem de programação Python.

A Matriz de Referência em Avaliação Matemática do PISA (Programa Internacional de Avaliação de Estudantes) (BRASIL, 2012), apresenta a seguinte definição para o Letramento Matemático:

> Letramento Matemático é a capacidade individual de formular, empregar e interpretar a matemática em uma variedade de contextos. Isso inclui raciocinar matematicamente e utilizar conceitos, procedimentos, fatos e ferramentas matemáticas para descrever, explicar e predizer fenômenos. Isso auxilia os indivíduos a reconhecer o papel que a matemática exerce no mundo e para que cidadãos construtivos, engajados e reflexivos possam fazer julgamentos bem fundamentados e tomar as decisões necessárias.

Consolidando o argumento de que o uso da linguagem de programação associada à linguagem matemática aprimora o letramento matemático dos estudantes e contribui para o desenvolvimento de várias competências, (RESNICK, 2013) menciona que

> No processo de aprender a programar, as pessoas aprendem muitas outras coisas. Eles não estão apenas aprendendo a codificar, eles estão codificando para aprender. Além de aprender ideias matemáticas e computacionais (como variáveis e condicionais), eles também estão aprendendo estratégias para resolver problemas, projetar projetos e comunicar ideias. Essas habilidades são úteis não apenas para cientistas da computação, mas para todos, independentemente de idade, interesses ou ocupação.

Isto posto, argumenta-se que aprender a programar é aprender a resolver problemas e, assim como na matemática, o aprendizado e a habilidade se consolidam com a prática. Além disso, conhecer várias linguagens de programação diferentes e trocar de uma linguagem para outra consolidam a aptidão de resolver problemas.

> Não pense que saber programar é decorar todos aqueles comandos, parâmetros e nomes estranhos. Programar é saber utilizar uma lin-

> guagem de programação para resolver problemas, ou seja, saber expressar uma solução usando uma linguagem de programação.
> (MENEZES, 2010, p.17)

Sendo assim, reforça-se que a utilização da Linguagem de Programação na Linguagem Matemática é potencialmente rica para o ensino e aprendizagem em matemática por contribuir com o desenvolvimento de competências fundamentais para o letramento matemático (raciocínio, representação, comunicação e argumentação), bem como para o desenvolvimento do pensamento computacional, sendo que, tudo isso, favorece o desenvolvimento da capacidade de ser um resolvedor de problemas.

2.3 A Linguagem de Programação Python presente no Currículo de Pernambuco

Uma vez que, neste trabalho, indica-se o Python dentre várias Linguagens de Programação existentes, então, nesta seção, pretende-se enfatizar que a Linguagem de Programação Python é apontada como uma unidade curricular no atual currículo de Pernambuco implementado no ano de 2021. Aqui, noticia o leitor que o professor que produz este trabalho atua no referido estado, na rede estadual de ensino, numa Escola de Referência em Ensino Médio, cuja carga horária é de tempo integral.

Como exposto anteriormente neste capítulo, a intenção de utilizar as tecnologias digitais e a computação para tratar de problemas de várias áreas do conhecimento é apontada tanto na BNCC, que orienta para a Formação Geral Básica, quanto nos Referencias Curriculares para a elaboração dos Itinerários Formativos. Com base nisso, afirma-se que a Secretaria de Educação de Pernambuco, ao definir a sua nova composição curricular, sugere uma trilha voltada para as tecnologias digitais ao orientar sobre os referenciais normativos para organização dos seus Itinerários Formativos, especificamente, para um itinerário integrado que incorpora Matemática e Ciências da Natureza.

É imposto que os Itinerários Formativos devem ser organizados por meio da oferta de diversos arranjos curriculares, conforme a relevância para o contexto local e as possibilidades dos sistemas de ensino. Sendo assim, o currículo de Pernambuco estabelece várias trilhas associadas aos itinerários formativos integrados, o que se alinha com essa imposição. Isso promove às várias escolas do estado a possibilidade e escolher trilhas apropriadas ao seu contexto local.

Integrando as áreas do conhecimento de Matemática e Ciências da Natureza, o currículo de Pernambuco sugere uma trilha denominada de Tecnologias Digitais. Tal trilha abarca as unidades temáticas nomeadas por Tecnologias e Linguagens, Criatividade na Era Digital, Intervenções Tecnológicas e Produções Tecnológicas. Nesta

última, a trilha exibe que Programação com Python é uma unidade curricular ligada ao já mencionado eixo estruturante Processos Criativos e que pode ser trabalhada no 1º semestre do 3º ano do Ensino Médio, conforme exibe a Figura 2 a seguir.

Figura 2 – Linguagem de Programação Python presente no Currículo de Pernambuco

Trilha: Tecnologias Digitais (MATEMÁTICA E NATUREZA)

TRILHA: TECNOLOGIAS DIGITAIS				
Perfil do egresso: Reconhecer-se atuando como um agente autônomo, criativo e reflexivo em torno das questões sociais, econômicas e tecnológicas, articulando saberes de automação, programação e sistema dinâmico no contexto dos conhecimentos matemáticos.				
Cursos superiores relacionados				
Matemática, Física, Engenharias, Ciências da Computação, Expressão Gráfica, entre outros relacionados				
3º Ano – 1º Semestre				
UNIDADE TEMÁTICA: PRODUÇÕES TECNOLÓGICAS				
Objetivo do Semestre: Levantar e testar hipóteses por meio de variáveis associadas a resolução de situações-problema, selecionando, sistematizando e mobilizando as diferentes produções tecnológicas em torno das mais variadas discussões sociais, inclusive dos conhecimentos matemáticos envolvidos.				
PERÍODO	**UNIDADE CURRICULAR**	**EIXO ESTRUTURANTE**	**HABILIDADE DA UNIDADE CURRICULAR**	**EMENTA**
3º Ano 1º Semestre	Programação com Python (40h) Formação docente: Matemática, Física	Processos Criativos	(EMIFMAT06PE) Propor e testar soluções éticas, estéticas, criativas e inovadoras para problemas reais, considerando a lógica de programação Python aplicada a partir dos conhecimentos matemáticos associados ao domínio de operações e relações matemáticas simbólicas e formais.	Proposição, testagem e resolução de problemas matemáticos utilizando algoritmo e a linguagem de programação Python. Seleção e mobilização dos conhecimentos da lógica de programação aplicada a sintaxes, variáveis, estruturas de repetição (loops), estruturas condicionais, noções de armazenamento e tratamento de dados.
		Mediação e Intervenção Sociocultural	(EMIFMAT09PE) Propor e testar estratégias de mediação e intervenção para resolver problemas de natureza sociocultural e de natureza ambiental relacionados à lógica de programação Python aplicada a partir de conhecimentos matemáticos.	

Fonte: Currículo de Pernambuco, (PERNAMBUCO, 2021, p. 582).

Caso uma escola escolha esta trilha, então será oportunizado aos seus alunos a possibilidade de discutir estratégias ligadas à produção tecnológica para resolver problemas, sobretudo utilizar a programação com Python para esse fim. Esta dissertação pode ser um excelente instrumento que pode auxiliar os professores de matemática na condução dessa discussão. Este trabalho também pode orientar professores de matemática das escolas pernambucanas que, eventualmente, não elejam a trilha apresentada na Figura 2, pois nada impede que o professor implemente em suas aulas corriqueiras as sugestões didáticas apresentadas neste trabalho.

Na verdade, esta dissertação pode nortear professores de matemática de modo geral, de todos os estados brasileiros. Diante, das recentes mudanças curriculares impostas, várias escolas se deparam com o desafio nas adaptações dessas modificações, bem como nas orientações aos professores para o uso das tecnologias digitais e da computação. Sendo assim, as orientações, instruções, aplicações e propostas didáticas apresentadas nesse trabalho podem enriquecer a formação e a atuação dos professores, como também favorecer a apropriação do uso das tecnologias digitais impostas nas mudanças curriculares.

Por essa razão, este trabalho sugere a Linguagem de Programação Python como um recurso didático tecnológico, onde apresenta-se um tutorial a cerca do seu manuseio no Capítulo 3, exibe-se problemas de programação aplicados à matemática no Capítulo 4 e apresenta-se uma proposta didática no Capítulo 5. Espera-se que tudo isso possa ser de grande valia para os professores neste momento oportuno.

3 A Linguagem de Programação Python

Almeja-se, neste capítulo, apontar as vantagens da utilização da Linguagem de Programação Python, mencionando ser conveniente o seu uso no ambiente educacional, justificando, assim, a sua escolha dentre outras linguagens de programação. Além disso, sugere-se a utilização de Ambientes de Desenvolvimento Integrado (IDEs), que facilitam a codificação e execução de programas e, ainda, favorecem o trato pedagógico na aprendizagem da linguagem de programação Python. Na sequência, apresenta-se um singelo tutorial sobre comandos primordiais para o manuseio desta linguagem de programação, onde exibe-se exemplos práticos de codificações seguidas de sua devida execução.

3.1 Sobre o Python: vantagens, instalação e programação

Dentre as várias Linguagens de Programação existentes, como *Java, Javascript, C++, Ruby, PHP, Python, C#* e etc, enaltece-se, neste trabalho, a **Linguagem de Programação Python** por ser uma das linguagens mais simples e objetivas ao ser codificada. Além disso, apresenta fácil compreensão para análise de dados e, inclusive, é compatível com vários sistemas operacionais, como Windows, Linux e MacOS.

Python é uma linguagem de programação simples e de alto nível que possui um modelo de desenvolvimento aberto e comunitário. Foi criada em 1991 pelo matemático e programador de computadores holandês Guido Van Rossum, estando ao alcance de qualquer usuário para manuseá-la em vários ambientes e com várias finalidades, inclusive no meio educacional. É administrada pela organização sem fins lucrativos Python Software Foundation que, com a participação de diversos colaboradores, torna o seu uso gratuito, livre e compatível com inúmeras arquiteturas computacionais.

Justificando a escolha do Python, afirma-se que esta linguagem de programação tem um grande potencial pedagógico, justamente pela sua simplicidade, que facilita o ensino e compreensão dos conceitos de codificação da ciência da computação. Essa característica contribui com o uso das tecnologias digitais e com as mudanças na dinâmica da Educação Básica mencionadas no capítulo anterior (Cap. 2). O Python pode simplificar o trabalho de aprendizado e fornecer grande poder de programação.

> A linguagem de programação Python é muito interessante como primeira linguagem de programação devido à sua simplicidade e clareza. Embora simples, é também uma linguagem poderosa, podendo ser usada para administrar sistemas e desenvolver grandes projetos. É uma linguagem clara e objetiva, pois vai direto ao ponto, sem rodeios. (MENEZES, 2010, p.21)

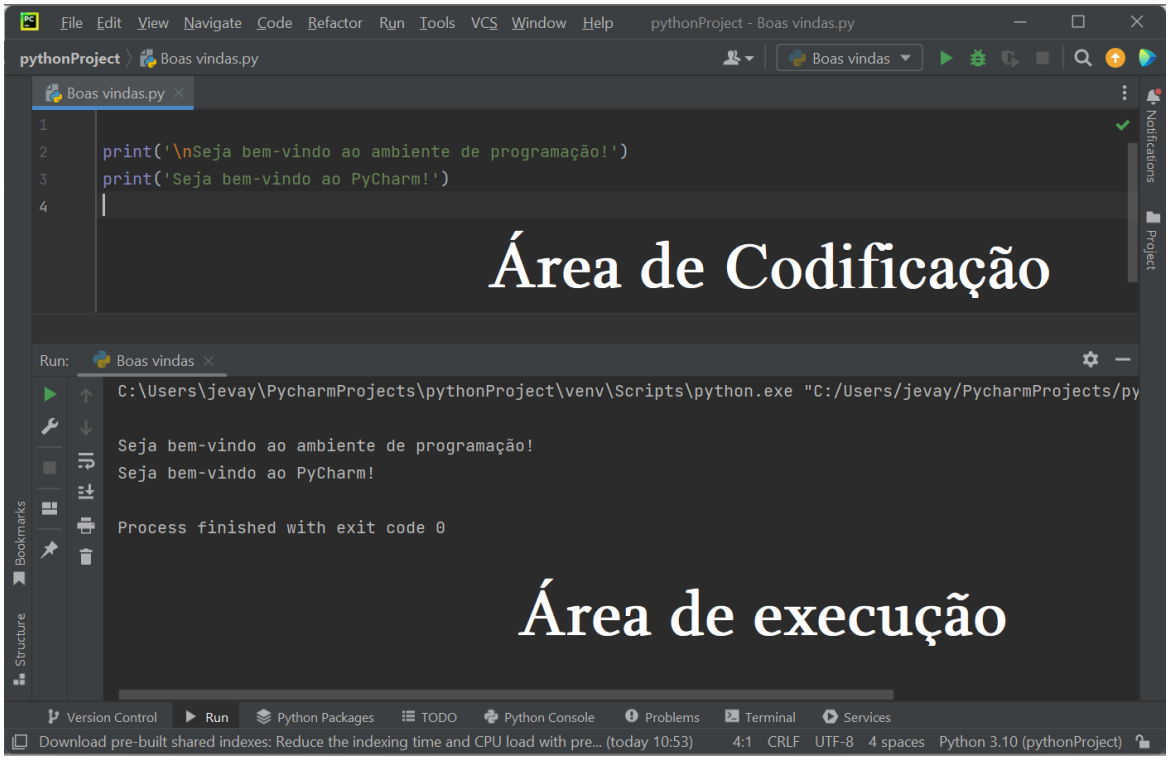
Considera-se a Linguagem de Programação Python vantajosa, pois é fácil, intuitiva, organizada e multiplataforma (funciona em vários sistemas operacionais). Além disso, é uma linguagem de propósito geral, ou seja, pode ser utilizada com vários objetivos e finalidades. Inclusive, já apresenta várias funcionalidades sem a necessidade de instalações, o que a comunidade de programação denomina por *"batteries included"*, ou seja, baterias incluídas, fazendo referência a um produto que pode ser usado prontamente. Além do mais, pode-se fazer uso de bibliotecas, que são instaladas posteriormente, para utilizar muitas funções já existentes escrevendo poucas linhas de código, bem como adicionar funcionalidades mais elaboradas e intencionais. "Python se destaca por seu fácil processo de aprendizagem, sintaxe simples e rica biblioteca que auxilia o estudante no decorrer do desenvolvimento de cada código computacional." (MARCONDES, 2018)

Os sistemas operacionais Linux e MacOS já possuem instalado o interpretador Python, que faz a interlocução entre o programador e o computador. O interpretador faz a leitura de um programa de alto nível (chamado de código-fonte) e executa-o, ou seja, realiza as ações da codificação do programa passo a passo. No Windows, no entanto, é necessário realizar sua instalação, que pode ser feita de forma rápida, simples e segura por meio do endereço eletrônico *https://www.python.org*. Além da instalação, através deste endereço pode-se realizar atualizações para novas versões que contemplam todos os sistemas operacionais aqui mencionados.

Nesta instalação, além do interpretador, obtém-se um terminal denominado por IDLE - Integrated Development and Learning Environment (Desenvolvimento Integrado e Ambiente de Aprendizagem), que se trata de um espaço facilitador para codificação e execução da programação. No entanto, sugere-se, neste trabalho, a utilização de plataformas mais completas e funcionais que, assim como o IDLE, tratam-se de um Ambiente de Desenvolvimento Integrado, também chamado de IDE (Integrated Development Environment). O IDE é um software que facilita a criação de um código-fonte de forma rápida e eficiente, possui muitas funcionalidades que auxiliam no seu desenvolvimento, além de possuir a capacidade de visualizar de forma organizada a produção e execução do código.

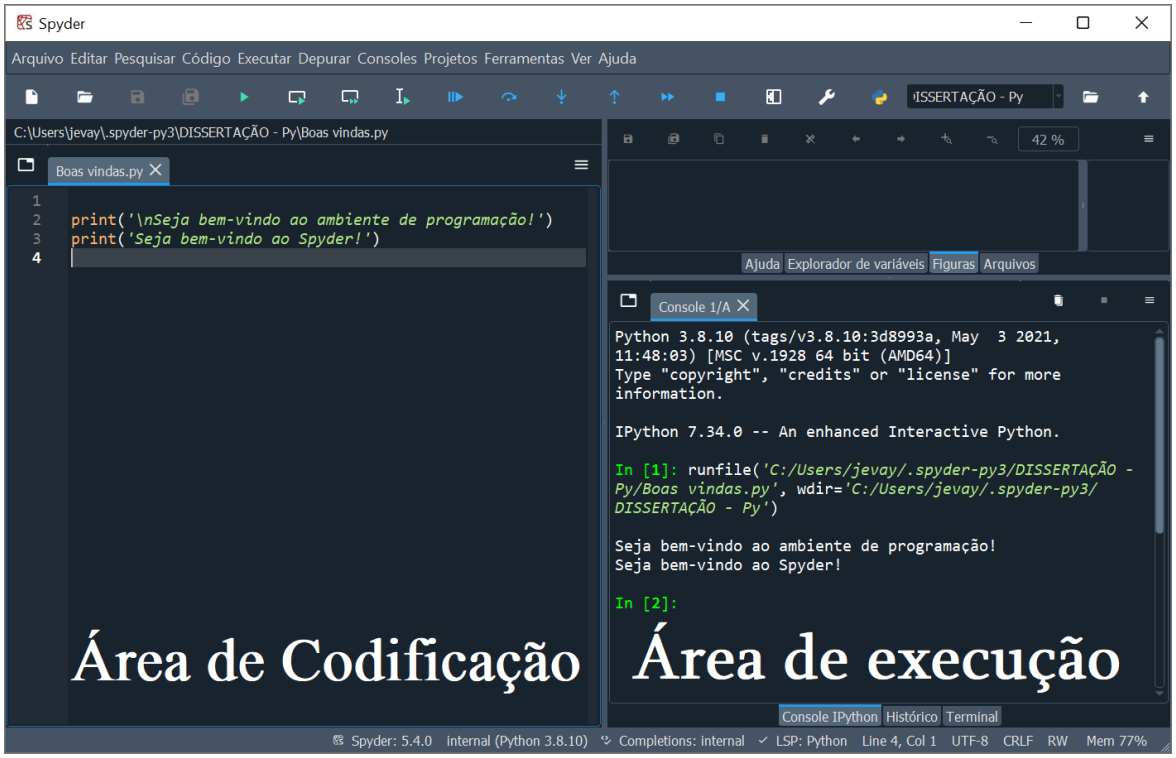
Indica-se, aqui, dois ambientes de desenvolvimento integrado: o PyCharm e o Spyder. O primeiro, cuja tela principal está ilustrada na Figura 3, trata-se de um IDE desenvolvido pela empresa JetBrains de origem tcheca; a sua utilização é simples e pode ser instalado gratuitamente e sem muita complexidade pelo endereço eletrônico *https://www.jetbrains.com/pycharm*. O segundo, por sua vez, expresso na Figura 4, é um IDE poderoso, simples de utilizar, de código aberto e pode ser obtido e instalado de forma gratuita pelo endereço eletrônico *https://www.spyder-ide.org*.

Figura 3 – IDE - PyCharm



Fonte: Elaborada pelo autor.

Figura 4 – IDE - Spyder



Fonte: Elaborada pelo autor.

Figura 5 – Ícones dos IDEs PyCharm e Spyder

Fonte: Elaborada pelo autor.

Ambos os IDEs sugeridos facilitam a visualização simultânea e separada da codificação e da execução. Um compilador lê o código fonte do programa e o traduz completamente antes de executá-lo. O código-fonte é o conjunto de comandos escritos numa linguagem de programação e funciona como uma lista de tarefas que o desenvolvedor manda o programa fazer, onde cada linha escrita define o que e como deve ser feito.

Nas Figuras 3 e 4, destaca-se os espaços utilizados para a edição do código-fonte, bem como para a sua execução. No Spyder, a área de execução também é denominada de Console e, diferentemente do PyCharm, ela pode agregar a execução de vários programas em um único espaço, o que pode deixá-la "poluída" devido ao excesso de execuções; alternativamente, o programador pode abrir uma nova área de console realizando uma rápida reinicialização.

Destaca-se, ainda, que os dois ambientes de desenvolvimento mencionados permitem salvar os projetos codificados para uma possível execução futura. Isso torna todo o trabalho do programador mais organizado e eficiente, pois novos projetos podem ser criados à medida em que se analisa e visita projetos antigos, editando-os e executando-os prontamente.

Menciona-se, também, que nesses IDEs, os códigos da programação são dispostos em linhas numeradas, o que facilita a organização e visualização da codificação. Ademais, eventuais erros são mais fáceis de encontrar e tratar, pois, ao executar o programa, o IDE aponta o erro e o número da linha em que o erro ocorreu, direcionando a sua correção.

Conforme as Figuras 3 e 4, a execução da mensagem "`Seja bem-vindo ao ambiente de programação!`" é realizada por meio da compilação de um comando representado por `print('')` que escreve/imprime a mensagem descrita. Comandos como esse serão explicados na próxima seção (3.2), bem como outros necessários para a codificação dos programas apresentados no Capítulo 4.

3.2 Introdução à Programação com Python

Nesta seção pretende-se apresentar um tutorial sucinto acerca dos comandos primordiais e necessários para se programar com Python. Serão expostos conceitos e exemplos de códigos que expressam mensagens, variáveis, comentários, operações, comparações, condições, importações de bibliotecas, entre outras aplicabilidades.

De forma pretensiosa, apresenta-se, aqui, instruções relacionadas aos códigos que serão utilizados na resolução dos problemas dos capítulos 4 e 5. Vale ressaltar que existe uma imensa quantidade de comandos e funções no Python, porém o objetivo do texto é apenas tratar daqueles que serão utilizados neste trabalho.

A linguagem de programação fornece um conjunto de funções, cada uma com uma tarefa simples e específica. Por exemplo, `print()` é uma função do Python utilizada para imprimir alguma mensagem na tela; para isso, a mensagem deve estar delimitada entre aspas simples (‘ ’) ou duplas (“ ”), conforme exibe as Figuras 3 e 4. Dessa forma, pode-se codificar `print("Olá! Bem-vindo à linguagem de programação Python!")` e o resultado obtido por meio da compilação desta codificação é `Olá! Bem-vindo à linguagem de programação Python!`.

3.2.1 Variáveis

O uso de variáveis, como um número ou texto, é frequente em um programa. Por exemplo, as variáveis 2, 3.5 e ‘Olá!’ são de tipos diferentes, sendo respectivamente, um número inteiro, um decimal e um texto. Diferentemente de outras linguagens de programação, no Python muitas vezes não é preciso declarar o tipo de variável ao se programar, pois isso já fica subentendido. No entanto, para a manipulação de alguns dados específicos, é necessário fazer a declaração do tipo de variável.

As variáveis são utilizadas para atribuir e manipular dados que ficam armazenados na memória da programação. Sobre os tipos e classes de variáveis mencionadas, o tipo `int` (inteiro) armazena números inteiros; o tipo `float` (flutuante) armazena números decimais; e o tipo `str` (string) armazena um conjunto de caracteres, como um texto. Este último deve ser representado utilizando aspas (simples ou duplas). Ademais, cada variável pode armazenar apenas um tipo de dado por vez.

Para declarar variáveis, é necessário definir um nome e atribuir um valor/significado para ela, usando o sinal de “=” conforme o exemplo expresso no Código-fonte 3.1 abaixo.

```
1 #EXEMPLOS DE VARIÁVEIS
2 a = 2              # atribui o valor 2 para a variável a
3 b = 3.5            # atribui o valor 3.5 para a variável b
4 mensagem = 'Olá!'  # atribui o significado Olá! para a variável mensagem
```

```
5  variável = '7.5'   # atribui o significado 7.5 para a variável variável
6  var1 = int(2)      # armazena o valor 2 como inteiro
7  var2 = float(2)    # armazena o valor 2 como flutuante (decimal)
8  var3 = str(2)      # armazena o valor 2 como string
9
10 #IMPRIMINDO E MANIPULANDO AS VARIÁVEIS
11 print(a)           # imprime a variável a execução do programa.
12 print(b)           # imprime a variável b execução do programa.
13 print(a+b)     # imprime a soma dos números armazenados pelas variáveis a e b.
14 print(mensagem)    # imprime o significado atribuído a variável mensagem.
15 print(variável)    # imprime o significado atribuído a variável variável.
16 print(mensagem + variável) #imprime a junção dos significados das variáveis
       mensagem e variável.
17 print(var1)        # imprime o número inteiro 2 atribuído.
18 print(var2)        # imprime o número decimal 2.0 atribuído.
19 print(var3)        # imprime o caractere/texto 2 atribuído.
20 print(var1 + var2) # imprime a soma dos valores numéricos atribuídos.
21 print(var1 + var1) # imprime a soma dos valores numéricos atribuídos.
22 print(var3 + var3) # imprime a junção dos caracteres atribuídos.
```

Código-fonte 3.1 – Variáveis (Codificação)

Ao executar no IDE esta codificação que exemplifica as variáveis, o resultado obtido imprime, em sequência, os seus valores/significados, conforme expresso a seguir.

```
1  2
2  3.5
3  5.5
4  Olá!
5  7.5
6  Olá!7.5
7  2
8  2.0
9  2
10 4.0
11 4
12 22
```

Código-fonte 3.2 – Variáveis (Execução)

Aqui, é importante alertar que, ao programar utilizando a linguagem Python, os números decimais são representados por meio de pontos e não por vírgulas. Sendo assim, uma variável flutuante, isto é, uma variável cujo valor é um número expresso por representação decimal, deve ser digitada de modo que um ponto separe a parte inteira da parte fracionária, pois a vírgula apresenta outra finalidade na codificação. A execução dos programas também representam os números decimais por meio de pontos.

Além do mais, alerta-se também para a diferença na manipulação dos dados no formato de texto (`str`) e no formato numérico (`int` e `float`). Por exemplo, na codificação `print(var1 + var1)` são processados dois valores numéricos (`2 + 2`) e o resultado impresso é `4`, diferentemente da codificação `print(var3 + var3)`, em que são manipuladas duas variáveis no formato de texto (`2 + 2`), cujo resultado impresso é `22`. A diferença é que as variáveis `var1` e `var2`, por terem sido declaradas como valores numéricos, serão interpretadas como tal pelo compilador; por outro lado, `var3` é uma string, logo o interpretador trabalha com o caractere/símbolo "2", e não com o valor que representa. Em cada caso, o operador "+" tem um efeito diferente, adequado ao tipo de variável que é manipulado. Sugere-se ao leitor a analise detalhada deste exemplo, que expressa a codificação seguida da execução, observando minuciosamente cada comando e sua respectiva impressão.

3.2.2 Comentários

Nas linguagens de programação é possível incluir comentários no código fonte com a finalidade de inserir observações e descrições pertinentes, conforme foi apresentado em vermelho no Código-fonte 3.1. Estes comentários são ignorados pelo compilador no momento da execução da codificação e são definidos por meio de comandos apropriados.

No Python os comentários podem ser definidos por meio de dois comandos distintos. Para comentar em uma única linha, utiliza-se o símbolo `#`; dessa forma, o que for digitado após este símbolo será ignorado na execução do programa. Para comentários com mais de uma linha, utiliza-se uma sequência de três aspas (“““ ”””) que abrem e fecham o comentário; sendo assim, os caracteres que se encontram entre as aspas são ignorados na execução, conforme exemplifica-se a seguir.

```
1 print('Inserindo comentários.')
2 # Este é um comentário e será ignorado no momento da execução do programa.
3
4 """
5 Este também é um comentário
6 que apresenta várias linhas
```

```
e também será ignorado no momento
da execução do programa.
"""
```

Código-fonte 3.3 – Comentários (Codificação)

```
Inserindo comentários.
```

Código-fonte 3.4 – Comentários (Execução)

3.2.3 Interatividade com o usuário

Alguns programas podem interagir com um usuário solicitando e fornecendo informações. Então, algumas variáveis podem ser definidas por meio desta interatividade, quando o programa solicitar ao usuário que este digite e atribua um valor/significado, o que torna a execução do programa mais dinâmica.

O Python possui uma função que captura a entrada de valores/significados: a função `input()`. Destaca-se que a tradução da palavra inglesa *input* é, justamente, entrada. Esta função possibilita uma pausa na execução do programa para que o usuário digite algo e, após apertar a tecla *Enter*, o valor/significado atribuído à variável é armazenado, processado e impresso como um `str` (string). A função `input()` sempre vai retornar um `str`; dessa forma, para a manipulação de dados dos tipos `int` e `float` (inteiros e flutuantes), é necessário que o programador especifique e declare o tipo da variável.

Conforme o exemplo exposto no Código-fonte 3.5 mais adiante, em um programa que solicita ao usuário nome, idade e altura com a intenção de armazenar, processar e imprimir esses dados, o programador deve declarar os diferentes tipos dessas variáveis para a eficiência na manipulação dos dados. Ressalta-se que, nesse contexto, as variáveis nome, idade e altura são, respectivamente dos tipos `str`, `int` e `float`. Alerta-se, também, a importância e necessidade desses comandos serem organizados entre parênteses e, ainda, adverte-se que o esquecimento de uma aspa ou um parênteses pode ocasionar erros na execução do programa.

```
nome = input('Digite seu nome: ')
idade = int(input('Digite a sua idade: '))
altura = float(input('Digite a sua altura (m): '))
print('\n')
print('O nome informado foi', nome, '.')
print('A idade informada foi', idade,'anos.')
```

```
7 print('A altura informada foi', altura,'m.')
8
9 """ Também estaria correto declarar o tipo string ao codificar:
10 nome = str(input('Digite seu nome: ')).
11 No entanto, não é necessário."""
```

Código-fonte 3.5 – Interatividade com o usuário (Codificação)

```
1 Digite seu nome: Fulano de Tal
2 Digite a sua idade: 25
3 Digite a sua altura (m): 1.78
4
5
6 O nome informado foi Fulano de Tal.
7 A idade informada foi 25 anos.
8 A altura informada foi 1.78 m.
```

Código-fonte 3.6 – Interatividade com o usuário (Execução)

Aponta-se que, no Código-fonte 3.5, o comando `\n` representa uma nova linha. Dessa forma, ao codificar `print('\n')`, o interpretador irá quebrar (ou pular) uma linha no momento da execução do programa, como exibe o Código-fonte 3.6. Além disso, sugere-se, nesta codificação, que o programador deixe um espaço no final da linha ao usar a função `input()`, pois favorece a estética da execução do programa.

Destaca-se, também, que para a impressão desses dados, por meio da função `print()`, é necessário mencionar a variável separando-a do texto por meio de uma vírgula. Outra maneira de imprimir esses dados seria utilizando a função `.format()`, que possibilita a substituição de campos entre chaves (`{}`) por um string definido previamente como uma variável. Esta função pode facilitar a impressão dos dados em uma única linha facilitando alguns comandos. Dessa forma, cada caso tem sua particularidade e o programador pode escolher usá-los conforme considere mais apropriado.

Expressa-se no Código-fonte 3.7 uma codificação alternativa que exibe a utilização da função `.format()`, promovendo ao leitor uma comparação com o Código-fonte 3.5, já apresentado. Espera-se que o leitor perceba que, na execução da codificação, os três pares de chaves são substituídos ordenadamente pelos respectivos valores/significados das variáveis definidas na utilização da função `.format()`.

```
1 nome = input('Digite seu nome: ')
2 idade = int(input('Digite a sua idade: '))
3 altura = float(input('Digite a sua altura (m): '))
```

```
4
5 print('\nSeu nome é {}, você tem {} anos e {} metros de altura.'.format(nome,
      idade, altura))
6 print('É um prazer te conhecer, {}.' .format(nome))
```

Código-fonte 3.7 – Interatividade com o usuário (Codificacão - opção alternativa)

```
1 Digite seu nome: Beltrano de Tal
2 Digite a sua idade: 26
3 Digite a sua altura (m): 1.83
4
5 Seu nome é Beltrano de Tal, você tem 26 anos e 1.83 metros de altura.
6 É um prazer te conhecer, Beltrano de Tal.
```

Código-fonte 3.8 – Interatividade com o usuário - (Execucão - opção alternativa)

Considera-se importante chamar a atenção, aqui, para a presença do ponto na representação da codificação que se expressa a função `.format()`, bem como a sua funcionalidade ser conjunta com a presença de um par de chaves. Aliás, um detalhe matemático interessante na utilização dessa função é que pode-se escolher a quantidade de casas decimais quando se usa uma variável do tipo `float`, o que pode favorecer a representação de dízimas. A codificação `{:.1f}` retorna um número flutuante com apenas uma casa decimal quando as chaves são substituídas pelo valor da variável definida na função `.format()`. De forma similar os códigos `{:.2f}`, `{:.3f}` e `{:.4f}`, são substituídos, respectivamente, por números flutuantes com duas, três e quatro casas decimais. Exemplifica-se a utilização deste comando nos Códigos-fonte 3.11 e 3.12.

3.2.4 Operadores básicos

Os operadores em Python possibilitam que o programador transcreva argumentos lógicos em códigos. Por meio dos operadores, consegue-se estabelecer uma correspondência entre a linguagem matemática e linguagem de programação. Existem tipos de operadores diferentes e com finalidades distintas, dos quais destaca-se, aqui, os operadores aritméticos, os operadores de comparação e os operadores lógicos.

Tabela 1 – Operadores Aritméticos

Operador	**Nome**	**Operação**	**Descrição**	**Exemplo**
`+`	Adição	*a* `+` *b*	Soma entre *a* e *b*	5 `+` 4 = 9
`-`	Subtração	*a* `-` *b*	Diferença entre *a* e *b*	5 `-` 4 = 1
`*`	Multiplicação	*a* `*` *b*	Produto entre *a* e *b*	5 `*` 4 = 20
`/`	Divisão	*a* `/` *b*	Quociente entre *a* e *b*	5 `/` 4 = 1.25
`//`	Divisão Inteira	*a* `//` *b*	Quociente inteiro entre *a* e *b*	5 `//` 4 = 1
`%`	Módulo	*a* `%` *b*	Resto da divisão de *a* por *b*	5 `%` 4 = 1
`**`	Potenciação	*a* `**` *b*	Base *a* elevada ao Expoente *b*	5 `**` 4 = 625

Fonte: Elaborado pelo autor.

Tabela 2 – Operadores de Comparação

Operador	**Nome**	**Operação**	**Descrição**	**Exemplo**
`==`	igual	*a* `==` *b*	*a* igual a *b*	a `==` 1
`!=`	diferente	*a* `!=` *b*	*a* diferente de *b*	b `!=` 2
`<`	menor do que	*a* `<` *b*	*a* menor que *b*	c `<` 3
`<=`	menor do que ou igual a	*a* `<=` *b*	*a* menor ou igual a *b*	d `<=` 4
`>`	maior do que	*a* `>` *b*	*a* maior que *b*	e `>` 5
`>=`	maior do que ou igual a	*a* `>=` *b*	*a* maior ou igual a *b*	f `>=` 6

Fonte: Elaborado pelo autor.

Tabela 3 – Operadores Lógicos

Operador	**Descrição**	**Exemplo**
`and`	conectivo e	`a != 0 and b > 4`
`or`	conectivo ou	`c == 2 or d <= 5`

Fonte: Elaborado pelo autor.

3.2.4.1 Operações e expressões aritméticas

É possível realizar operações matemáticas por meio de codificações na linguagem Python. Para isso, basta digitar a sequência operacional que se deseja em uma linha de comando. Pode-se construir uma expressão como uma combinação de valores, variáveis e operadores. Tal expressão pode ainda ser definida como uma variável e, ao imprimi-la na execução, obtêm-se o resultado (similar a uma calculadora).

Logo a seguir, exemplifica-se a utilização e aplicação dos operadores aritméticos por meio da codificação e execução de comandos simples.

```
1  n1 = int(input('Digite um número: '))
2  n2 = int(input('Digite outro número: '))
3  print('='*10)      # imprime o símbolo = dez vezes.
4  print(n1 + n2)     # imprime a soma dos inteiros n1 e n2 atribuídos.
5  print(n1 - n2)     # imprime a diferença entre os inteiros n1 e n2 atribuídos.
6  print(n1 * n2)     # imprime a produto entre os inteiros n1 e n2 atribuídos.
7  print(n1 / n2)     # imprime o quociente entre os inteiros n1 e n2 atribuídos.
8  print(n1 // n2)    # imprime a quociente inteiro entre n1 e n2 atribuídos.
9  print(n1 % n2)     # imprime o resto da divisão entre n1 e n2 atribuídos.
10 print(n1 ** n2)    # imprime a potência, onde n1 é a base e n2 é o expoente.
11 print('='*10)      # imprime o símbolo = dez vezes.
```

Código-fonte 3.9 – Operadores Aritméticos (Codificação)

```
1  Digite um número: 2
2  Digite outro número: 6
3  ==========
4  8
5  -4
6  12
7  0.3333333333333333
8  0
9  2
10 64
11 ==========
```

Código-fonte 3.10 – Operadores Aritméticos (Execução)

Relembra-se a necessidade de declarar o tipo de variável utilizada, para evitar erros no processamento dos dados. Por exemplo, no Código-fonte 3.9, especifica-se que as variáveis são números inteiros (`int`), promovendo uma correta manipulação dos dados. Alerta-se que, na ausência do comando `int`, as variáveis seriam consideradas como string (textos), o que provocaria erros nas operações, pois a execução do programa, como apresentada no Código-fonte 3.10, até apresentaria o valor `26` para o comando `print(n1 + n2)`, no entanto, para as demais operações, o programa encerraria a execução e acusaria um erro na codificação.

Apesar disso, podemos operar com strings também dentro de um contexto pretensioso, como exemplifica o código `print('='*10)` apresentado, que multiplica por dez o símbolo "=", cujo formato refere-se a um string. Dessa maneira, o símbolo "=" se

repete dez vezes. Apresentando um outro exemplo similar, mostra-se que o código `print('\n'*10)` permite pular a linha dez vezes.

Vale ressaltar que, na ausência da radiciação dentre os sete operadores apresentados, pode-se recorrer à potenciação com um expoente fracionário, em conformidade com a propriedade aritmética: $\sqrt[n]{a^m} = a^{\frac{m}{n}}$. Por exemplo, seguindo a codificação apresentada no código-fonte 3.9 e digitando os comandos `n1 = 2` e `print(n1**(1/2))`, a execução exibiria o resultado `1.4142135623730951`, referente a $\sqrt{2} = 1.4142135623730951...$.

É importante destacar que, no caso dos operadores aritméticos, existe uma hierarquia na ordem da realização das operações. Assim como numa expressão numérica resolve-se primeiramente a radiciação e a potenciação (na ordem em que aparecem), posteriormente a multiplicação e a divisão (na ordem em que aparecem) e, por fim, a adição e a subtração (na ordem em que aparecem), de maneira similar, os operadores aritméticos no Python obedecem a mesma ordem. No entanto, é necessário se atentar às grafias e representações desses comandos, sendo fundamental utilizar parênteses para organizar melhor e obter êxito em determinados comandos.

Seguindo a estética da codificação anterior, apresenta-se, abaixo, um exemplo de uma codificação de uma expressão aritmética qualquer.

```
1 # Exemplo de uma expressão aritmética
2 print('Calcularemos o quociente entre o quadrado da soma de dois números
      INTEIROS e o produto entre esses dois números.')
3 n1 = int(input('Digite o primeiro número: '))
4 n2 = int(input('Digite o segundo número: '))
5 expressão = ((n1 + n2)**2)/(n1*n2)
6 print(expressão)
7 print('O valor da expressão é aproximadamente {:.2f}'.format(expressão))
8 # O comando {:.2f} possibilita que o valor da variável 'expressão' seja
      representada com apenas duas casas decimais.
```

Código-fonte 3.11 – Expressão - Operadores Aritméticos (Codificação)

```
1 Calcularemos o quociente entre o quadrado da soma de dois números INTEIROS e o
      produto entre esses dois números.
2 Digite o primeiro número: 2
3 Digite o segundo número: 6
4
5 5.333333333333333
6 O valor da expressão é aproximadamente 5.33
```

Código-fonte 3.12 – Expressão - Operadores Aritméticos (Codificação)

No Código-fonte 3.11, por exemplo, a variável definida por $expressão = \frac{(n1+n2)^2}{n1 \cdot n2}$ é representada na codificação por `expressão = ((n1 + n2)**2)/(n1*n2)`, mas deve-se atentar para não cometer equívocos ao esquecer parênteses, por exemplo, na representação do denominador de `expressão = ((n1 + n2)**2)/n1*n2`; claramente se obteria um resultado diferente, pois obedecendo a hierarquia das operações, primeiro se realizaria a divisão, por n1, de todo valor obtido no numerador para, então, multiplicar todo o resultado por n2.

Exibe-se, agora, a codificação e a execução de programas em que se aplicam expressões matemáticas significativas, com a intenção de exemplificar melhor o uso de códigos que exploram os operadores aritméticos numa expressão numérica e num conjunto de operações mais elaboradas. Apresenta-se uma sequência de três programas: o primeiro se trata de um conversor de temperaturas Celcius (C), Fahrenheit (F) e Kelvin (K) (Códigos-fonte 3.13 e 3.14), em que se usa a proporção $\frac{C}{5} = \frac{F-32}{9} = \frac{K-273}{5}$, estudada em Termometria na disciplina de Física no 2º ano do Ensino Médio, para estabelecer as expressões `F = (9*C + 160)/5` e `K = C + 273` na codificação; o segundo, por sua vez, refere-se a um programa que calcula a hipotenusa (h) de um triângulo retângulo, com catetos c_1 e c_2, por meio do Teorema de Pitágoras ($h^2 = c_1^2 + c_2^2$), onde a expressão que calcula a hipotenusa é definida por `h = (c1**2 + c2**2)**(1/2)` (Códigos-fonte 3.15 e 3.16); por fim, o terceiro programa permite calcular a média aritmética (denotada por MA) e geométrica (MG) entre três valores n_1, n_2 e n_3 (Códigos-fonte 3.17 e 3.18), onde representa-se as expressões $MA = \frac{n_1+n_2+n_3}{3}$ e $MG = \sqrt[3]{n_1 \cdot n_2 \cdot n_3}$ por meio das codificações `MA = (n1+n2+n3)/3` e `MG = (n1*n2*n3)**(1/3)`. As codificações dos três programas declaram as variáveis como `float`, possibilitando, assim, a entrada de valores decimas.

```
1  #Delimitando o ''cabeçalho'' do programa.
2  print('='*25)
3  print('CONVERSOR DE TEMPERATURA')
4  print('='*25) #Fim do ''cabeçalho''.
5
6  print('\nPodemos converter uma temperatura em Celsius para Fahrenheit e Kelvin
       .')
7  C = float(input('Informe a tempertura em ºC: '))
8  F = (9*C + 160)/5  #Expressão que converte Celsius para Fahrenheit.
9  K = C + 273        #Expressão que converte Celsius para Kelvin.
10 print('A temperatura de {}ºC corresponde a {:.2f}ºF, bem como a {:.2f}K.'.
       format(C, F, K))
```

Código-fonte 3.13 – Conversor de Tempertura (Codificação)

```
1 ========================
2 CONVERSOR DE TEMPERATURA
3 ========================
4
5 Podemos converter uma temperatura em Celsius para Fahrenheit e Kelvin.
6 Informe a tempertura em ºC: 36.6
7 A temperatura de 36.6ºC corresponde a 97.88ºF, bem como a 309.60K.
```

Código-fonte 3.14 – Conversor de Tempertura (Execução)

```
1 print('='*20)
2 print('TEOREMA DE PITÁGORAS')
3 print('='*20)
4
5 print('\nVAMOS CALCULAR A HIPOTENUSA DE UM TRIÂNGULO RETÂNGULO?')
6 c1 = float(input('Digite o comprimento de um dos catetos: '))
7 c2 = float(input('Digite o comprimento do outro cateto: '))
8 h = (c1**2 + c2**2)**(1/2) #Expressão que calcula a hiptenusa.
9 print('\nA medida procurada da hipotenusa é {:.2f}.'.format(h))
```

Código-fonte 3.15 – Teorema de Pitágoras (Codificação)

```
1 ====================
2 TEOREMA DE PITÁGORAS
3 ====================
4
5 VAMOS CALCULAR A HIPOTENUSA DE UM TRIÂNGULO RETÂNGULO?
6 Digite o comprimento de um dos catetos: 12
7 Digite o comprimento do outro cateto: 10
8
9 A medida procurada da hipotenusa é 15.62.
```

Código-fonte 3.16 – Teorema de Pitágoras (Execução)

```
1 print('='*30)
2 print('MÉDIAS ARITMÉTICA E GEOMÉTRICA')
3 print('='*30)
4
5 print('\nVAMOS CALCULAR AS MÉDIAS ARITMÉTICA E GEOMÉTRICA ENTRE TRÊS NÚMEROS?'
    )
```

```
6 n1 = float(input('Digite um número: '))
7 n2 = float(input('Digite outro número: '))
8 n3 = float(input('Digite outro número: '))
9
10 MA = (n1+n2+n3)/3       #Expressão que calcula a Média Aritmética.
11 MG = (n1*n2*n3)**(1/3)  #Expressão que calcula a Média Geométrica.
12
13 print('\nCom base nos três números informados, a média aritmética é {:.2f} e a
       média geométrica é {:.2f}.'.format(MA, MG))
```

Código-fonte 3.17 – Cáculos das médias aritmética e geométrica (Codificação)

```
1 ==============================
2 MÉDIAS ARITMÉTICA E GEOMÉTRICA
3 ==============================
4
5 VAMOS CALCULAR AS MÉDIAS ARITMÉTICA E GEOMÉTRICA ENTRE TRÊS NÚMEROS?
6 Digite um número: 24
7 Digite outro número: 35.2
8 Digite outro número: 87
9
10 Com base nos três números informados, a média aritmética é 48.73 e a média
     geométrica é 41.89.
```

Código-fonte 3.18 – Cáculos das médias aritmética e geométrica (Execução)

3.2.5 Indentação

A indentação é uma distância em relação à margem esquerda expressa no início de uma linha em que se representam caracteres, ou seja, é um espaço no começo de uma linha escrita, como uma espécie de parágrafo.

O uso da indentação é relevante para o funcionamento da execução da codificação, pois alguns comandos exigem uma organização por meio de uma separação de estruturas. Dessa forma, um conjunto de comandos com a mesma indentação possuem a mesma estrutura. Um exemplo de sua aplicação é expresso na Subseção 3.2.6. Para a eficiência na funcionalidade da execução da codificação é necessário a separação e organização dessas estruturas, o que ocorre por meio desses espaços das linhas em relação à margem esquerda.

3.2.6 Estruturas de Decisão: Condicionais `if`, `else` e `elif`.

Nas linguagens de programação existem estruturas de decisão que são caracterizadas pela composição de um conjunto de instruções que são codificadas para serem executadas de forma condicional. As estruturas de decisão permitem moldar o fluxo e modificar a trajetória da execução de um programa, baseando-se na veracidade de um argumento lógico. Esta utilidade permite analisar e validar a entrada de dados que respeitam certas condições. Basicamente, os condicionais determinam a sequência pela qual as instruções de um programa são executadas, a programação testa se uma condição pré-definida é verdadeira ou falsa e então executa comandos de acordo com esse resultado.

No Python utiliza-se os comandos `if` e `else` (cujas traduções da língua inglesa são, respectivamente, "se" e "senão") para estabelecer certas condições e estruturar decisões. Além desses, o comando `elif` também pode compor uma estrutura de decisão como um condicional intermediário entre `if` - `else`, apresentando-se como um complemento de ambos.

As estruturas podem se apresentar como uma condição simples, uma condição composta ou uma condição aninhada, conforme detalha-se a seguir:

- Condição Simples: `if` - Usada, de forma simples, para decidir se determinado trecho do programa deve ou não ser executado. Impõe uma condição que será executada somente se a atribuição da condição for verdadeira.
- Condição Composta: `if - else` - Usada para decidir, entre dois comandos, qual será executado. Estrutura-se duas condições, onde uma condição é processada como verdadeira e a outra como falsa, e somente uma delas será executada.
- Condição Aninhada: `if - elif - else` - Usada quando há várias condições, onde cada uma é associada a um comando que será, ou não, executado conforme sua veracidade.

Para entender, na prática, a utilização das estruturas de decisão, apresenta-se, a seguir, o código-fonte de um programa que analisa a maioridade de um usuário e, na sequência expõe-se considerações explicativas ao seu respeito. Divide-se a codificação em três partes, conforme os três tipos de condições apresentadas, onde todas solicitam a idade do usuário para processar a informação e imprimir uma mensagem conforme uma condição pré-definida.

```
1 #CONDIÇÃO SIMPLES
2 idade = int(input('Quantos anos você tem? '))
```

```
3 if idade >=18:
4    print('Você é maior de idade.')
5 print('Sua idade é', idade, 'anos.')
6
7 #CONDIÇÃO COMPOSTA
8 idade = int(input('Quantos anos você tem? '))
9 if idade >=18:
10   print('Você é maior de idade.')
11 else:
12   print('Você é menor de idade.')
13 print('Sua idade é', idade, 'anos.')
14
15 #CONDIÇÃO ANINHADA
16 idade = int(input('Quantos anos você tem? '))
17 if idade >=18:
18   print('Você é maior de idade.')
19 elif idade <=10:
20   print('Você é uma criança.')
21 else:
22   print('Você é menor de idade.')
23 print('Sua idade é', idade, 'anos.')
```

Código-fonte 3.19 – Estruturas de Decisão (Codificação)

A princípio, destaca-se o uso da indentação para organizar as estruturas de decisão e a sua utilização é necessária para a correta execução do programa. Os comandos presentes nas linhas sem a indentação serão sempre executados, já os comandos com a indentação estabelecida podem, ou não, serem executados, o que depende da decisão tomada na execução do programa.

A condição simples impõe que a codificação definida por `print('Você é maior de idade.')` somente seja executada caso o valor da variável "idade" fornecida pelo usuário seja um número maior do que ou igual a 18. Caso contrário (idade < 18), o comando não será executado e o interpretador trabalhará nas linhas subsequentes.

A condição composta, por sua vez, ordena que a codificação representada por `print('Você é maior de idade.')` seja executada somente **se** a idade fornecida pelo o usuário for maior ou igual a 18 (idade ≥ 18). Caso isso não ocorra (**senão**), isto é, se a idade não for maior ou igual a 18 (idade < 18), então o compilador passará à próxima linha e executará o próximo comando `print('Você é menor de idade.')` definido previamente pelo condicional `else`.

Por fim, a condição aninhada determina três condições em que somente uma delas será executada. O programa imprimirá a mensagem “ `Você é maior de idade.` ” somente se o usuário fornecer um valor maior ou igual a 18. Caso o usuário forneça uma idade menor ou igual a 10 anos, então a execução do programa imprimirá: “`Você é uma criança.`”. Finalmente, se nenhuma das opções acima ocorrer, isto é, se o usuário fornecer uma idade entre 10 e 18 anos ($10 < \text{idade} < 18$), então o programa executará a mensagem “`Você é menor de idade.`” e, na sequência, seguirá para o fim de sua execução. Baseado nisso, menciona-se que a codificação apresentada para a condição aninhada (Código-fonte 3.19) pode ser substituída por uma equivalente, onde o último condicional (`else:`) é substituído por `elif 10 < idade < 18:` com a finalidade de descrever melhor o seu comando, o que não interfere na execução, pois trata-se de uma programação equivalente (conforme apresentada no Código-fonte 3.20). Isso possibilita inserir mais condições, em sequência, utilizando o condicional `elif`.

```
1 #CONDIÇÃO ANINHADA
2 idade = int(input('Quantos anos você tem? '))
3 if idade >=18:
4    print('Você é maior de idade.')
5 elif idade <=10:
6    print('Você é uma criança.')
7 elif 10 < idade < 18: #Descreve-se também a última condição.
8    print('Você é menor de idade.')
9 print('Sua idade é', idade, 'anos.')
```

Código-fonte 3.20 – Condição Aninhada (Codificação equivalente)

Alerta-se para a necessidade do uso de dois pontos para a correta definição do condicional. A ausência dos dois pontos acarreta erro na execução do programa.

Com a intenção de explorar exemplos distintos sobre as estruturas de decisão, apresenta-se, agora, a codificação e a execução de quatro programas que aplicam o uso de condicionais em situações e contextos diferentes. O primeiro utiliza uma condição composta e o operador aritmético `%` para analisar se um número é, ou não, divisível por 3 (Códigos-fonte 3.21 e 3.22), quando o operador analisa se o resto da divisão é zero ou não. O segundo, utiliza os operadores lógicos `and` e `or` associados a outros operadores aritméticos e de comparação para definir condicionais que analisam se um ano é, ou não, bissexto (Códigos-fonte 3.23 e 3.24). O terceiro, faz uso de uma condição aninhada que estrutura três condições que informam o status de aprovação de um aluno ao calcular a sua média escolar (Códigos-fonte 3.25 e 3.26). Por fim, o quarto programa utiliza uma condição aninhada com várias condições estabelecidas pelo comando `elif`,

que informam a condição de saúde de uma pessoa diante do cálculo do seu índice de massa corporal (Códigos-fonte 3.27 e 3.28).

```
1  print('='*20)
2  print('DIVISIBILIDADE POR 3')
3  print('='*20)
4
5  print('\nPosso informar se um número é DIVISÍVEL por 3 ou não!')
6  número = int(input('Diga-me um número qualquer: '))
7  resultado = número % 3
8
9  if resultado == 0:
10    print('O número {} É DIVISÍVEL por 3.'.format(número))
11 else:
12    print('O número {} NÃO É DIVISÍVEL por 3.'.format(número))
```

Código-fonte 3.21 – Divisibilidade por 3 (Codificação)

```
1  ====================
2  DIVISIBILIDADE POR 3
3  ====================
4
5  Posso informar se um número é DIVISÍVEL por 3 ou não!
6  Diga-me um número qualquer: 8472
7  O número 8472 É DIVISÍVEL por 3.
```

Código-fonte 3.22 – Divisibilidade por 3 (Execução)

Para a codificação do próximo programa, é necessário ter o conhecimento das definições impostas pelo calendário gregoriano. Em um ano bissexto acrescenta-se um dia extra aos 365 dias normais já existentes, totalizando 366 dias. Isso ocorre a cada quatro anos (exceto anos múltiplos de 100 que não são múltiplos de 400). Dessa forma, dividimos a nossa primeira condição (`if`) em duas partes: primeiro, o ano deve ser múltiplo de 4 **e** não ser múltiplo de 100, o que é codificado por (`ano % 4 == 0 and ano % 100 != 0`), isto é, o ano deixa resto zero na divisão por 4 e deixa resto diferente de zero na divisão por 100; segundo, o ano deve ser múltiplo de 400, o que automaticamente o torna múltiplo de 4, sendo assim, codifica-se `(ano % 400 == 0)`, o que significa que o ano deve deixar resto zero na divisão por 400. Separa-se essas duas partes utilizando o operador lógico `or`, indicando que ou uma ou outra parte dessa condição deve ocorrer

para que o ano seja bissexto.

```
1  print('='*12)
2  print('ANO BISSEXTO')
3  print('='*12)
4
5  print('\nPosso lhe informar se um ano é ou não BISSEXTO!')
6  print('Duvida!?')
7  ano = int(input('Qual ano você gostaria de verificar? '))
8
9  if (ano % 4 == 0 and ano % 100 != 0) or (ano % 400 == 0):
10    print('O ano {} É BISSEXTO.'.format(ano))
11 else:
12    print('O ano {} NÃO É BISSEXTO.'.format(ano))
```

Código-fonte 3.23 – Ano Bissexto (Codificação)

```
1  ============
2  ANO BISSEXTO
3  ============
4
5  Posso lhe informar se um ano é ou não BISSEXTO!
6  Duvida!?
7  Qual ano você gostaria de verificar? 2022
8  O ano 2022 NÃO É BISSEXTO.
```

Código-fonte 3.24 – Ano Bissexto (Execução)

```
1  print('='*13)
2  print('MÉDIA ESCOLAR')
3  print('='*13)
4
5  print('Vamos calcular a média de um determinado aluno e analisar seu status de
       Aprovação!')
6  n1 = float(input('Digite a primeira nota: '))
7  n2 = float(input('Digite a segunda nota: '))
8
9  m = (n1 + n2)/2
10 print('Sendo suas notas {:.1f} e {:.1f}, então sua MÉDIA é igual a {:.1f}.'.
      format(n1, n2, m))
```

```
11
12 if m >= 5 and m < 7:      #equivalente a  if 5 <= m < 7:
13    print('O aluno está em RECUPERAÇÃO!')
14 elif m < 5:
15    print('O aluno está REPROVADO!')
16 else:
17    print('O aluno está APROVADO!')
```

Código-fonte 3.25 – Média Escolar (Codificação)

```
1 =============
2 MÉDIA ESCOLAR
3 =============
4 Vamos calcular a média de um determinado aluno e analisar seu status de aprova
     ção!
5 Digite a primeira nota: 5.8
6 Digite a segunda nota: 8.1
7 Sendo suas notas 5.8 e 8.1, então sua MÉDIA é igual a 6.9.
8 O aluno está em RECUPERAÇÃO!
```

Código-fonte 3.26 – Média Escolar (Execução)

```
1 print('='*24)
2 print('ÍNDICE DE MASSA CORPORAL')
3 print('='*24)
4
5 massa = float(input('Qual a sua massa(kg)?  '))
6 altura = float(input('Qual a sua altura(m)?  '))
7 imc = massa/(altura**2)
8
9 print('Seu IMC é {:.1f} kg/m^2.'.format(imc))
10 print('STATUS:')
11 if imc < 16:
12    print("Magreza grave.")
13 elif imc < 17:
14    print("Magreza moderada.")
15 elif imc < 18.5:
16    print("Magreza leve.")
17 elif imc < 25:
18    print("Saudável.")
```

```
19 elif imc < 30:
20     print("Sobrepeso.")
21 elif imc < 35:
22     print("Obesidade.")
23 elif imc < 40:
24     print("Obesidade severa.")
25 else:
26     print("Obesidade mórbida.")
```

Código-fonte 3.27 – Índice de Massa Corporal (Codificação)

```
1 ========================
2 ÍNDICE DE MASSA CORPORAL
3 ========================
4 Qual a sua massa(kg)?  75.4
5 Qual a sua altura(m)?  1.74
6 Seu IMC é 24.9 kg/m^2.
7 STATUS:
8 Saudável.
```

Código-fonte 3.28 – Índice de Massa Corporal (Execução)

3.2.7 Importação de bibliotecas

As bibliotecas são um conjunto de funções úteis que reduzem o uso de códigos na elaboração de um programa. O Python dispõe de mais de 137 mil bibliotecas que facilitam o tratamento de dados. Pode-se utilizar uma função pronta por meio de uma importação da biblioteca, o que pode favorecer o programador ao tentar criar um comando muito elaborado, pois o que poderia levar várias linhas de codificação pode ser codificado de forma sucinta utilizando uma função importada. "Isso aumenta a produtividade do programador, pois ao utilizarmos bibliotecas usamos programas desenvolvidos e testados por outras pessoas. Isso reduz o número de erros e permite que você se concentre realmente no problema que quer resolver." (MENEZES, 2010).

Menciona-se, aqui, a biblioteca `math`, que pode favorecer a codificação de muitos programas que utilizam funções matemáticas. No entanto, ressalta-se, também, que a proposta pedagógica deste trabalho sugere que as elaborações dos programas sejam conduzidas pelo professor sem a utilização de bibliotecas, de modo que o pensamento matemático dos alunos seja desenvolvido por meio da associação de uma linguagem corrente e matemática com a linguagem de programação.

A importação das bibliotecas deve ser declarada de forma explícita no programa e deve ser feita utilizando o comando `import`. Isso pode ser feito de duas formas diferentes: importando-se a biblioteca inteira ou apenas uma função específica. Por exemplo, com o comando `import math` é possível importar todas as funções matemáticas da biblioteca `math`; já na utilização do comando `from math import hypot` importa-se, desta biblioteca, apenas a função `hypot` que permite calcular a hipotenusa de um triângulo retângulo. Utilizar o primeiro caso pode não ser conveniente, pois, ao importar mais de uma biblioteca, algumas funções podem ter o mesmo nome, mas apresentam aplicabilidades distintas, o que pode comprometer o comportamento da execução do programa.

Exemplifica-se, no Código-fonte 3.29 abaixo, a utilização da função `hypot` da biblioteca `math` mencionada acima. Dessa forma, possibilita-se ao leitor uma comparação de uma programação alternativa de uma codificação já apresentada na Subseção 3.2.4 por meio do Código-fonte 3.15.

```
1 from math import hypot  #Importando a função hypot da biblioteca math que será
      utilizada na linha 5.
2 print('\nVAMOS CALCULAR A HIPOTENUSA DE UM TRIÂNGULO RETÂNGULO?\n')
3 c1 = float(input('Digite o comprimento de um dos catetos: '))
4 c2 = float(input('Digite o comprimento do outro cateto: '))
5 h = math.hypot(c1, c2)
6 print('A medida procurada da hipotenusa é {:.2f}.'.format(h))
```

Código-fonte 3.29 – Biblioteca math (Codificação)

3.2.8 Erros

O Python possui uma sintaxe própria, um vocabulário próprio. Deve-se atentar para a correta representação dos comandos para se obter êxito na execução das codificações. Digitar algo que o interpretador não entenda causará um erro no programa. Alguns erros podem ocorrer por falta de atenção ou aptidão, como esquecer parênteses ou de colocar a mensagem entre aspas (simples ou duplas), ou ainda não organizar uma indentação.

> Programar exige muita paciência e, principalmente, atenção a detalhes. Uma simples vírgula no lugar de um ponto ou aspas esquecidas podem arruinar seu programa. No início é comum perder a calma ou mesmo se desesperar até aprender a ler o que realmente escrevemos em nossos programas. Nessas horas, paciência nunca é demais. Leia novamente a mensagem de erro ou pare para entender o que não está funcionando corretamente. (MENEZES, 2010)

Os IDEs PyCharm e Spyder evidenciam os erros quando estes ocorrem e direcionam a sua correção por meio de uma orientação, apontando a linha da codificação de um eventual comando errado.

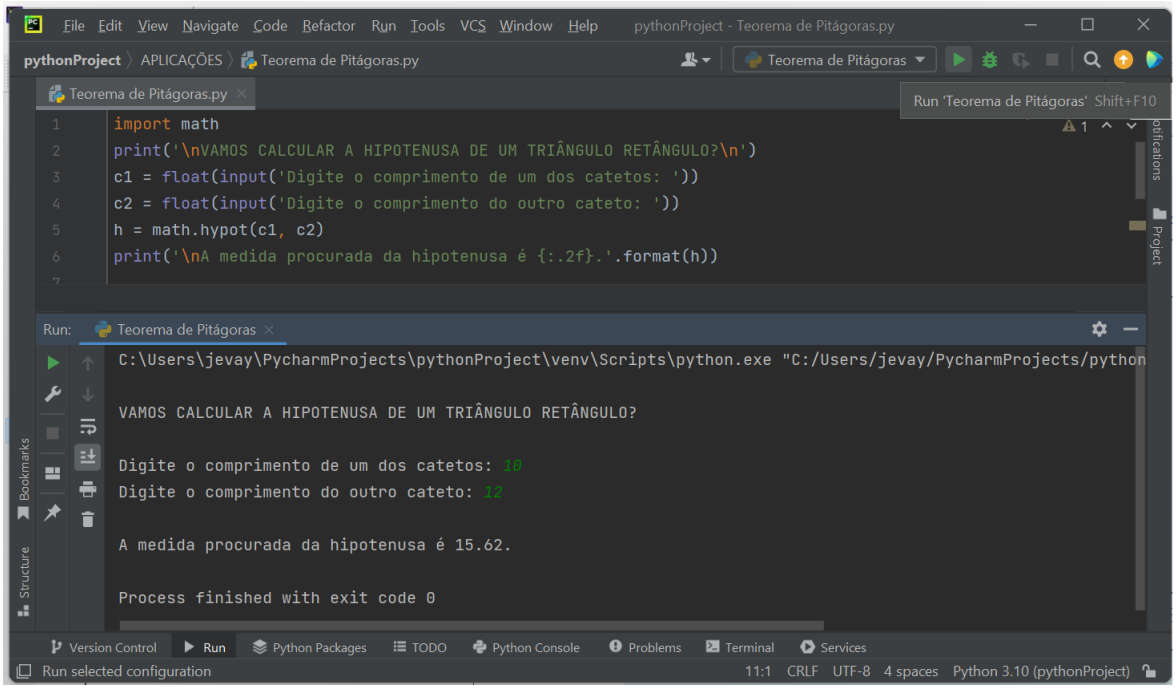
3.3 O Python no Computador: IDEs PyCharm e Spider

Após a apresentação deste tutorial, pretende-se, agora, utilizar um dos códigos-fonte expostos para ilustrar melhor o uso dos IDEs sugeridos (PyCharm e Spider). Na ilustração destes Ambientes de Desenvolvimento Integrado, espera-se que o leitor perceba os espaços apropriados para a elaboração dos códigos e a execução destes, como também a forma como a codificação é elaborada e como o programa interage com o usuário ao ser executado. Na execução são solicitados dados numéricos que são armazenados e manipulados após pressionar a tecla Enter.

As Figuras 6 e 7 exibem os espaços adequados para a codificação e execução nos referidos IDEs. Para estas ilustrações, utiliza-se do último código-fonte (3.29) apresentado no tutorial, que exibe a importação da biblioteca `math` para a utilização da função `hypot` que calcula a hipotenusa de um triângulo retângulo. Informa-se ao leitor que, para executar o programa no Pycharm, basta clicar no botão "Run" ou pressionar os botões "shift + F10"; já no Spyder, de maneira similar, clica-se no botão "Executar" ou pressiona-se o botão "F5".

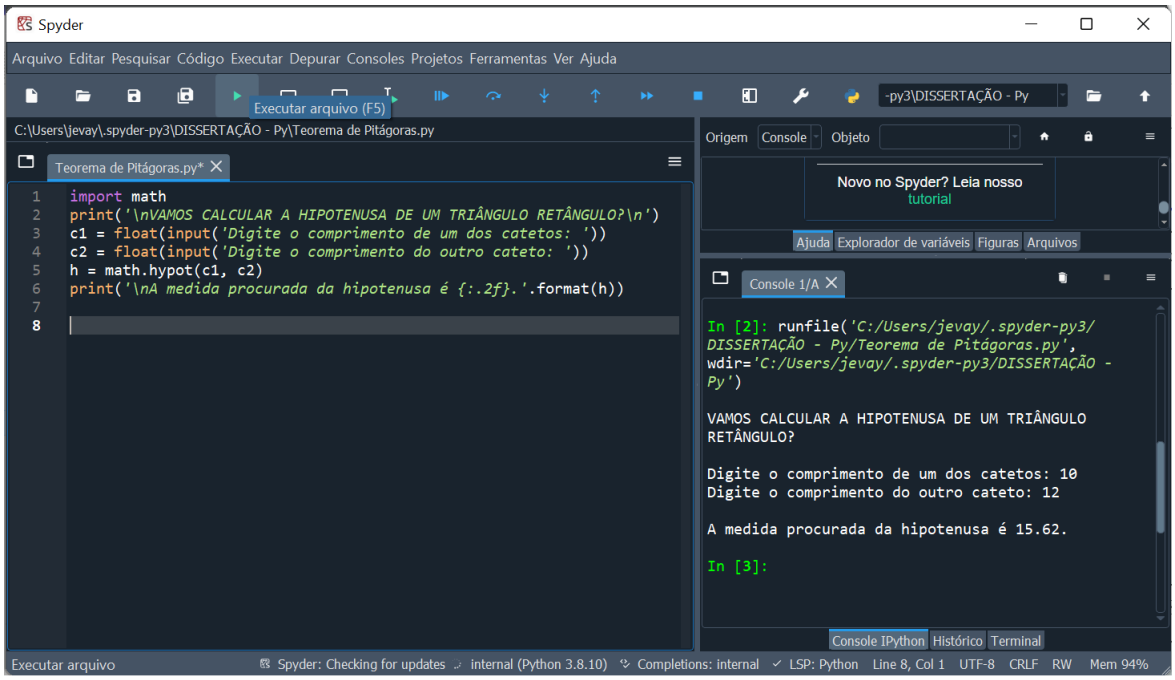
Ressalta-se que, em ambos os IDEs, é possível salvar os projetos elaborados, para alterá-los ou executá-los posteriormente.

Figura 6 – IDE PyCharm - Teorema de Pitágoras



Fonte: Elaborada pelo autor.

Figura 7 – IDE Spyder - Teorema de Pitágoras



Fonte: Elaborada pelo autor.

3.4 O Python no Celular: aplicativos para Android e IOS

Com a intenção de fortalecer o aprendizado da Linguagem de Programação Python, sugere-se, nesta seção, o uso de aplicativos para celular que podem se tornar um recurso adicional nos estudos desta linguagem de programação. Defende-se que programar no computador é muito mais confortável e eficiente, no entanto, usar o celular como um item complementar, pode favorecer o aprendizado e a prática.

Aponta-se, aqui, aplicativos que promovem um tutorial com explicações teóricas, como também lições e exercícios acerca desta linguagem de programação. Além disso, indica-se, também, o uso de aplicativos que podem funcionar como um IDE, o que permite a elaboração e execução de códigos-fonte, sobretudo armazená-los na memória do celular.

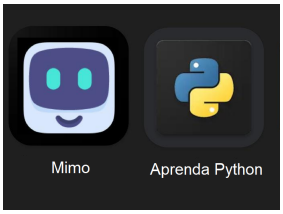
Quanto a essas indicações, deixa-se claro ao leitor que se trata de uma mera sugestão e acrescenta-se a orientação de que outros aplicativos podem ser buscados e testados. Neste instante, sugere-se dois aplicativos que promovem um tutorial para reforçar o aprendizado do Python.

O primeiro, denominado por "Mimo", promove um pequeno curso com explicações, lições e exercícios que conduzem a prática diária, por meio da elevação de níveis. Este aplicativo está disponível tanto para celulares com o sistema operacional IOS quanto Android. Além do Python, o Mimo também oferta uma instrução sobre o desenvolvimento na web por meio do HTML (Linguagem de Marcação de HiperTexto) que descreve a aparência (CSS) e a funcionalidade (JavaScript) de uma página da internet. O Mimo pode ser usado gratuitamente, podendo, inclusive, se obter mais recursos funcionais por meio de uma versão paga.

O segundo, por sua vez, é totalmente gratuito e está disponível apenas para Android e é denominado por "Aprenda Python". Este aplicativo promove uma abordagem dos conceitos teóricos sobre a programação com Python, exibindo exemplos e viabilizando a realização de exercícios práticos.

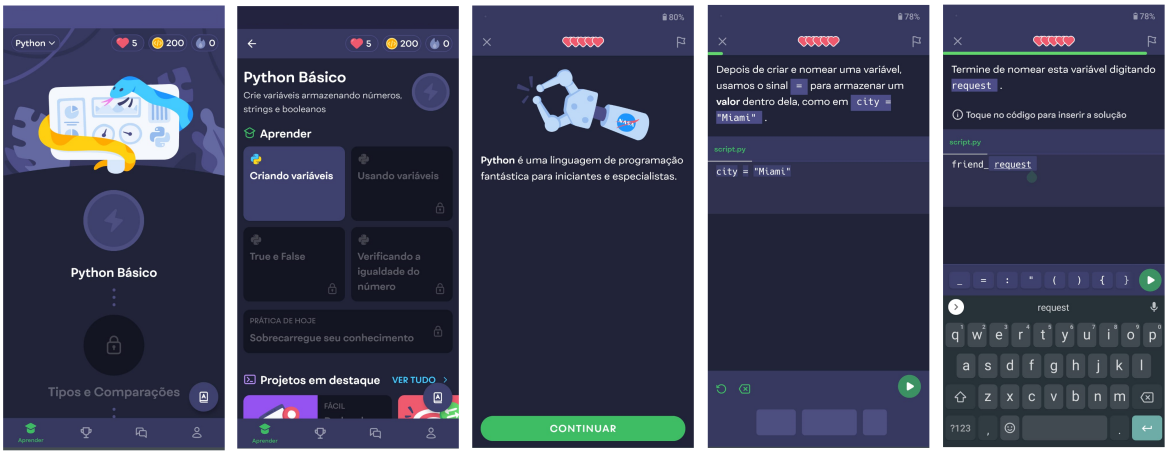
Abaixo, ilustra-se a utilização destes aplicativos.

Figura 8 – Ícones dos aplicativos sugeridos



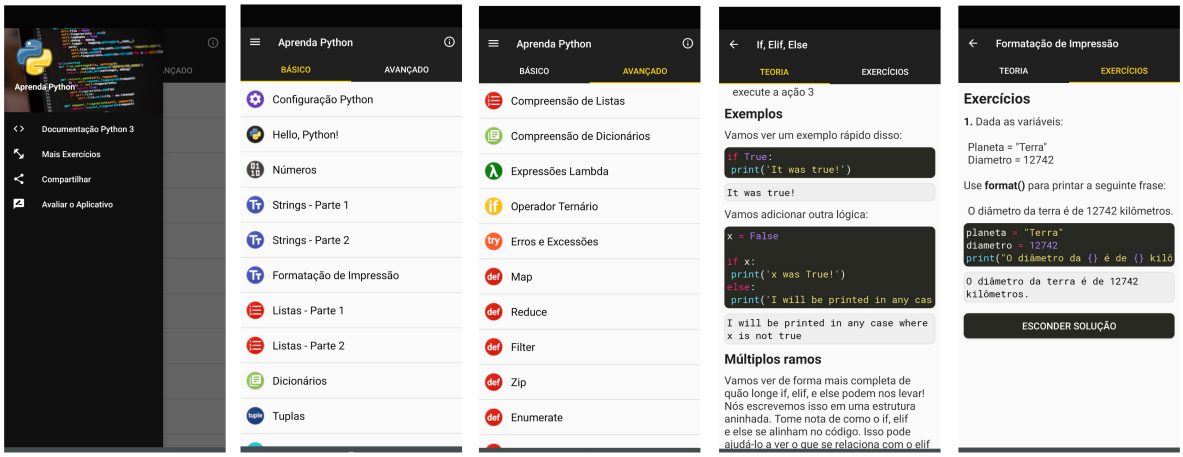
Fonte: Elaborada pelo autor.

Figura 9 – Aplicativo Mimo



Fonte: Elaborada pelo autor.

Figura 10 – Aplicativo Aprenda Python



Fonte: Elaborada pelo autor.

Sugere-se, agora, um outro aplicativo que funciona como um Ambiente de Desenvolvimento Integrado e permite a codificação e execução de programas. O aplicativo designado por "Pydroid 3" está disponível apenas para celulares com o sistema operacional Android.

A Figura 11 exibe a utilização do Pydroid para a elaboração de um programa que permite a conversão de temperaturas, demonstrando a codificação e a execução do programa sugerido nos Códigos-fonte 3.13 e 3.13 expostos no tutorial da Seção 3.2 deste capítulo (3).

Figura 11 – Aplicativo Pydroid 3

Fonte: Elaborada pelo autor.

Espera-se que os IDEs mencionados neste capítulo (Pydoid, PyCharm e Spyder) possam auxilar os professores e alunos que recorrem às propostas pedagógicas abordadas neste trabalho, contribuindo para o manuseio do Python e facilitando a resolução dos problemas apresentados nos Capítulos 4 e 5.

4 Problemas e Aplicações

Neste capítulo, pretende-se apresentar alguns problemas que indicam o uso da Linguagem de Programação Python para criar programas que empreguem em sua codificação conceitos matemáticos do Ensino Médio. A proposta é que a produção e aplicação dessas codificações resolvam problemas sobre tópicos específicos da área de Matemática e Suas Tecnologias, apoiando, assim, o processo de ensino e aprendizagem desta área do conhecimento.

Sugere-se que os professores desafiem os seus alunos a produzirem programas que utilizem conceitos matemáticos em sua codificação, porém sem a importação de bibliotecas, uma vez que tais ferramentas promovem a resolução de um problema de programação matemática de forma mecânica, rápida e objetiva, sem o emprego de uma análise matemática operacional mais profunda e detalhada.

Dessa forma, propõem-se que os alunos sejam desafiados a realizar uma programação onde seja necessário ter domínio dos conceitos, das linguagens e das representações matemáticas essenciais relacionadas ao tópico matemático abordado, com o intuito de obter a solução do problema. Para isso, é necessário, também, que os docentes tratem previamente com os discentes algumas instruções simples de como utilizar a Linguagem de Programação Python, conforme foi exposto no Capítulo 3.

É imprescindível que os alunos tenham o conhecimento da hierarquia presente no uso de símbolos e operações na linguagem matemática, por exemplo, a organização na estrutura, na ordem e na quantidade dos parenteses utilizados, ou ainda, o fato de que numa expressão numérica a multiplicação sempre é realizada antes da adição, mesmo que na representação da expressão a multiplicação se apresente depois.

Deseja-se mostrar que, mesmo com alguns poucos comandos, é possível realizar uma grande variedade de atividades utilizando a Linguagem de Programação Python. Serão discutidos, neste capítulo, problemas envolvendo conceitos, linguagens, representações e aplicações de Funções Afim e Quadrática, Progressões Aritmética e Geométrica, Matemática Financeira e, também, Área e Classificação de Triângulos.

Os problemas serão apresentados obedecendo exatamente essa ordem, se aproximando da ordem com que esses tópicos são estudados no Ensino Médio, e não numa ordem hierárquica de complexidade na programação. Sendo assim, o grau de dificuldade de cada problema varia de acordo com cada tópico estudado e, ainda, a complexidade na elaboração de cada codificação independe da ordem de apresentação dos problemas.

Antes das apresentações dos problemas, realiza-se uma discussão sobre como cada tópico é abordado em sala de aula, relembrando definições e expondo experiências do-

centes que possam ser sugestivas para a vivência em sala de aula. Isso pode contribuir para a formulação das estratégias pedagógicas dos professores que leiam este trabalho acadêmico. Sugere-se que, após as discussões teóricas dos conceitos em sala de aula, sejam usados os problemas de linguagem de programação aqui expostos, com a finalidade de enriquecer a estratégia pedagógica do professor.

Após as exposições dos enunciados dos problemas de programação, são apresentados comentários que direcionam para a sua solução. Na sequência apresentam-se soluções sugestivas, onde exibe-se um programa executado e, em seguida, o referido código-fonte elaborado. Sugere-se aos professores leitores desse trabalho que tentem solucionar os problemas antes de visualizar a sua resposta. Inclusive, orienta-se, também, que os códigos aqui expostos sejam digitados manualmente pelo leitor, pois isso pode favorecer a consolidação do domínio e da habilidade da programação ao exigir, nisso, o exercício e a prática constante. Porém, o leitor pode recorrer, também, ao ato de copiar e colar, pragmaticamente, os códigos apresentados.

Almeja-se, ainda, expor considerações sobre os problemas, de modo particular, ressaltando pontos importantes e comentando detalhes simples que podem favorecer o aprendizado do tópico matemático em questão, auxiliar no desenvolvimento do letramento matemático dos discentes e aprimorar o manejo da linguagem de programação na resolução dos problemas.

4.1 Funções Afim e Quadrática

Ao discutir em sala de aula sobre as formalidades nas definições, linguagens e representações das Funções, verbaliza-se corriqueiramente, com a intenção de facilitar o entendimento, que Função é a relação entre duas grandezas, em que uma é dada em função da outra, ou ainda, em que uma grandeza depende da outra. Geralmente tais grandezas são representadas pelas varáveis x e y, onde a grandeza y depende da grandeza x.

Formalmente, argumenta-se que Função é a relação entre dois conjuntos A e B, não vazios, tal que a todo elemento x de A corresponda, de acordo com uma lei de formação, a um único elemento y de B e representa-se tal relação por $f : A \to B$.

Além disso, são tratados em sala de aula conceitos como valor numérico e raiz de uma função, expondo para o aluno operações e aplicações onde pode-se encontrar o valor de y atribuindo-se um valor para x, bem como encontrar o valor de x que torna nulo o valor de y.

Apresenta-se, a seguir, problemas de programação que analisam algumas das formalidades nas representações dessas relações de dependência em funções, como também algumas aplicações.

4.1.1 Função Polinomial do 1º Grau (Função Afim)

Sendo a e b números reais e $a \neq 0$, dá-se o nome de função polinomial do 1º grau, ou função afim, a toda função que pode ser escrita sob a forma $f : \mathbb{R} \to \mathbb{R}$ com $f(x) = ax + b$, também representada por $y = ax + b$. Nesse caso, a e b são chamados coeficientes da função, onde a é o coeficiente angular e b o coeficiente linear.

Em sala de aula, expõe-se que esta função pode ser representada geometricamente no plano cartesiano por meio de uma reta. Além disso, induz-se o aluno a perceber que a variável y depende da operacionalização da variável x com coeficientes de naturezas diferentes que moldam a estética dessa relação de dependência. Dessa forma, compreendido isso, a efetivação do letramento matemático do aluno o faz perceber, por exemplo, que o coeficiente angular a influencia na inclinação da reta e, ainda, na razão da variação da variável dependente y, ou até mesmo, que o coeficiente linear b permite a visualização de quando a variável independente se anula, isto é, $x = 0$.

A concretização do letramento matemático do aluno permite que este perceba o motivo das distintas nomenclaturas para os coeficientes e para as variáveis, além de notar a necessidade e as diferenças de suas manipulações.

Defende-se, aqui, que utilizar a Programação para criar projetos digitais que abordem esses conceitos pode auxiliar na ludicidade dos seus estudos e favorecer a efetivação da sua compreensão, pois seus tratos se apoiam em aplicações diretas. Inclusive, o êxito dos alunos na programação desses projetos e solução desses problemas só ocorre, de fato, com a consumação do letramento matemático.

Logo adiante, sugere-se problemas que discorrem sobre essas colocações.

Problema 1 - Crie um programa que forneça a raiz de uma função afim. Para isso, o programa deve solicitar ao usuário quais os coeficientes angular e linear da função para, então, representá-la e expressar a sua raiz.

Ao apresentar o enunciado de um problema desta natureza, o professor pode regular o nível de dificuldade ao provocar a inspiração dos alunos, onde a decisão do discente do melhor caminho a se percorrer na elaboração da codificação da programação pode mudar conforme o enunciado se apresenta. Por exemplo, expor meramente o enunciado "Crie um programa que informe ao usuário a raiz de uma função afim." provoca uma percepção diferente daquela em que se apresentam sugestões de comandos como "o programa deve solicitar ao usuário quais os coeficientes angular e linear da função...". Dessa maneira, o docente tem a possibilidade de moldar a sua trajetória de forma didática.

O professor pode, ainda, apresentar uma execução sugestiva do programa pronto conforme o código-fonte executado em 4.1. Isso pode facilitar a elaboração da estética

da programação a ser usada pelo aluno. Dessa forma, o professor pode gerenciar a condução dos problemas, tratando-os como desafios ou exercícios com graus de dificuldade diferentes baseado-se no nível em que a turma se apresenta no domínio dos comandos e codificações da linguagem de programação Python.

```
1  FUNÇÃO POLINOMIAL DO 1º GRAU (Função Afim)
2  -=--=--=--=--=--=--=--=--=--=--=--=--=--=-
3
4  Vamos calcular a RAIZ de uma Função Afim?
5  Informe os coeficientes da função.
6
7  Digite o coeficiente angular: -2
8  Digite o coeficiente linear: 3.5
9
10 A função é do 1º grau.
11 A função pretendita é f(x) = -2.0x + (3.5).
12 A raiz da função é 1.75.
```

Código-fonte 4.1 – Raiz de uma Função Afim (Execução)

Para elaborar esse programa e resolver esse problema, o aluno deve ter o conhecimento de que raiz de uma função é o valor da variável independente x que torna o valor da variável dependente y igual a zero. No que se refere à Função Afim ($y = ax + b$), para que $y = 0$, é necessário que $x = \dfrac{-b}{a}$, pois

$$y = ax + b \implies y = a \cdot \frac{-b}{a} + b \implies y = 0.$$

Portanto, ao elaborar a codificação deste programa, o aluno deve dominar que a raiz de uma função afim é obtida por meio do quociente entre o simétrico do coeficiente linear e o coeficiente angular.

Abaixo, no Código-fonte 4.2, expressa-se uma possível solução para o Problema 1.

```
1 print("\nFUNÇÃO POLINOMIAL DO 1º GRAU (Função Afim)")
2 print('-=-'*14)
3
4 print('\nVamos calcular a RAIZ de uma Função Afim?')
5 print('Informe os coeficientes da função.\n')
6
7 a = float(input('Digite o coeficiente angular: '))
8 b = float(input('Digite o coeficiente linear: '))
```

```
9
10 if a != 0:
11     x = -b / a
12     print('\nA funcao é do 1º grau.')
13     print('A função pretendita é f(x) = {}x + ({}).'.format(a, b))
14     print('A raiz da função é {}'.format(x))
15 else:
16     print('\nA função não é do 1º grau.')
```

Código-fonte 4.2 – Raiz de uma Função Afim (Codificação)

Nesta codificação, destaca-se que o programa emprega uma interatividade com o usuário por meio da função `input()` nas linhas 7 e 8, onde solicita a entrada dos coeficientes (isto é, solicita que o usuário digite os coeficientes). Atenta-se para o espaço deixado no final desse comando para favorecer a estética na execução do programa. Os valores fornecidos podem ser números inteiros ou reais não inteiros, uma vez que a variável foi declarada do tipo `float()`. Dessa forma, a função pode apresentar coeficientes decimais, sobretudo sua raiz. Caso o programador deseje que os coeficientes sejam inteiros, deve-se usar a função `int()` ao invés de `float()`.

Ressalta-se, também, que a codificação aplica estruturas de decisão, por meio dos condicionais `if` e `else`, para tratar da existência da função afim, onde utiliza-se do argumento de que se $a \neq 0$ (`if a!=0:`), então o programa escreve que a função é do 1º grau; caso isso não ocorra (`else:`), o programa escreve que a função não é do 1º grau. Evidencia-se, ainda, a importância da indentação ao se fazer uso desses condicionais. Tal espaço que se deixa entre a margem esquerda e o começo da linha escrita é fundamental para que ocorra a correta execução e funcionalidade do programa elaborado.

Caso o usuário forneça o valor 0 (zero) para o coeficiente angular a, o programa se encarrega de classificar a função como não sendo do 1º grau. Em sala de aula essa classificação é discutida com profundidade e, ao elaborar esta codificação, a aluno pode ter em mente que, para $a = 0$, a representação do tipo $f(x) = b$ expressa uma função classificada como constante. Inclusive, considerando a possibilidade do usuário fornecer o valor 0 (zero) para o coeficiente linear b, o aluno pode dominar que $f(x) = ax$ representa uma função classificada como linear. Outras classificações que o aluno deve dominar são funções crescentes e decrescentes, atentando-se para a possibilidade de que o usuário pode fornecer um valor positivo ou negativo para o coeficiente a.

Além disso, um ponto importante dessa codificação, que também merece destaque, é o uso da função `.format()` que possibilita a criação de `strings`, que são conjuntos de caracteres de texto que podem ser entendidos como representações de informações escri-

tas dentro de um código, como os símbolos `{}` que serão substituídos por uma variável definida no código. Por exemplo, em `f(x) = {}x + ({})` seguido de `.format(a,b)`, define-se que os símbolos das chaves serão substituídos pelos valores a e b, nessa ordem, informados pelo usuário no momento da execução do programa.

Problema 2 – Crie um programa que forneça o valor numérico de uma função afim. Para isso, o programa deve solicitar ao usuário quais os coeficientes angular e linear da função para, então, representá-la. Na sequência, o programa deve solicitar ao usuário o valor da variável independente pretendida para, então, se obter o valor da variável dependente dela, isto é, calcular o valor numérico da função.

Como no Problema 1, o nível de dificuldade do problema 2 pode ser intensificado caso o enunciando seja apresentado de forma sucinta como "Crie um programa que informe ao usuário o valor numérico de uma função afim." A partir disso, a construção da codificação que permite produzir esse programa exige um domínio dos conceitos teóricos sobre função afim e sobre os procedimentos operacionais para se obter o valor numérico, bem como conceitos sobre as estruturas e funcionalidades dos códigos de programação.

Apresenta-se, abaixo, a execução de um programa pronto e, na sequência, a sua codificação, resolvendo, assim, o Problema 2.

```
1  FUNÇÃO POLINOMIAL DO 1º GRAU (Função Afim)
2  -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
3
4  Vamos calcular o VALOR NUMÉRICO de uma Função Afim?
5  Informe os coeficientes da função.
6
7  Digite o coeficiente angular: -2.5
8  Digite o coeficiente linear: 4
9  A função informada foi  f(x) = -2.5x + (4.0).
10 Você pretende calcular o valor de f(x) quando x assumir qual valor??
11
12 Informe o valor de x: 3
13 Para x = 3.0, o valor numérico da função f(x) = -2.5x + (4.0) é -3.5.
```

Código-fonte 4.3 – Valor Numérico de uma Função Afim (Execução)

```
1  print('\nFUNÇÃO POLINOMIAL DO 1º GRAU (Função Afim)')
2  print('-=-'*14)
3
4  print('\nVamos calcular o VALOR NUMÉRICO de uma Função Afim?')
```

```
5 print('Informe os coeficientes da função.\n')
6
7 a = float(input('Digite o coeficiente angular: '))
8 b = float(input('Digite o coeficiente linear: '))
9
10 print('A função informada foi f(x) = {}x + ({}).'.format(a, b))
11 print('Você pretende calcular o valor de f(x) quando x assumir qual valor??')
12 x = float(input('\nInforme o valor de x: '))
13
14 y = a*x + b
15
16 print('Para x = {}, o valor numérico da função f(x) = {}x + ({}) é {}.'.format
    (x, a, b, y))
```

Código-fonte 4.4 – Raiz de uma Função Afim (Codificação)

Expõe-se, agora, outro problema sobre Função Afim, do qual explora o cálculo da raiz e do valor numérico aplicada a uma situação corriqueira.

Problema 3 – Desenvolva um programa que calcule o valor a se pagar por um serviço de táxi quando o usuário informa a quantidade de quilômetros rodados. Inclua no programa a funcionalidade de calcular, também, quantos quilômetros é possível andar nesse táxi quando o usuário informa a quantia a ser utilizada. Para isso, considere que um serviço de táxi cobra R$ $6,28$ a bandeirada, mais R$ $14,75$ por quilômetro rodado.

Neste problema, explora-se os conceitos da Função Afim aplicados a uma situação do cotidiano. O aluno deve perceber que o valor a se pagar pelo serviço de táxi é dado em função (depende) da quantidade de quilômetros rodados e, antes de tudo, estabelecer uma lei de formação para esta função.

O professor tem a liberdade de modificar os coeficientes sugeridos nesse problema, bem como mudar a situação de dependência entre as grandezas envolvidas nesse tipo de função. Além dessa circunstância, que indica que o preço a se pagar depende da distância percorrida no táxi, podem ser explorados, também, situações como a de que o salário de um vendedor é dado em função da quantidade dos produtos vendidos, ou que o valor a se pagar numa padaria é dado em função da quantidade de pães que se compra, ou ainda, que o volume no esvaziamento de um reservatório de água é dado em função do tempo, entre outras situações pertinentes e enriquecedoras.

A seguir, expõe-se a execução de um programa pronto que resolve este problema.

```
1 FUNÇÃO POLINOMIAL DO 1º GRAU (Função Afim)
2 ----------
3 PROBLEMA
4 O valor a se pagar por um serviço de táxi é dado em FUNÇÃO da quantidade de
      quilômetros rodados.
5 Um serviço de táxi cobra R$ 6,28 a bandeirada mais R$ 14,75 por quilômetro
      rodado.
6
7 Informe a quantidade de quilômetros rodados nesse táxi e nós informaremos o
      valor a ser pago pela corrida.
8 Digite a quantidade de quilômetros rodados: 4.5
9
10 Ao percorrer 4.5km nesse táxi, o valor a ser pago deve ser R$ 72.66
11
12 Agora, informe a quantia que você tem e nós informaremos quantos quilômetros
      você poderá andar nesse táxi.
13 Digite o valor (em reais): 95
14 Com R$ 95.00 você poderá andar 6.01 km nesse táxi.
```

Código-fonte 4.5 – Função Afim - Problema/Aplicação (Execução)

De forma pretensiosa, este problema é uma prossecução dos dois anteriores, onde, por meio de uma situação rotineira, o aluno aplicará conceitos para o cálculo do valor numérico e, até mesmo, da raiz da função. Ao programar para que o usuário informe a quantidade de quilômetros rodados, o programa irá fornecer um valor numérico para a função. Por outro lado, ao programar para que o usuário forneça a quantia, o programa informará a distância que pode se percorrer nesse táxi. Nesse último caso, por exemplo, o valor da quantia informado pode ser R$ 0,00, o que possibilita ao programa encontrar a raiz da função. No entanto, o aluno deve perceber que algebricamente isso pode até fazer sentido, mas dominando os conceitos relacionados aos conjuntos Domínio, Contradomínio e Imagem de uma função aplicados a esse problema específico, não tem coerência pagar zero reais por uma corrida ou percorrer zero quilômetros nesse táxi.

Sendo, então, y o valor a ser pago ao se percorrer x quilômetros nesse táxi, o aluno deve estabelecer a relação $y = 6,28 + 14,75x$ e entender que o programa pode fornecer o valor de y (valor numérico) caso o usuário forneça o valor de x, ou o contrário, fornecer o valor de x caso o usuário informe o valor de y. Porém, como argumentado anteriormente, o fato de $x = 0$ ou $y = 0$ não faz sentido, pois sendo $x = 0$, tem-se $y = 6,28$ que representa somente a bandeirada; e sendo $y = 0$, obtêm-se $x = -0.43$ que representa, em quilômetros, uma distância negativa. Inclusive, não faz sentido

os usuários fornecerem dados negativos para a distância e para o preço. De forma intuitiva, os alunos devem perceber que os futuros usuários sensatos desse programa não terão a intenção de fornecer valores que não pertençam aos conjuntos domínio e imagem dessa função, contudo, ainda assim, os discentes devem analisar a importância de pensar nessas questões ao elaborar um programa que resolva esse problema.

Tudo isso contribui com o desenvolvimento da capacidade do aluno de formular, empregar e interpretar a matemática em um determinado contexto, ou seja, contribui para o desenvolvimento do seu letramento matemático. Trabalhar isso em sala de aula fornece ao aluno (um principiante em programação), uma boa experiência prévia que, no futuro, possa fornecer um norte para desenvolver programas mais elaborados e minuciosos que resolvam problemas desse tipo com uma codificação mais eficiente do que essa sugerida neste trabalho. Quando esses estudantes terminarem o ensino médio e, porventura, decidirem que a linguagem de programação esteja presente em sua atuação profissional, terão uma preocupação detalhista e minuciosa no trato de problemas como esse.

Logo mais, no Código-fonte 4.6, apresenta-se uma codificação que soluciona o Problema 3.

```
1  print('\nFUNÇÃO POLINOMIAL DO 1º GRAU (Função Afim)')
2  print('=-='*14)
3  print('PROBLEMA')
4
5  print('O valor a se pagar por um serviço de táxi é dado em FUNÇÃO da
       quantidade de quilômetros rodados.')
6  print('Um serviço de táxi cobra R$ 6,28 a bandeirada mais R$ 14,75 por quilô
       metro rodado.')
7
8  print('\nInforme a quantidade de quilômetros rodados nesse táxi e nós
       informaremos o valor a ser pago pela corrida.')
9  x = float(input('Digite a quantidade de quilômetros rodados: '))
10
11 y = 6.28 + 14.75*x
12 if x <= 0:
13    print('Quilometragem inválida! A quilometragem deve ser superior a zero.')
14 else:
15    print('\nAo percorrer {}km nesse táxi, o valor a ser pago deve ser R$ {:.2f
       }'.format(x, y))
16
17 print('\nAgora, informe a quantia que você tem e nós informaremos quantos quil
```

```
    ômetros você poderá andar nesse táxi.')
18 y = float(input('Digite o valor (em reais): '))
19
20 x = (y - 6.28)/14.75
21 if y <= 6.28:
22    print('Não é possível usufruir dos serviços desse táxi com essa quantia,
       pois é insuficiente.')
23    print('A quantia deve ser superior a R$ 6.28.')
24 else:
25    print('Com R$ {:.2f} você poderá andar {:.2f} km nesse táxi.'.format(y, x))
```

Código-fonte 4.6 – Função Afim - Problema/Aplicação (Codificação)

Enfatiza-se que, ao elaborar esta codificação, o aluno deve ter a aptidão com a hierarquia das operações e dos símbolos presentes nas expressões numéricas e algébricas. Por exemplo, é imprescindível entender que o uso dos parenteses na expressão `x = (y - 6.28)/14.75` promove um resultado diferente de `x = y - 6.28/14.75`. Além disso, o uso da função `float()` é pretensiosa, pois permite o manuseio de números decimais, bem como a representação `{:.2f}`, que define que a variável apresente apenas duas casas decimais. Vale lembrar que, na linguagem de programação, a representação dos decimais é feita por meio de um ponto e não utilizando vírgula.

Ao codificar esse programa, o aluno deve estabelecer previamente os conjuntos domínio e imagem da função, onde a sua correta representação pode auxiliar na clareza da codificação do programa. Para este problema específico, o conjunto domínio é dado por $D(f) =]0, +\infty[$, o que direciona o aluno a codificar que a variável x não deve ser um número menor ou igual a zero. Dessa forma, os alunos podem resolver esse fato utilizando os condicionais `if` e `else`, onde `if x <= 0:` se encarrega de alertar aos usuários sobre valores inválidos e `else:` promove o curso mais aceitável na execução desse programa. O mesmo se aplica ao conjunto imagem dado por $Im(f) =]6.28, +\infty[$. Reforça-se que o letramento matemático pode ser aperfeiçoado à medida em que os alunos interagem com programações desse tipo.

Defende-se que discutir estratégias em sala de aula para codificar um programa com essa configuração pode favorecer o efetivo entendimento dos conceitos teóricos que envolvem essa temática, sobretudo suas aplicações. Fazer uso da linguagem de programação beneficia os alunos a ter o domínio do uso de algoritmos escritos em linguagem corrente e matemática para resolver problemas. Programar melhora não só o desenvolvimento do letramento matemático dos alunos, mas também a sua capacidade de resolver problemas, analisando-os sob várias perceptivas.

4.1.2 Função Polinomial do 2º Grau (Função Quadrática)

Sendo $\{a; b; c\} \subset \mathbb{R}$ e $a \neq 0$, dá-se o nome de função polinomial do 2º grau, ou função quadrática, a toda função que pode ser escrita sob a forma $f : \mathbb{R} \to \mathbb{R}$ com $f(x) = ax^2 + bx + c$, também representada por $y = ax^2 + bx + c$. Nesse caso, a, b, e c são chamados coeficientes da função.

Na vivência da sala de aula, o professor descreve que a variável y depende da variável x conforme a representação acima e destaca características importantes a respeito dessa função, como as descritas nos itens a seguir:

- O gráfico de uma função quadrática, de domínio real e contradomínio real, é sempre uma parábola.
- A abertura de uma parábola é chamada de concavidade. Para uma função quadrática na forma $y = ax^2 + bx + c$, tem-se:

  - Se $a > 0$, então a parábola tem concavidade voltada para cima.

  - Se $a < 0$, então a parábola tem concavidade voltada para baixo.
- Toda a função quadrática intercepta o eixo Oy em $(0, c)$.
- Os valores de x para os quais a função intercepta o eixo Ox são raízes ou zeros da função e ocorrem em $(x, 0)$, isto é, quando $y = 0$.

  - Para determinar as raízes da função do 2º grau $f(x) = ax^2+bx+c$, basta resolver a equação $ax^2+bx+c = 0$. Lembrando que as equações do 2º grau completas são aquelas que apresentam todos os coeficientes, ou seja a, b e c são diferentes de zero $(a, b, c \neq 0)$; e que uma equação quadrática é incompleta quando $b = 0$ ou $c = 0$ ou $b = c = 0$.

  - Pode-se usar $x = \dfrac{-b \pm \sqrt{\Delta}}{2a}$, em que $\Delta = b^2 - 4ac$, sendo Δ o discriminante da função.

  - Dessa forma, sobre as raízes de uma função quadrática pode-se afirmar:

  Se $\Delta > 0$, então existem duas raízes reais e diferentes $(x_1 \neq x_2)$.

  Se $\Delta = 0$, então existem duas raízes reais e iguais $(x_1 = x_2)$.

  Se $\Delta < 0$, então não existem raízes reais.
- As coordenadas do vértice da parábola de uma função quadrática podem ser dadas por: $V(\dfrac{-b}{2a}, \dfrac{-\Delta}{4a})$.
- O vértice da parábola pode ser classificado como ponto mínimo ou ponto máximo. Dada a função $f : \mathbb{R} \to \mathbb{R} / f(x) = ax^2 + bx + c$.

- Se $a > 0$, então $f(x)$ admite valor mínimo determinado pela ordenada do vértice da parábola.

- Se $a < 0$, então $f(x)$ admite valor máximo determinado pela ordenada do vértice da parábola.

A seguir, exibe-se problemas de programação que exploram a aplicação de alguns desses conceitos da Função Quadrática.

> **Problema 4** - Crie um programa que forneça a raiz de uma função quadrática. Para isso, o programa deve solicitar ao usuário quais os coeficientes da função para, então, representá-la e expressar as suas raízes. Lembre-se que a natureza das raízes mudam conforme o valor do discriminante.

O problema acima exige que os alunos tenham conhecimento do caminho a se percorrer para encontrar as raízes de uma função quadrática, sobretudo que elas são descritas e diferenciadas sob condições distintas. Desse modo, expor meramente o enunciado "Crie um programa que informe ao usuário a raiz de uma função quadrática.", pode tornar o início da elaboração da codificação um pouco mais desafiador, diferentemente de quando se acrescenta um trecho ao enunciado sobre a importância do cálculo do discriminante, o que facilita e norteia o caminho que o aluno pode escolher ao codificar o programa.

Conforme a dinâmica utilizada na subseção anterior, abaixo apresenta-se a execução de um programa que utiliza da interatividade com o usuário para solicitar os coeficientes e, a partir deles, determinar as raízes de uma função quadrática. Na sequência, o Código-fonte 4.8 exibe a codificação que deu origem a esse programa.

```

FUNÇÃO POLINOMIAL DO 2º GRAU (Função Quadrática)
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

Vamos calcular as RAIZES de uma Função Quadrática?
Informe os coeficientes da função.

Digite o coeficiente a de x^2: -1
Digite o coeficiente b de x: 6
Digite o termo independente c: -9

A função é do 2º grau.
A função pretendida é  f(x) = -1.0x^2 + (6.0)x + (-9.0).

```

```
A função possui duas raízes reais e iguas.
As raízes são 3.0 e 3.0.
```

Código-fonte 4.7 – Raizes de uma Função Quadrática (Execução)

```
print('\nFUNÇÃO POLINOMIAL DO 2º GRAU (Função Quadrática)')
print('-=-'*14)

print('\nVamos calcular as RAIZES de uma Função Quadrática?')
print('Informe os coeficientes da função.\n')

a = float(input('Digite o coeficiente a de x^2: '))
b = float(input('Digite o coeficiente b de x: '))
c = float(input('Digite o termo independente c: '))


if a != 0:
    delta = b**2 - 4*a*c
    x1 = (-b + (delta)**(1/2)) / (2*a)
    x2 = (-b - (delta)**(1/2)) / (2*a)
    print('\nA função é do 2º grau.')
    print('A função pretendida é f(x) = {}x^2 + ({})x + ({}).'.format(a, b, c))
    if delta > 0:
        print('\nA função possui duas raízes reais e distintas.\nAs raízes são
    {} e {}.'.format(x1, x2))
    elif delta == 0:
        print('\nA função possui duas raízes reais e iguas.\nAs raízes são {} e
    {}.'.format(x1, x2))
    elif delta < 0:
        print('\nA função não possui raízes reais, suas raízes são complexas.\
    nAs raízes complexas são {} e {}.'.format(x1, x2))
else:
    print('A função NÃO é do 2º grau.')
```

Código-fonte 4.8 – Raizes de uma Função Quadrática (Codificação)

Aponta-se, nesta codificação, o uso de condições aninhadas, que podem ser entendidas como estruturas condicionais dentro de outras estruturas condicionais. A primeira condição estabelecida no código é indicada pela ausência da indentação no `if` e no `else` e determina a forma como o programa analisa a existência da função quadrática.

Dessa forma, **se** $a \neq 0$ (`if a != 0:`), então o programa exibe que a função é do 2º grau, **senão** (`else:`), então a função não é do 2º grau, e o programa se encarrega de informar isso. Mostra-se, aqui, que o comando `else:` interpreta a informação fornecida pelo usuário de que $a = 0$, diferente do comando `if a != 0:` que interpreta a informação antagônica, isto é, $a \neq 0$. A execução do programa se encarrega de implementar apenas um desses comandos em execução.

Agora, outras estruturas condicionais são definidas internamente em outra categoria por meio da indentação, o que estabelece outro grau de importância para essas novas condições, pois serão executadas apenas se a primeira, ajustada por `if a != 0:`, também o for. O `elif` é entendido como uma estrutura condicional intermediária dentro da seção `if - else` e se apresenta como um complemento de ambos.

Destaca-se que parte da codificação sugerida no Código-fonte 4.8 poderia ser substituída pelo trecho destacado na Figura 12. Dessa forma, sua execução não se alteraria. Na Figura 12 é possível perceber o `elif` como um condicional intermediário entre `if - else`. No entanto, fazendo esta escolha, a representação do último condicional deve se apresentar de forma sucinta como `else:`, com a presença dos dois pontos. Esclarece-se, aqui, que a diferença na representação da codificação do Código-fonte 4.8 acima, apresentada pelo comando `elif delta < 0:`, foi escolhida de forma pretensiosa, apenas para mostrar no código os três contextos do discriminante. Mas optar por `else:` ou `elif delta < 0:` não modifica a execução do código, na verdade, deixa o cenário equivalente.

Figura 12 – Condições Aninhadas.

```
17    print('A função pretendida é f(x) = {}x^2 + ({
18    if delta > 0:
19        print('\nA função possui duas raízes reais
20    elif delta == 0:
21        print('\nA função possui duas raízes reais
22    else:
23        print('\nA função não possui raízes reais,
24 else:
25    print('A função NÃO é do 2º grau.')
```

Fonte: Elaborada pelo autor.

Ademais, essas últimas estruturas condicionais aqui analisadas verificam o valor do discriminante Δ, o que aponta que a codificação sugere que o seu valor seja calculado e analisado separadamente, onde sua representação é descrita por `delta = b**2 - 4*a*c`. Logo na sequência, são calculadas as raízes, cuja codificação se apresenta como `x1 = (-b + (delta)**(1/2))/(2*a)` e `x2 = (-b - (delta)**(1/2))/(2*a)`.

Menciona-se que deve ser do domínio dos alunos a forma como as expressões são representadas, como o correto uso e organização dos parenteses e a representação da radiciação por `**(1/2)`.

Problema 5 - Crie um programa que informe ao usuário o valor numérico de uma função quadrática.

Considerando todas as observações descritas anteriormente, dessa vez, sugere-se que enunciado desse problema se apresente de forma sucinta e direta. Julga-se, inclusive, que o leitor já tenha um conhecimento necessário para solucioná-lo. Da mesma forma, acredita-se que se um professor, ao utilizar desse material, fizer uso da mesma ordem apresentada nesse trabalho, os seus alunos terão maturidade para resolvê-lo.

No entanto, dependendo das intensões pedagógicas e dos objetivos do professor, pode-se acrescentar detalhes sugestivos no enunciado como "o programa deve solicitar ao usuário os coeficientes da função para, então, representá-la. Na sequência, o programa deve pedir o valor da variável independente pretendida para então calcular o seu valor numérico". Como mencionado anteriormente, isso ajuda a moldar a forma como o aluno interpreta o problema e toma suas decisões para resolvê-lo.

Apresenta-se, abaixo, a execução de um programa sugestivo para este problema e, na sequência, a sua codificação.

```
1  FUNÇÃO POLINOMIAL DO 2º GRAU (Função Quadrática)
2  ——————————————————
3
4  Vamos calcular o VALOR NUMÉRICO de uma Função Quadrática?
5  Informe os coeficientes da função.
6
7  Digite o coeficiente a de x^2: 2
8  Digite o coeficiente b de x: 3.5
9  Digite o termo independente c: 4
10 A função informada foi  f(x) = 2.0x^2 + (3.5)x + (4.0).
11 Você pretende calcular o valor de f(x) quando x assumir qual valor??
12
13 Informe o valor de x: —2
14 Para x = —2.0, o valor numérico da função f(x) = 2.0x^2 + (3.5)x + (4.0) é
       5.0.
```

Código-fonte 4.9 – Valor Numérico de uma Função Quadrática (Execução)

```
1 print('\nFUNÇÃO POLINOMIAL DO 2º GRAU (Função Quadrática)')
2 print('-=-'*14)
3
4 print('\nVamos calcular o VALOR NUMÉRICO de uma Função Quadrática? \nInforme
      os coeficientes da função.\n')
5
6 a = float(input('Digite o coeficiente a de x^2: '))
7 b = float(input('Digite o coeficiente b de x: '))
8 c = float(input('Digite o termo independente c: '))
9
10 print('A função informada foi f(x) = {}x^2 + ({})x + ({}).'.format(a, b, c))
11 print('Você pretende calcular o valor de f(x) quando x assumir qual valor??')
12 x = float(input('\nInforme o valor de x: '))
13
14 y = a*(x**2) + b*x + c
15
16 print('Para x = {}, o valor numérico da função f(x) = {}x^2 + ({})x + ({}) é
      {}'.format(x, a, b, c, y))
```

Código-fonte 4.10 – Valor Numérico de uma Função Quadrática (Codificação)

O próximo problema explora conceitos relacionados a alguns pontos importantes no gráfico de uma função quadrática.

Problema 6 - Desenvolva um programa que analise uma função quadrática e informe algumas características do seu gráfico, como
- a concavidade da parábola;
- onde a parábola intercepta o eixo das ordenadas;
- onde a parábola intercepta o eixo das abscissas.
- as coordenadas do vértice da parábola.

O enunciado deste problema é bem objetivo quanto a sua finalidade. Resolvê-lo requer conhecimento sobre características específicas da função quadrática. Conforme a dinâmica adotada neste trabalho, segue-se, em sequência, a execução e a codificação de um programa que resolve este problema.

```
1 FUNÇÃO POLINOMIAL DO 2º GRAU (Função Quadrática)
2 -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
3 PONTOS IMPORTANTES NO GRÁFICO.
4
```

```
Vamos analisar alguns pontos e algumas características no gráfico de uma Função Quadrática?
Informe os coeficientes da função.

Digite o coeficiente a de x^2: -1
Digite o coeficiente b de x: 3
Digite o termo independente c: 4

A função informada foi  f(x) = -1.0x^2 + (3.0)x + (4.0).

Observa-se que:
- A parábola tem concavidade PARA BAIXO.
- A parábola intersepta o eixo das Ordenadas no ponto (0, 4.0)
- A parábola intersepta o eixo das Abscissas nos pontos (-1.0, 0) e (4.0, 0).
- O vértice da parábola é o ponto (1.5, 6.25)
```

Código-fonte 4.11 – Função Quadrática: Pontos importantes no gráfico (Execução)

```
print('\nFUNÇÃO POLINOMIAL DO 2º GRAU (Função Quadrática)')
print('-='*16)
print('PONTOS IMPORTANTES NO GRÁFICO.')

print('\nVamos analisar alguns pontos e algumas características no gráfico de uma Função Quadrática? \nInforme os coeficientes da função.\n')

a = float(input('Digite o coeficiente a de x^2: '))
b = float(input('Digite o coeficiente b de x: '))
c = float(input('Digite o termo independente c: '))

delta = b**2 - 4*a*c
x1 = (- b + (delta)**(1/2))/(2*a)
x2 = (- b - (delta)**(1/2))/(2*a)
xv = -b/(2*a)
yv = -delta/(4*a)

print('\nA função informada foi f(x) = {}x^2 + ({})x + ({}).'.format(a, b, c))
print('\nObserva-se que:')

#CONCAVIDADE
```

```
21 if a > 0:
22    print('— A parábola tem concavidade PARA CIMA.')
23 else:
24    print('— A parábola tem concavidade PARA BAIXO.')
25
26 #EIXO DAS ORDENADAS
27 print('— A parábola intersepta o eixo das Ordenadas no ponto (0, {})'.format(c
       ))
28
29 #EIXO DAS ABSCISSAS
30 if delta > 0:
31    print('— A parábola intersepta o eixo das Abscissas nos pontos ({}, 0) e
       ({}, 0).'.format(x1, x2))
32 elif delta == 0:
33    print('— A parábola intersepta o eixo das Abscissas no ponto ({}, 0).'.
       format(x1))
34 else:
35    print('— A parábola NÃO intersepta o eixo das Abscissas em nenhum ponto.')
36
37 #VÉRTICE
38 print('— O vértice da parábola é o ponto ({} , {})'.format(xv, yv))
```

Código-fonte 4.12 – Função Quadrática: Pontos importantes no gráfico (Codificação)

4.2 Progressões

Nesta seção pretende-se explorar problemas de programação que recorram a informações sobre definições de progressões aritmética e geométrica.

As progressões são apresentadas em sala de aula por meio dos estudos sobre sequências. Geralmente, o professor define as sequências apresentando exemplificações como uma lista ordenada de objetos, figuras ou números, onde se analisa modelos e formulações de padrões. Em seguida, são expostos conceitos sobre sequências numéricas especiais denominadas de progressões.

Progressão é definida como uma sequência que possui uma regularidade de um termo para o outro, conhecida como razão. Existem dois casos de progressão: a progressão aritmética e a progressão geométrica.

Sugere-se que, após as discussões teóricas dos conceitos de progressões em sala de aula, sejam usados problemas de programação com a finalidade de enriquecer a

estratégia pedagógica que visa contribuir para o efetivo domínio teórico dos conceitos.

4.2.1 Progressão Aritmética

Define-se Progressão Aritmética (PA) como toda sequência em que cada termo, a partir do segundo, é igual a **soma** do termo anterior com uma constante r, denominada razão. Dessa forma, sendo a sequência $(a_n) = (a_1, a_2, a_3, ..., a_n, ...)$ uma Progressão Aritmética de razão r, pode-se afirmar que

- Um termo qualquer é igual ao seu anterior mais a razão: $a_n = a_{n-1} + r$, $\forall n \geq 2$.
- A razão é determinada pela diferença entre um termo qualquer (a partir do segundo) e o seu anterior: $r = a_n - a_{n-1}$.
- Se $r > 0$, a PA é crescente; se $r < 0$, a PA é decrescente; se $r = 0$, a PA é constante.
- O n-ésimo termo de uma PA é dado por $a_n = a_1 + (n-1)r$.
- A soma dos n primeiros termos de uma PA é obtida pela expressão $S_n = \frac{(a_1 + a_n)n}{2}$, para $n \in \mathbb{N}$.

Agora, explora-se, no problema a seguir, a construção de uma programação que utiliza-se das representações e formalidades desses conceitos.

Problema 7 - Desenvolva um programa que determine um termo qualquer de uma progressão aritmética e obtenha a soma, a partir do primeiro, até um termo desejado.

Este problema, com enunciado sucinto, sugere que o aluno pense no procedimento para determinar um termo qualquer de uma PA, o que pode ser feito por meio da formulação do termo geral dado por $a_n = a_1 + (n-1)r$. O problema, ainda, não menciona uma PA específica e deixa aberto a possibilidade para se explorar uma grande variedade de progressões ariméticas. Sendo assim, um possível caminho seria propor a análise de uma PA qualquer, o que pode ser feito questionando ao usuário o primeiro termo e a razão e, logo em seguida, construir uma codificação que exiba a aparência da progressão analisada. A partir dessa exibição, pode-se definir as variáveis com as operações necessárias para se determinar um termo qualquer e a soma dos n primeiros termos que o usuário desejar. Abaixo, exprime-se uma sugestão que resolve este problema, mostra-se a execução de um programa e, na sequência, a sua codificação.

```
PROGRESSÃO ARITMÉTICA
-=--=--=--=--=--=--=--=-
Podemos determinar um termo qualquer de uma Progressão Aritmétca,
basta você nos informar a posição que ele ocupa, além de alguns dados!

Informe o primeiro termo: -8
Informe a razão: 3
Agora, informe a posição do termo: 100

A Progressão Aritmética pode ser representada por (-8, -5, -2, 1, 4, ...)
E o 100º termo dessa P.A. é 289.

Ademais, podemos informar a soma dos primeiros termos dessa P.A.
Quantos números você deseja somar: 5
A soma dos 5 primeiros termos dessa P.A. é -10.0.
```

Código-fonte 4.13 – Progressão Aritmética (Execução)

```
print('\nPROGRESSÃO ARITMÉTICA')
print('-=-'*8)

print('Podemos determinar um termo qualquer de uma Progressão Aritmétca,
    \nbasta você nos informar a posição que ele ocupa, além de alguns dados!')

a1 = int(input('\nInforme o primeiro termo: '))
r = int(input('Informe a razão: '))
n = int(input('Agora, informe a posição do termo: '))

an = a1 + (n-1)*r

print('\nA Progressão Aritmética pode ser representada por ({}, {}, {}, {},
    {}, ...).'.format(a1, a1+r, a1+2*r, a1+3*r, a1+4*r))
print('E o {}º termo dessa P.A. é {}.'.format(n, an))

print('\nAdemais, podemos informar a soma dos primeiros termos dessa P.A.')
n = int(input('Quantos números você deseja somar: '))

an = a1 + (n-1)*r #Há a necessidade de definir a variável an novamente, uma
```

```
    vez que o seu valor é utilizado para calcular sn.
19 sn = (a1 + an)*n/2
20
21 print('A soma dos {} primeiros termos dessa P.A. é {}.'.format(n, sn))
```

Código-fonte 4.14 – Progressão Aritmética (Codificação)

Nesta codificação, as variáveis `a1` e `r` permitem exibir a aparência da PA mostrando os seus cinco primeiros termos dados por `a1, a1+r, a1+2*r, a1+3*r, a1+4*r`. Dominar essa representação mostra que o aluno entendeu que qualquer termo de uma PA é obtido por meio da soma do primeiro termo com a razão uma quantidade de vezes que se refere ao antecessor da posição que ele ocupa. A variável `n`, por sua vez, quando unida às duas anteriores, permite o cálculo do n-ésimo termo pretendido, definido por meio da variável `an`. Para o cálculo da soma, ao definir a variável `sn`, é necessário ajustar novos valores para `n` e `an`, uma vez que o n-ésimo termo desejado pode mudar.

4.2.2 Progressão Geométrica

Denomina-se Progressão Geométrica (PG) toda sequência em que cada termo, a partir do segundo, é igual ao **produto** do termo anterior com uma constante q, denominada razão. Dessa forma, sendo a sequência $(a_n) = (a_1, a_2, a_3, ..., a_n, ...)$ uma Progressão Geométrica de razão q, pode-se afirmar que

- Um termo qualquer é igual ao seu anterior vezes a razão: $a_n = a_{n-1} \cdot q$, $\forall n \geq 2$.
- A razão é determinada pelo quociente entre um termo qualquer (a partir do segundo) e o seu anterior: $q = \dfrac{a_n}{a_{n-1}}$.
- Sobre a classificação

  - Se $q < 0$, a PG é alternada ou oscilante;

  - Se $q = 0$, a PG é constante;

  - Se $a_1 > 0$ e $q > 1$ ou se $a_1 < 0$ e $0 < q < 1$, a PG é crescente;

  - Se $a_1 < 0$ e $q > 1$ ou se $a_1 > 0$ e $0 < q < 1$, a PG é decrescente.

- O termo geral de uma PG é definido por $a_n = a_1 \cdot q^{(n-1)}$.
- A soma dos n primeiros termos de uma PG é obtida por $S_n = a_1 \cdot \dfrac{1 - q^n}{1 - q}$.

Agora, verifica-se, a seguir, um problema sobre a construção de uma programação que faz uso das representações e das formalidades desses conceitos.

Problema 8 - Desenvolva um programa que determine um termo qualquer de uma progressão geométrica e obtenha a soma dos termos que o usuário desejar.

Como no Problema 7, o Problema 8 também apresenta um enunciado sucinto. O nível de dificuldade pode ser facilitado caso ele exponha detalhes como "o programa deve solicitar ao usuário o primeiro termo e a razão da PG", entre outras informações que podem nortear como o aluno pode moldar a elaboração do seu programa. O professor tem a liberdade para modificar a forma como o enunciado se apresenta, desde que a essência e o objetivo da solução sejam os mesmos.

De maneira semelhante ao que foi exposto no Problema 7 sobre PA, abaixo exprime-se uma sugestão que resolve o Problema 8, além de sua execução e sua codificação.

```
1
2 PROGRESSÃO GEOMÉTRICA
3 -=--=--=--=--=--=--=--=-
4 Podemos determinar um termo qualquer de uma Progressão Geométrica,
5 basta você nos informar a posição que ele ocupa, além de alguns dados!
6
7 Informe o primeiro termo: 3
8 Informe a razão: -2
9 Agora, informe a posição do termo: 10
10
11 A Progressão Geométrica pode ser representada por (3, -6, 12, -24, 48, ...).
12 E o 10º termo dessa P.G. é -1536.
13
14 Ademais, podemos informar a soma dos primeiros termos dessa P.G.
15 Quantos números você deseja somar: 5
16 A soma dos 5 primeiros termos dessa P.G. é 33.0.
```

Código-fonte 4.15 – Progressão Geométrica (Execução)

```
1 print('\nPROGRESSÃO GEOMÉTRICA')
2 print('-=-'*8)
3
4 print('Podemos determinar um termo qualquer de uma Progressão Geométrica,\
    nbasta você nos informar a posição que ele ocupa, além de alguns dados!')
5
6 a1 = float(input('\nInforme o primeiro termo: '))
7 r = float(input('Informe a razão: '))
8 n = int(input('Agora, informe a posição do termo: '))
```

```
9
10 an = a1 * r**(n-1)
11
12 print('\nA Progressão Geométrica pode ser representada por ({}, {}, {}, {},
       {}, ...).'.format(a1, a1*r, a1*r**2, a1*r**3, a1*r**4))
13 print('E o {}º termo dessa P.G. é {}.'.format(n, an))
14
15 print('\nAdemais, podemos informar a soma dos primeiros termos dessa P.G.')
16 n = int(input('Quantos números você deseja somar: '))
17
18 # an = a1 * r**(n-1). Não há a necessidade de definir a variável an novamente.
19 sn = a1*(r**n - 1)/(r-1)
20 print('\nA soma dos {} primeiros termos dessa P.G. é {}.'.format(n, sn))
```

Código-fonte 4.16 – Progressão Geométrica (Codificação)

A estética desse programa e de sua codificação é muito similar àquela apresentada para o Problema 7, modificando-se, apenas, a forma como as operações das variáveis são definidas em virtude do Problema 8 tratar de uma progressão distinta. Diferentemente da codificação para a P.A., na codificação deste problema sobre P.G. não há a necessidade de definir a variável `an` novamente, uma vez que ela não é utilizada para o cálculo da soma dos primeiros termos `sn`.

4.3 Matemática Financeira

A Matemática Financeira utiliza uma série de conceitos matemáticos aplicados à análise de dados financeiros em geral, estudando as relações entre os valores monetários e suas datas. Os problemas clássicos de matemática financeira são ligados a questão do valor do dinheiro no decorrer do tempo e como isso é aplicado a empréstimos, investimentos e avaliação financeira de projetos.

Nesta seção, serão discutidos problemas de programação que envolvem conceitos de porcentagem, juros simples e juros compostos. No entanto, antes disso, discorre-se sobre os conceitos de taxa percentual, juros e montante. Argumenta-se que Porcentagem se refere a números fracionários e que Juro é a remuneração que se recebe de uma instituição, ou a ela se paga, em relação ao capital, bem como que a quantidade de dinheiro que se tem após o recebimento do juro é chamada de Montante. Ao aplicar um capital por determinado tempo, a uma certa taxa de juro, o montante pode crescer ou diminuir segundo dois regimes: o de Juros Simples e o de Juros Compostos.

4.3.1 Porcentagem - Descontos e Acréscimos

Taxa percentual ou porcentagem é a razão entre um número real P e o número 100 que pode ser indicada por $\frac{P}{100} = P\%$.

Ao expor as definições para os alunos, o professor tem a preocupação de argumentar que a porcentagem se trata da parte de um todo e, por se tratar de uma divisão por 100, é necessário entender representações diferentes de números racionais como fração, decimal e taxa percentual.

Propõe-se, a seguir, um problema de programação que emprega a experiência em acréscimos e descontos sobre determinados valores monetários.

Problema 9 - Desenvolva um programa que calcule um desconto e um acréscimo percentual de um determinado valor em reais.

Ao codificar um programa que resolva esse problema, o aluno deve decidir como pretende representar a porcentagem na programação, seja por meio da taxa percentual utilizando o símbolo %, ou por meio de um número decimal. Caso considere a primeira opção mais apropriada para a situação, então, o aluno deve se atentar à necessidade de dividir o valor utilizado por 100.

Uma codificação sugestiva seria utilizar de uma interatividade para solicitar ao usuário o preço e valor da taxa percentual em que se pretende promover o desconto ou o aumento, deixando claro que o usuário não deve fornecer a porcentagem representando-a no formato de um número decimal. Dessa forma, o aluno pode definir uma variável que calcule separadamente o valor fracionário a ser subtraído ou adicionado ao preço fornecido, obtendo, assim, o novo preço requisitado.

Logo mais, apresenta-se uma proposta de como esse programa poderia ser desenvolvido, onde expõe-se, abaixo, tanto sua execução como sua codificação.

```
1  PORCENTAGEM
2  = = = = = = = = = =
3  Podemos lhe informar o novo preço de uma mercadoria após um DESCONTO
       PERCENTUAL!
4  Qual o preço atual do produto? Digite o preço: R$ 240
5  Qual o porcentagem do desconto? Digite o valor (%):  15
6  O produto que custava R$ 240.00, agora, após o desconto de 15.0%, vai custar
       R$ 204.00.
7
8  Além disso, podemos lhe informar o novo preço de uma mercadoria após um
       AUMENTO PERCENTUAL!
9  Qual o preço atual do produto? Digite o preço: R$ 240
10 Qual o porcentagem do aumento? Digite o valor (%): 15
```

```
11 O produto que custava R$ 240.00, agora, após o aumento de 15.0%, vai custar R$
       276.00.
```

Código-fonte 4.17 – Porcentagem (Execução)

```
1  print('\nPORCENTAGEM')
2  print('-=-'*8)
3
4  #DESCONTO
5  print('Podemos lhe informar o novo preço de uma mercadoria após um DESCONTO
       PERCENTUAL!')
6  preço = float(input('Qual o preço atual do produto? Digite o preço: R$ '))
7  porcentagem = float(input('Qual o porcentagem do desconto? Digite o valor (%):
        '))
8
9  p = porcentagem/100
10 novo = preço - preço * p
11 print('O produto que custava R$ {:.2f}, agora, após o desconto de {}%, vai
       custar R$ {:.2f}.'.format(preço, porcentagem, novo))
12
13 #AUMENTO
14 print('\nAlém disso, podemos lhe informar o novo preço de uma mercadoria após
       um AUMENTO PERCENTUAL!')
15 preço = float(input('Qual o preço atual do produto? Digite o preço: R$ '))
16 porcentagem = float(input('Qual o porcentagem do aumento? Digite o valor (%):
       '))
17
18 p = porcentagem/100
19 novo = preço + preço * p
20 print('O produto que custava R$ {:.2f}, agora, após o aumento de {}%, vai
       custar R$ {:.2f}.'.format(preço, porcentagem, novo))
```

Código-fonte 4.18 – Porcentagem (Codificação)

Um caminho alternativo nessa codificação seria solicitar ao usuário para fornecer a porcentagem no formato decimal, por exemplo `porcentagem = float(input('Qual o porcentagem do desconto? Digite o valor (em decimal): '))`, dessa forma, o novo preço seria calculado meramente pelo comando representado por `novo = preço - preço*porcentagem` sem a necessidade de definir a variável `p = porcentagem/100`.

Isso exige do aluno e do usuário a habilidade na representação e operacionalização com números racionais.

4.3.2 Juros Simples

Nesta subseção pretende-se abordar um problema de programação que faça uso de conceitos relacionados ao cálculo de Juros Simples. Antes da apresentação do enunciado do problema de programação, relembra-se, aqui, que no regime de Juros Simples, o juro incide apenas sobre o capital investido, e o montante resgatado nesse regime depende de três fatores: do capital, do tempo de aplicação e da taxa de juros. Quando um capital C é aplicado durante t unidades de tempo e a taxa i de juros, por unidade de tempo, incide apenas sobre o capital inicial, então o juro J é denominado de juros simples e, ao final da aplicação, é calculado por $J = C \cdot i \cdot t$. Como o montante M é definido como o valor final após uma aplicação, então ele é calculado pela soma entre capital e os juros: $M = C + J$.

Além desses conceitos, o aluno deve dominar que, para o cálculo do juro, o tempo e a taxa devem sempre estar descritos segundo o mesmo período de tempo, caso contrário há a necessidade de converter uma delas por meio de uma proporcionalidade. Com isso, o professor conclui, com os alunos, que para calcular os juros simples sobre determinado valor monetário é necessário se trabalhar com a mesma unidade de tempo: por exemplo se a taxa percentual usada for $a.m.$ – ao mês, o tempo também deve ser usado em meses, sendo necessário realizar alguma conversão eventualmente.

Geralmente o professor preocupa-se em debater profundamente com os alunos sobre esses conceitos relacionados ao tempo e argumenta que para cálculos dessa natureza considera-se o ano comercial com 360 dias, sendo 12 meses de 30 dias cada. Além disso, o professor expõe representações diferentes para as taxas, como $a.d.$ – ao dia, $a.m.$ – ao mês, $a.b.$ – ao bimestre, $a.t.$ – ao trimestre, $a.s.$ – ao semestre, $a.a.$ – ao ano; inclusive conceitua e exemplifica taxas equivalentes, por exemplo apresentando taxas do tipo 2% ao mês e 24% ao ano; 1% ao bimestre e 3% ao semestre; 5% ao trimestre e 20% ao ano; 2% ao dia e 60% ao mês.

O aluno deve dominar tudo isso, ao desenvolver um programa que resolva um problema como o apresentado a seguir:

> **Problema 10** - Desenvolva um programa que solicite ao usuário o capital, a taxa de juros e o tempo de aplicação ou empréstimo a juros simples para, então, calcular o montante após o tempo informado.

O enunciado desse problema poderia ser menos detalhista e objetivo como “Desenvolva um programa que informe o montante de uma aplicação monetária a regime de juros simples”. No entanto, o enunciado do problema norteia o aluno programador para

a utilização de uma interatividade com o usuário. Tais comandos interativos devem-se atentar à unidade de tempo solicitada, deixando claro ao usuário qual a unidade utilizada. Exibe-se, abaixo, uma solução para o problema apresentado.

```
1 JUROS SIMPLES
2 -=-=-=-=-=-=-=-=-=-=-
3 Informe o capital aplicado/emprestado: R$ 2500
4 Informe a taxa percentual (% a.m.): 5
5 Informe o tempo (em meses) da/o aplicação/empréstimo: 12
6
7 Após aplicação/empréstimo do capital de R$ 2500.0 a uma taxa de 5.0% a.m., por
      um período de 12.0 meses a regime de Juros Simples,
8 os juros a serem pagos devem ser de R$ 1500.0.
9 Dessa forma, o montante será igual a R$ 4000.0.
```

Código-fonte 4.19 – Juros Simples (Execução)

```
1 print('\nJUROS SIMPLES')
2 print('-=-'*10)
3
4 capital = float(input('Informe o capital aplicado/emprestado: R\$ '))
5 taxa = float(input('Informe a taxa percentual (% a.m.): '))
6 tempo = float(input('Informe o tempo (em meses) da/o aplicação/empréstimo: '))
7
8 juros = capital * (taxa/100) * tempo
9 montante = capital + juros
10
11 print('\nApós aplicação/empréstimo do capital de R$ {} a uma taxa de {}% a.m.,
      por um período de {} meses a regime de Juros Simples,\nos juros a serem
     pagos devem ser de R$ {}.'.format(capital, taxa, tempo, juros))
12 print('Dessa forma, o montante será igual a R$ {}.'.format(montante))
```

Código-fonte 4.20 – Juros Simples (Codificação)

Aponta-se que não há a necessidade de dividir por 100 caso a taxa seja solicitada na sua representação decimal. Desse modo, deixando isso claro para o usuário, o comando `juros = capital * taxa * tempo`, nesse caso, se torna apropriado.

4.3.3 Juros Compostos

Apresenta-se, nesta subseção, um problema de programação que direcione aos procedimentos para o cálculo de Juros Compostos. Ao realizar exposições sobre Matemática Financeira, o professor conceitua o regime de Juro Composto definindo-o como o rendimento obtido ao final de cada aplicação que é incorporado ao capital inicial, dando origem assim a um novo montante. A partir daí, calcula-se o juro sempre sobre o resultado da aplicação anterior, o que corriqueiramente chama-se de "juro sobre juro". Dessa forma, se um capital C é aplicado em regime de juro composto durante t unidades de tempo, à taxa constante i por unidade de tempo, então o montante acumulado é dado por $M = C \cdot (1 + i)^t$.

Ressalta-se que, similar ao cálculo do juro simples, para aplicar a fórmula do montante acumulado a juro composto, as variáveis i e t devem estar relacionadas à mesma unidade de tempo.

Adiante, sugere-se um problema de programação envolvendo os conceitos de juros compostos. Na sequência, exibe-se uma sugestão para a sua solução.

Problema 11 - Desenvolva um programa que solicite ao usuário o capital, a taxa de juros e o tempo de aplicação ou empréstimo a juros compostos para, então, calcular o montante após o tempo informado.

```
JUROS COMPOSTOS
-=--=--=--=--=--=--=--=--=--=-
Informe o capital aplicado/emprestado: R$ 1200
Informe a taxa percentual (% a.m.): 2
Informe o tempo (em meses) da/o aplicação/empréstimo: 8

Após aplicação/empréstimo do capital de R\$ 1200.00 a uma taxa de 2.0% a.m.
    por um período de 8.0 meses a regime de Juros Compostos,
o montante ser pago será igual a R$ 1405.99.
```

Código-fonte 4.21 – Juros Compostos (Execução)

```
print('\nJUROS COMPOSTOS')
print('-=-'*10)

capital = float(input('Informe o capital aplicado/emprestado: R$ '))
taxa = float(input('Informe a taxa percentual (% a.m.): '))
tempo = float(input('Informe o tempo (em meses) da/o aplicação/empréstimo: '))

montante = capital*((1+taxa/100)**tempo)
```

```
9
10 print('\nApós aplicação/empréstimo do capital de R$ {:.2f} a uma taxa de {}% a
       .m. por um período de {} meses a regime de Juros Compostos,'.format(
       capital, taxa, tempo))
11 print('o montante ser pago será igual a R$ {:.2f}.'.format(montante))
```

Código-fonte 4.22 – Juros Compostos (Codificação)

4.4 Triângulos

Nesta seção serão abordados problemas de programação que utilizam conceitos relacionados à condição de existência de um triângulo, bem como à classificação de triângulos de acordo com a medida dos lados e com a medida dos ângulos. Além disso, explora-se, na sequência, um problema relacionado ao cálculo da área de um triângulo por meio da fórmula de Heron.

Os quatro problemas apresentados sugerem que a programação solicite ao usuário as três medidas de um triângulo e, a partir disso, analise e imprima a sua existência, a sua classificação e a sua área. De fato, isso é possível dispondo apenas das medidas dos três lados de um triângulo.

Primeiro, quanto à condição de existência, argumenta-se que nem sempre três medidas podem formar um triângulo. Para que um triângulo exista é necessário e suficiente que a medida de cada um dos lados seja menor do que a soma dos outros dois lados. Segundo, os triângulos podem ser classificados quanto às medidas dos seus lados como escaleno (três medidas diferentes), isósceles (duas medidas iguais) e equilátero (três medidas são iguais). Terceiro, a classificação de um triângulo quanto à medida dos ângulos também pode ser determinada a partir da medida dos três lados por meio da Síntese de Clairaut, além disso, pode-se dizer que um triângulo é acutângulo caso os seus três ângulos sejam agudos (medidas entre 0° e 90°), um triângulo é classificado como retângulo caso apresente um ângulo reto (medida igual a 90°) e, ainda, classificado como obtusângulo caso apresente um ângulo obtuso (medida entre 90° e 180°). Por fim, a área de um triângulo pode ser obtida a partir das medidas dos seus três lados por meio da relação estabelecida por Heron envolvendo o seu semiperímetro.

Formalmente, sendo a, b e c as medidas dos lados de um triângulo, tem-se que:

- Para que um triângulo exista é necessário e suficiente que $a < b + c$, $b < a + c$ e $c < a + b$;
- Se $a = b = c$, então o triângulo é classificado como equilátero; se $a \neq b \neq c$, então o triângulo é escaleno; se $a = b \neq c$ ou $a \neq b = c$ ou $a = c \neq b$, então o triângulo

é classificado como isósceles.

- A Síntese de Clairaut afirma que, sendo a a medida do maior lado do triângulo, se $a^2 < b^2 + c^2$, então o triângulo é classificado como acutângulo; se $a^2 = b^2 + c^2$, então o triângulo é retângulo; se $a^2 > b^2 + c^2$, então o triângulo é classificado como obtusângulo.

- A área pode ser obtida por meio de uma relação estabelecida por Heron: $A = \sqrt{p(p-a)(p-b)(p-c)}$, onde $p = \dfrac{a+b+c}{2}$.

Logo mais, apresenta-se quatro problemas de programação que exploram esses conceitos. São exibidos, de forma pragmática, os enunciados dos problemas e, na sequência, a execução e a codificação de programas sugestivos que solucionam os problemas apresentados. Separadamente, os problemas tratam da condição de existência de um triângulo (Seção 4.4.1), da classificação quanto a medida dos lados (Seção 4.4.2), da classificação quanto a medida dos ângulos (Seção 4.4.3) e do cálculo da área por meio da fórmula de Heron (Seção 4.4.4). No entanto, existe a possibilidade de codificar um programa que unifique os quatro apresentados; sendo assim, desafia-se o leitor a pensar e elaborar um único programa que informe a existência, as duas classificações e a área de um triângulo. O Apêndice A apresenta um Código-fonte que faz essa unificação.

4.4.1 Condição de existência de um triângulo

Problema 12 - Crie um programa que solicite ao usuário três medidas e informe se tais medidas podem ou não formar um triângulo.

```
1  CONDIÇÃO DE EXISTÊNCIA DE UM TRIÂNGULO
2  = = = = = = = = = = = = = = = = = = = =
3
4  Para que um triângulo exista, é necessário que cada um dos lados seja menor
       que a soma dos outros dois!
5  Vamos analisar a existência de um triângulo?
6
7  Informe as medidas dos seus lados.
8  Medida do primeiro lado: 3
9  Medida do segundo lado: 2
10 Medida do terceiro lado: 7
11 NÃO! As medidas dos segmentos acima NÃO podem formar um triângulo.
```

Código-fonte 4.23 – Condição de Existência de um Triângulo (Execução)

```
1 print('\nCONDIÇÃO DE EXISTÊNCIA DE UM TRIÂNGULO')
2 print('-=-'*13)
3 print('\nPara que um triângulo exista, é necessário que cada um dos lados seja
      menor que a soma dos outros dois!')
4 print('Vamos analisar a existência de um triângulo?\n')
5 print('Informe as medidas dos seus lados.')
6 m1 = float(input('Medida do primeiro lado: '))
7 m2 = float(input('Medida do segundo lado: '))
8 m3 = float(input('Medida do terceiro lado: '))
9
10 if m1 < m2 + m3 and m2 < m1 + m3 and m3 < m1 + m2:
11    print('SIM! Os segmentos acima podem formar um triângulo!')
12 else:
13    print('NÃO! As medidas dos segmentos acima NÃO podem formar um triângulo.')
```

Código-fonte 4.24 – Condição de Existência de um Triângulo (Codificação)

4.4.2 Classificação dos triângulos quanto à medida dos lados

Problema 13 - Desenvolva um programa que solicite ao usuários três medidas e informe se tais medidas podem formar um triângulo equilátero, isósceles ou escaleno.

```
1 CLASSIFICAÇÃO DE UM TRIÂNGULO QUANTO À MEDIDA DOS LADOS
2 -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
3
4 Vamos analisar a existência de um triângulo e a sua classificação?
5
6 Informe as medidas dos seus lados.
7 Medida do primeiro lado: 4
8 Medida do segundo lado: 4
9 Medida do terceiro lado: 5
10
11 SIM! Os segmentos acima podem formar um triângulo
12 ISÓSCELES!
```

Código-fonte 4.25 – Classificação de um triângulo quanto à medida dos lados (Execução)

```
print('\nCLASSIFICAÇÃO DE UM TRIÂNGULO QUANTO À MEDIDA DOS LADOS')
print('-=-'*13)

print('\nVamos analisar a existência de um triângulo e a sua classificação?')
print('\nInforme as medidas dos seus lados.')
m1 = float(input('Medida do primeiro lado: '))
m2 = float(input('Medida do segundo lado: '))
m3 = float(input('Medida do terceiro lado: '))

if m1 < m2 + m3 and m2 < m1 + m3 and m3 < m1 + m2:
   print('SIM! Os segmentos acima podem formar um triângulo ')
   if m1==m2==m3:
      print('EQUILÁTERO!')
   elif (m1==m2 and m1!=m3) or (m3==m2 and m1!=m3) or (m1==m3 and m2!=m3):
      print('ISÓSCELES!')
   else:
      print('ESCALENO!')
else:
   print('NÃO! As medidas dos segmentos acima NÃO podem formar um triângulo.')
```

Código-fonte 4.26 – Classificação de um triângulo quanto à medida dos lados (Codificação)

4.4.3 Classificação dos triângulos quanto a medida dos ângulos

Problema 14 - Desenvolva um programa que solicite ao usuários três medidas e informe se elas podem formar um triângulo acutângulo, retângulo ou obtusângulo.

```
CLASSIFICAÇÃO QUANTO À MEDIDA DOS ÂNGULOS
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

Vamos analisar a existência de um triângulo e a sua classificação?
Informe as medidas dos seus lados.

Medida do primeiro lado: 3
Medida do segundo lado: 4
Medida do terceiro lado: 5
SIM! Os segmentos acima podem formar um triângulo
RETÂNGULO!
```

Código-fonte 4.27 – Classificação de um triângulo quanto à medida dos ângulos (Execução)

```
print('\nCLASSIFICAÇÃO QUANTO À MEDIDA DOS ÂNGULOS')
print('-=-'*13)

print('\nVamos analisar a existência de um triângulo e a sua classificação?')
print('Informe as medidas dos seus lados.\n')

m1 = int(input('Medida do primeiro lado: '))
m2 = int(input('Medida do segundo lado: '))
m3 = int(input('Medida do terceiro lado: '))

#Definindo a sequência (m1 > m2 > m3):
maior = m1
meio = m2
menor = m3

#Definindo o maior e o menor lado:
if (m2 <= m3 <= m1):
    maior = m1
    meio = m3
    menor = m2

if (m2 <= m1 <= m3):
    maior = m3
    meio = m1
    menor = m2

if (m1 <= m2 <= m3):
    maior = m3
    meio = m2
    menor = m1

if (m1 <= m3 <= m2):
    maior = m2
    meio = m3
```

```
35     menor = m1
36
37 if (m3 <= m1 <= m2):
38     maior = m2
39     meio = m1
40     menor = m3
41
42 if (m1 <= m2 <= m3):
43     maior = m3
44     meio = m2
45     menor = m1
46
47 if maior < menor + meio:
48     print('SIM! Os segmentos acima podem formar um triângulo ')
49     if maior**2 < menor**2 + meio**2:
50         print('ACUTÂNGULO!')
51     elif maior**2 > menor**2 + meio**2:
52         print('OBTUSÂNGULO!')
53     elif maior**2 == menor**2 + meio**2:
54         print('RETÂNGULO!')
55 else:
56     print('NÃO! As medidas dos segmentos acima NÃO podem formar um triângulo.')
```

Código-fonte 4.28 – Classificação de um triângulo quanto à medida dos ângulos (Codificação)

4.4.4 Área de um triângulo - Fórmula de Heron

Problema 15 - Desenvolva um programa que solicite ao usuários três medidas de um triângulo e, a partir delas, calcule a sua ÁREA.

```
1 HERON — ÁREA DE UM TRIÂNGULO
2 -=-=-=-=-=-=-=-=-=-=-=-
3 Podemos calcular a área de um triângulo a partir da medida de seus três lados.
4 Informe as medidas dos seus lados.
5
6 Medida do primeiro lado: 9
7 Medida do segundo lado: 8
8 Medida do terceiro lado: 7
```

```
9
10 A área do triângulo é 26.83.
```

Código-fonte 4.29 – Área de um triângulo - Fórmula de Heron (Execução)

```
1  print('\nHERON — ÁREA DE UM TRIÂNGULO')
2  print('—=—'*10)
3
4  print('Podemos calcular a área de um triângulo a partir da medida de seus três
       lados.')
5  print('Informe as medidas dos seus lados.\n')
6
7  a = float(input('Medida do primeiro lado: '))
8  b = float(input('Medida do segundo lado: '))
9  c = float(input('Medida do terceiro lado: '))
10
11 p = (a + b + c)/2
12 A = (p*(p—a)*(p—b)*(p—c))**(1/2)
13
14 if a < b + c and b < a + c and c < a + b:
15    print('\nA área do triângulo é {:.2f}.'.format(A))
16 else:
17    print('\nATENÇÃO! As medidas dos segmentos acima NÃO podem formar um triâ
       ngulo.')
```

Código-fonte 4.30 – Área de um triângulo - Fórmula de Heron (Codificação)

5 Sequência Didática: Linguagem Matemática e Linguagem de Programação Python - Funções Polinomiais definidas por mais de uma sentença.

Neste capítulo, sugere-se, para a disciplina de Matemática, uma Sequência Didática para o Ensino Médio que conduz a manipulação e solidificação dos conceitos e aplicações relacionados às Funções Polinomiais definidas por mais de uma sentença, por meio da Linguagem de Programação Python.

Uma Sequência Didática é um valioso instrumento de planejamento de ensino para professores de qualquer disciplina. Assim como os projetos didáticos e as atividades permanentes ou ocasionais, as sequências didáticas são uma modalidade organizativa de ensino, cuja finalidade é expressar uma sequência de etapas que tem o objetivo de sistematizar um conteúdo e promover a assimilação de um saber. Sequência Didática é "um conjunto de atividades ordenadas, estruturadas e articuladas para a realização de certos objetivos educacionais, que têm um princípio e um fim conhecido tanto pelos professores como pelos alunos". (ZABALA, 1998)

Dessa forma, apresenta-se, mais adiante, a sua estruturação e espera-se que seja de grande valia para nortear, auxiliar e inspirar professores de matemática que objetivam o uso da Linguagem de Programação Python em suas aulas e, inclusive, anseiam relacioná-la com a Linguagem Matemática e apoiar o desenvolvimento do letramento matemático de seus alunos.

Propõe-se que esta Sequência Didática seja vivenciada em sala de aula logo após o estudo das Funções Polinomiais definidas por mais de uma sentença, sendo necessário toda a assimilação da Linguagem Matemática deste conteúdo para, então, executar os seus direcionamentos. Sugere-se que, nas primeiras aulas desta Sequência Didática, seja promovido aos alunos um tutorial acerca do manuseio da Linguagem de Programação Python, em conformidade com o que expõe o Capítulo 3 deste trabalho. Nas aulas subsequentes, pretende-se apresentar problemas matemáticos que irão aumentando o nível de dificuldade de compreensão e programação.

Como o conteúdo de Funções é apontado pelo currículo de Matemática nos três anos do Ensino Médio, então propõe-se que estas atividades possam ser trabalhadas em qualquer uma das séries desta etapa da educação básica. Alerta-se que, antes da realização destas atividades, é necessário que os alunos já devam ter o conhecimento

das definições, linguagens e representações das funções, sobretudo estudado a estética da representação de funções que são definidas por mais de uma sentença. Recorda-se que uma função é definida por mais de uma sentença quando cada uma delas está associada a um subdomínio $D_1, D_2, D_3, ..., D_n$ e a união destes n subconjuntos forma o domínio D da função original, ou seja, cada domínio D_i $(1 \leq i \leq n)$ é um subconjunto de D.

A Sequência Didática proposta, aqui, consiste em uma prossecução de 9 aulas, onde cada uma das aulas é organizada por três tópicos: Tempo estimado, Desenvolvimento e Resultados esperados dos alunos. A Aula 1 (Subseção 5.1.1) apresenta a Linguagem de Programação Python e alguns IDEs que permitem o seu manuseio. A Aula 2 (Subseção 5.1.2) introduz a programação em Python promovendo um tutorial e um exercício sobre comandos e funções primordiais. A Aula 3 (Subseção 5.1.3) dá sequência ao tutorial apresentado e exercita novos comandos e funções necessárias para a codificação dos programas dessa Sequencia Didática. A Aula 4 (Subseção 5.1.4) diagnostica os conhecimentos prévios e as lacunas dos alunos; a Aula 5 (Subseção 5.1.5) propõe o primeiro problema a ser programado, o qual determina o salário de um vendedor em função da quantidade de produtos vendidos; a Aula 6 (Subseção 5.1.6), por sua vez, propõe o segundo problema, onde é observado a indicação da codificação de um programa que fornece o salário de um metalúrgico em função da quantidade semanal de horas trabalhadas; na Aula 7 (Subseção 5.1.7), o problema exibido analisa o valor a ser pago pelo consumo residencial de água; na sequência, a Aula 8 (Subseção 5.1.8) apresenta um problema que propõe a elaboração de um programa que informa ao usuário o valor, em reais, do imposto de renda a ser pago em função da renda anual de um cidadão contribuinte; por fim, a Aula 9 (Subseção 5.1.9) sugere a realização de uma avaliação sobre o que foi vivenciado no decorrer da sequência didática.

Em todas as aulas, o desenvolvimento é composto por dois momentos, onde destaca-se que, para as Aulas 5, 6, 7 e 8, o primeiro momento analisa a Linguagem Matemática do problema e o segundo mo mento explora a Linguagem de Programação. Nas demais aulas, os dois momentos são organizados de outra forma, onde, nas Aulas 1, 2 e 3, o primeiro momento explora o tutorial e um estudo de conceitos, enquanto o segundo momento aborda a prática dos conceitos vivenciados; nas Aulas 4 e 9 sãos realizadas avaliações.

Ao término da exposição da estrutura dessa Sequência Didática, são realizadas algumas ponderações que podem sugerir orientações adicionais relativas à execução desta.

5.1 Designação da Sequência Didática

Título: Linguagem Matemática e Linguagem de Programação Python - Funções Polinomiais definidas por mais de uma sentença.

Professor: João Evayr de Souza
Turmas: Ensino Médio (1º, 2º e 3º anos).
Duração: 18 horas-aulas.

Área do conhecimento: Matemática e Suas Tecnologias.
Componente curricular: Matemática.
Campo/Eixo: Álgebra e Funções.
Objeto de Conhecimento: Funções polinomiais definidas por mais de uma sentença.

Habilidades:

- (EM13MAT405) Utilizar conceitos iniciais de uma linguagem de programação na implementação de algoritmos escritos em linguagem corrente e/ou matemática.
- (EM13MAT302) Construir modelos empregando as funções polinomiais de 1º ou 2º graus, para resolver problemas em contextos diversos, com ou sem apoio de tecnologias digitais.
- (EM13MAT404) Analisar funções definidas por uma ou mais sentenças (tabela do Imposto de Renda, contas de luz, água, gás etc.), em suas representações algébrica e gráfica, identificando domínios de validade, imagem, crescimento e decrescimento, e convertendo essas representações de uma para outra, com ou sem apoio de tecnologias digitais.
- (EM13MAT401) Converter representações algébricas de funções polinomiais de 1º grau em representações geométricas no plano cartesiano, distinguindo os casos nos quais o comportamento é proporcional, recorrendo ou não a softwares ou aplicativos de álgebra e geometria dinâmica.

Objetivos:

- Apresentar a Linguagem de Programação Python como um recurso digital e tecnológico para trabalhar Funções Polinomiais definidas por mais de uma sentença.
- Promover um tutorial sobre comandos e funções primordiais para o manuseio da Linguagem de Programação Python.

- Sugerir problemas que utilizem a linguagem de programação para codificar programas que usem conceitos sobre tópicos específicos de Matemática do Ensino Médio.
- Dominar as diferentes formas de representação de uma função e identificar as relações entre elas.
- Operar com funções e sua classificação, de acordo com seu comportamento.
- Reconhecer a representação algébrica de funções.
- Resolver e elaborar problemas envolvendo funções.
- Modelar e resolver problemas que envolvem variáveis socioeconômicas ou técnico-científicas, usando representações algébricas.
- Identificar representações algébricas que expressem a relação entre grandezas.
- Compreender a ideia de grandezas formadas por relações entre outras grandezas e resolver e elaborar problemas envolvendo essas ideias.
- Resolver situação-problema cuja modelagem envolva conhecimentos algébricos.
- Reconhecer função como modelo matemático para o estudo das variações entre grandezas do mundo natural ou social.
- Construir e/ou analisar gráficos associados a uma situação do mundo natural ou social.

Recursos:

- Quadro branco e lápis;
- Projetor Multimídia e Notebooks;
- Laboratório de Informática;
- Computadores e Celulares.

5.1.1 Aula 1 - Apresentação da Linguagem Python e dos IDEs

- Tempo estimado: 2 aulas de 50 min.

  - Momento I: 1 aula (50 min)
  - Momento II: 1 aula (50 min)

- Desenvolvimento:

  - Momento I: Apresenta-se para a turma a Linguagem de Programação Python, expondo sua história, suas características e suas potencialidades. Além disso, orientar para a correta instalação e utilização de três IDEs (Ambientes de Desenvolvimento Integrados): PyCharm, Spyder e Pydroid.

  - Momento II: Apresenta-se a função `print()` e propõe-se para a turma, como forma de exercício, a codificação e a execução do um primeiro comando: imprimir a mensagem "`Seja bem-vindo ao ambiente de programação!`" nos três IDEs sugeridos. A partir disso, fazer comparações sobre as diferenças do manuseio de cada um dos IDEs.

- Resultados esperados dos alunos:

  - Momento II: Espera-se que os alunos analisem as diferenças no manuseio dos três IDEs e, a partir disso, elejam um deles para utilizar no decorrer das aulas desta Sequência Didática.

5.1.2 Aula 2 - Introdução à Programação com Python: Estudo sobre as Variáveis, Comentários, Interatividade com o Usuário e Operadores Básicos.

- Tempo estimado: 2 aulas de 50 min.

  - Momento I: 1 aula (50 min)
  - Momento II: 1 aula (50 min)

- Desenvolvimento:

  - Momento I: Promove-se para a turma um tutorial sobre comandos e funções primordiais para a programação com Python. Uma aula conceitual, expositiva e demonstrativa sobre os três tipos de Variáveis (`str, int e float`), sobre como inserir comentários na codificação, sobre como criar uma interatividade com o usuário por meio a função `input()` e sobre os Operadores Básicos (aritméticos, comparação e lógicos). Sugere-se a utilização de um projetor multimídia para expor os conceitos e demonstrações.

  - Momento II: Propõe-se para a turma um exercício que possibilite a prática e o uso dos conceitos estudados no Momento I. Apresenta-se, a seguir, uma sugestão de questões para este exercício.

Tabela 4 – Aula 2: Exercícios

AULA 2 - EXERCÍCIOS
QUESTÃO 1 - Elabore um programa que solicite o nome e o sobrenome do usuário. Ao final, o programa deverá apresentar o nome completo do usuário na tela. QUESTÃO 2 - Elabore um programa que solicite três números inteiros do usuário e imprima a soma destes. QUESTÃO 3 - Desenvolva um programa que calcule as médias aritmética e geométrica entre quatro números racionais fornecidos pelo usuário.

Fonte: Elaborado pelo autor.

- Resultados esperados dos alunos:

  - Momento II:

```
1  nome = input('Digite seu nome: ')
2  sobrenome = input('Digite seu sobrenome: ')
3  print('Seu nome completo é {} {}.'.format(nome, sobrenome))
```

Código-fonte 5.1 – Aula 2: Exercício (Questão 1)

```
1 print('VAMOS CALCULAR A SOMA ENTRE TRÊS NÚMEROS INTEIROS?\n')
2 n1 = int(input('Digite o primeiro número: '))
3 n2 = int(input('Digite o segundo número: '))
4 n3 = int(input('Digite o terceiro número: '))
5 print('A soma entre os três números informados é ', n1+n2+n3.)
```

Código-fonte 5.2 – Aula 2: Exercício (Questão 2)

```
1  print('VAMOS CALCULAR A MÉDIA ARITMÉTICA E GEOMÉTRICA ENTRE TRÊS NÚMEROS
       ?\n')
2  n1 = float(input('Digite um número: '))
3  n2 = float(input('Digite outro número: '))
4  n3 = float(input('Digite outro número: '))
5  n4 = float(input('Digite outro número: '))
6
7  ma = (n1+n2+n3+n4)/4
8  mg = (n1*n2*n3*n4)**(1/4)
9
10 print('\nCom base nos três números informados, a média aritmética é {:.2
       f} e a média geométrica é {:.2f}.'.format(ma, mg))
```

Código-fonte 5.3 – Aula 2: Exercício (Questão 3)

5.1.3 Aula 3 - Programação com Python: Estudo sobre Indentação e sobre os Condicionais `if`, `else` e `elif`.

- Tempo estimado: 2 aulas de 50 min.

  - Momento I: 1 aula (50 min)
  - Momento II: 1 aula (50 min)

- Desenvolvimento:

  - Momento I: Proporciona-se uma aula conceitual, expositiva e demonstrativa sobre o uso e a importância da Indentação e sobre os Condicionais `if`, `else` e `elif`. Sugere-se a utilização de um projetor multimídia para expor os conceitos e demonstrações.

  - Momento II: Propõe-se para a turma um exercício que possibilite a prática e o uso dos conceitos estudados no Momento I. Apresenta-se, a seguir, uma sugestão de questões para este exercício.

Tabela 5 – Aula 3: Exercícios

AULA 3 - EXERCÍCIOS

QUESTÃO 1 - Elabore um programa que solicite a idade do usuário. Ao final, o programa deverá apresentar o se o usuário é criança, adolescente, adulto ou idoso.
QUESTÃO 2 - Elabore um programa que solicite um número inteiro ao usuário e informe se este é par ou ímpar.
QUESTÃO 3 - O Índice de Massa Corporal (IMC) é utilizado para mensurar o peso ideal de uma pessoa. O IMC é calculado por meio da divisão entre a massa e o quadrado da altura de uma pessoa. Desenvolva um programa que solicite a massa (em kg) e a altura (em m) do usuário para, então, calcular o seu IMC e classificar este resultado de acordo com a regra a seguir:
- $IMC < 17$: Magreza Moderada
- $17 \leq IMC < 18,5$: Magreza leve
- $18,5 \leq IMC < 25$: Peso normal
- $25 \leq IMC < 30$: Sobrepeso
- $30 \leq IMC < 35$: Obesidade
- $35 \leq IMC < 40$: Obesidade severa
- $IMC \geq 40$: Obesidade mórbida

Fonte: Elaborado pelo autor.

- Resultados esperados dos alunos:

  - Momento II:

```
idade = int(input('Qual a sua idade? '))

if idade < 13:
   print('Você é uma criança!')
elif 13 <= idade < 18:
   print('Você é um adolescente!')
elif 18 <= idade < 60:
   print('Você é um adulto!')
else:
   print('Você é um idoso!')
```

Código-fonte 5.4 – Aula 3: Exercício (Questão 1)

```
print('\nPosso lhe informar se um número é PAR ou ÍMPAR!')
número = int(input('Diga-me um número qualquer: '))

resultado = número % 2

if resultado == 0:
   print('O número {} é PAR'.format(número))
else:
   print('O número {} é ÍMPAR!'.format(número))
```

Código-fonte 5.5 – Aula 3: Exercício (Questão 2)

```
massa = float(input('Qual a sua massa(kg)?  '))
altura = float(input('Qual a sua altura(m)?  '))

imc = massa/(altura**2)

print('Seu IMC é {:.1f} kg/m^2.'.format(imc))
print('RESULTADO:')
if imc < 17:
   print("Magreza moderada.")
elif 17 <= imc < 18.5:
   print("Magreza leve.")
elif 18.5 <= imc < 25:
   print("Saudável.")
elif 25 <= imc < 30:
```

```
15     print("Sobrepeso.")
16 elif 30 <= imc < 35:
17     print("Obesidade.")
18 elif 35 <= imc < 40:
19     print("Obesidade severa.")
20 else:
21     print("Obesidade mórbida.")
```

Código-fonte 5.6 – Aula 3: Exercício (Questão 3)

5.1.4 Aula 4 - Avaliação Diagnóstica

- Tempo estimado: 2 aulas de 50 min.

  - Momento I: 1 aula (50 min.)
  - Momento II: 1 aula (50 min.)

- Desenvolvimento:

  - Momento I: Realização de uma Avaliação Diagnóstica para sondar os conhecimentos dos alunos e diagnosticar as suas habilidades e lacunas acerca do manuseio da linguagem de programação Python, como também das definições, linguagens e representações das funções definidas por mais de uma sentença.

  - Momento II: Discussão coletiva conduzida pelo professor sobre as respostas corretas das questões da Avaliação Diagnóstica. Sugere-se que, a partir do diagnóstico pragmático realizado após a aplicação da avaliação, o professor conduza um debate, permitindo a participação dos alunos, para suprir as deficiências e lacunas referentes ao manuseio da linguagem de programação Python, como também referente à representação das funções definidas por mais de uma sentença.

  Apresenta-se abaixo uma sugestão para esta Avaliação Diagnóstica.

Tabela 6 – Aula 4: Avaliação Diagnóstica

AULA 4 - AVALIAÇÃO DIAGNÓSTICA

SOBRE A LINGUAGEM DE PROGRAMAÇÃO PYTHON

QUESTÃO 1 - Qual a função que se utiliza para imprimir uma mensagem na execução de um programa?

QUESTÃO 2 - Qual a função que se utiliza para promover uma interatividade com o usuário na execução de um programa?

QUESTÃO 3 - Como se realiza comentários na codificação de um programa?

QUESTÃO 4 - Quais são os tipos de variáveis utilizadas no Python e como elas são declaradas?

QUESTÃO 5 - Quais os tipos de operadores básicos utilizados na linguagem de programação?

QUESTÃO 6 - Quais são os operadores aritméticos utilizados no Python?

QUESTÃO 7 - O que é a indentação e qual a sua finalidade?

QUESTÃO 8 - Sobre as estruturas de decisão, como funcionam os condicionais `if, else e elif`?

SOBRE AS FUNÇÕES POLINOMIAIS DEFINIDAS POR MAIS E UMA SENTENÇA

QUESTÃO 9 - Construa o gráfico da função $f : \mathbb{R} \to \mathbb{R}$ definida por:

$$f(x) = \begin{cases} 5, \text{ se } x \leq 3 \\ x + 1, \text{ se } x > 3 \end{cases} .$$

QUESTÃO 10 - A distribuidora de energia elétrica de um estado cobra uma taxa mínima de R$ 10,00 de cada consumidor. Além dessa taxa, cada cliente paga R$ 0,07 por quilowatt-hora pelos primeiros quilowatt-hora consumidos; e R$ 0,10 por quilowatt-hora pelo que ultrapassar 30 kWh de consumo. O valor pago $f(x)$, em real, é dado em função do consumo x, em quilowatt-hora, de cada cliente.

a) Determine a lei de associação entre o valor a ser pago $f(x)$, em reais, com a quantidade x de quilowatt-hora consumido.

b) Represente graficamente esta função.

Fonte: Elaborado pelo autor.

- Resultados esperados dos alunos:

Tabela 7 – Aula 4: Correção da Avaliação Diagnóstica

AVALIAÇÃO DIAGNÓSTICA - Respostas esperadas dos alunos

SOBRE A LINGUAGEM DE PROGRAMAÇÃO PYTHON

QUESTÃO 1 - `print()`

QUESTÃO 2 - `input()`

QUESTÃO 3 - Por meio do símbolo `#`, para uma única linha, e por meio de três aspas (`""" """`) para mais de uma linha.

QUESTÃO 4 - As variáveis são do tipo inteiro (para números inteiros), flutuantes (para números decimais) e strings (para textos e um conjunto de caracteres). Essas variáveis são declaradas respectivamente pelos comandos `int(), float()` e `str()`.

QUESTÃO 5 - Operadores aritméticos, operadores de comparação e operadores lógicos.

QUESTÃO 6 - Adição (`+`), subtração (`-`), multiplicação (`*`), divisão (`/`), divisão inteira (`//`), módulo (`%`) e potenciação (`**`).

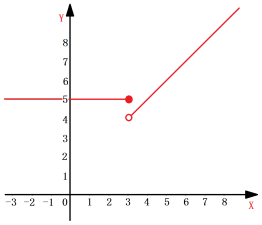
QUESTÃO 7 - É uma distância em relação à margem esquerda em uma linha de codificação, cuja finalidade é a organização e a correta execução de estruturas de códigos.

QUESTÃO 8 - Os condicionais `if, else e elif` são utilizados para definir condições que devem ser ou não executadas conforme a veracidade das afirmações definidas no código. Por exemplo, se uma afirmação for verdadeira então um comando definido por `if` (se) será executado, caso isso não ocorra, então um comando definido por `else` (senão) é executado.

SOBRE AS FUNÇÕES POLINOMIAIS DEFINIDAS POR MAIS E UMA SENTENÇA

QUESTÃO 9 - :

Figura 13 – Avaliação Diagnóstica - Questão 9



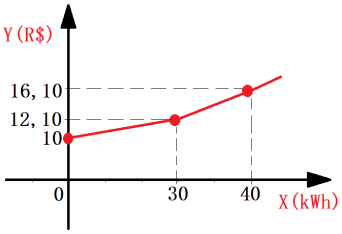
QUESTÃO 10 -

a)

$$f : \mathbb{R}_+ \to \mathbb{R}_+ \quad \text{com} \quad f(x) = \begin{cases} 10 + 0,07x, \text{ se } x \leq 30 \\ 12,10 + 0,10x, \text{ se } x > 30 \end{cases} .$$

b)

Figura 14 – Avaliação Diagnóstica - Questão 10 (item b))



Fonte: Elaborada pelo autor.

5.1.5 Aula 5 - Salário de um vendedor

- Tempo estimado: 2 aulas de 50 min.

  - Momento I: 20 min.
  - Momento II: 80 min.

- Desenvolvimento:

  - Momento I (Linguagem Matemática): Apresenta-se para a turma o enunciado de um problema matemático que envolve uma relação entre duas grandezas por meio de uma função definida por mais de uma sentença. Na sequência, solicita-se a obtenção da lei de formação da função apresentada no problema e, em seguida, a representação do seus conjuntos Domínio, Contradomínio e Imagem.

  - Momento II (Linguagem de Programação): Apresenta-se para turma a solicitação da elaboração de um programa que forneça um valor numérico da função discutida no problema, isto é, um programa que forneça ao usuário o valor do salário que um vendedor deve receber de acordo com a quantidade de produtos vendidos.

  Apresenta-se, abaixo, o enunciado do problema sugerido.

Tabela 8 – Aula 5: Problema - Salário de um vendedor

Problema - Salário de um vendedor.
A) Linguagem Matemática. Certo vendedor tem seu salário mensal calculado da seguinte maneira: ele ganha um valor fixo de R$ 750,00, mais uma comissão de R$ 3,00 para cada produto vendido. Caso ele venda mais de 100 produtos, sua comissão passa a ser de R$ 9,00 para cada produto vendido, a partir do 101º produto vendido. Determine a lei de associação entre o valor do salário $f(x)$, em reais, com a quantidade x de produtos vendidos. Determine, também, os conjuntos Domínio, Contradomínio e Imagem dessa função.
B) Linguagem de Programação. Crie um programa que solicite ao usuário a quantidade de produtos vendidos, para então, fornecer a este o salário apropriado a ser pago para o vendedor.

Fonte: ENEM 2012, adaptado pelo autor (INEP, 2022).

- Resultados esperados dos alunos:

  - Momento I:

  Conjunto Domínio da função: $Dm(f) = \mathbb{N}$.
  Conjunto Contradomínio da função: $CDm(f) = \mathbb{R}$.

Conjunto Imagem da função: $Im(f) = [750, +\infty[$.
O salário $f(x)$ é dado em função de x, para:
1) $0 \leq x \leq 100$, é $f(x) = 750 + 3x$
2) $x \geq 101$, é $f(x) = 750 + 3 \cdot 100 + 9(x - 100) = 150 + 9x$.

Dessa forma, a lei de formação da função pode ser expressa por:

$$f : \mathbb{N} \to \mathbb{R} \ \text{ com } \ f(x) = \begin{cases} 750 + 3x, \text{ se } \ 0 \leq x \leq 100 \\ 150 + 9x, \text{ se } x \geq 101 \end{cases}$$

- Momento II:

```
print('\nFUNÇÕES DEFINIDAS POR MAIS DE UMA SENTENÇA')
print('-='*14)
print('PROBLEMA - Salário de um vendedor.')

x = int(input('\nDigite a quantidade de produtos vendidos: '))

if x < 0:
   print('Quantidade de produtos inválida!')

elif 0 <= x <= 100:
   s = 750 + 3*x
   print('Ao vender {} produtos, o salário do vendedor será R$ {:.2f}.'.
    format(x, s))

elif x >= 101:
   s = 9*x + 150
   print('Ao vender {} produtos, o salário do vendedor será R$ {:.2f}.'.
    format(x, s))
```

Código-fonte 5.7 – Problema - Salário de um vendedor

5.1.6 Aula 6 - Salário de um metalúrgico

- Tempo estimado: 2 aulas de 50 min.

  - Momento I: 30 min.
  - Momento II: 70 min.

- Desenvolvimento:

- Momento I (Linguagem Matemática): Apresenta-se para a turma o enunciado de um problema matemático que envolve uma relação entre duas grandezas por meio de uma função definida por mais de uma sentença. Neste problema, solicita-se a obtenção da lei de formação da função apresentada no problema e, na sequência, a representação do seus conjuntos Domínio, Contradomínio e Imagem.

- Momento II (Linguagem de Programação): Apresenta-se para turma a solicitação da elaboração de um programa que forneça um valor numérico da função discutida no problema, isto é, um programa que forneça ao usuário o valor do salário que um metalúrgico deve receber de acordo com o tempo trabalhado.

Apresenta-se abaixo o enunciado do problema sugerido.

Tabela 9 – Aula 6: Problema - Salário de um metalúrgico.

Problema - Salário de um metalúrgico.

A) Linguagem Matemática.
Um metalúrgico recebe R$ 12,00 por hora trabalhada até o limite de 44 horas semanais, e são acrescidos 30% ao salário/hora em cada hora que excede o limite. Determine a lei de associação entre o valor do salário $f(x)$, em reais, com a quantidade x de horas de trabalho semanal. Determine, também, os conjuntos Domínio, Contradomínio e Imagem dessa função.

B) Linguagem de Programação.
Crie um programa que solicite ao usuário a quantidade de horas de trabalho semanal exercidas por um metalúrgico, para então, fornecer o salário apropriado que deve ser recebido.

Fonte: Elaborado pelo autor.

- Resultados esperados dos alunos:

  - Momento I:

  Conjunto Domínio da função: $Dm(f) = \mathbb{R}_+$.
  Conjunto Contradomínio da função: $CDm(f) = \mathbb{R}_+$.
  Conjunto Imagem da função: $Im(f) = \mathbb{R}_+$.
  O salário $f(x)$ é dado em função de x, para:
  1) $0 \leq x \leq 44$; é $f(x) = 12x$
  2) $x > 44$; é $f(x) = 12 \cdot 44 + (12 + 0,3 \cdot 12).(x - 44) = 528 + 15,6 \cdot (x - 44) = 15,6x - 158,40$.

  Dessa forma, a lei de formação da função pode ser expressa por:

$$f : \mathbb{R}_+ \to \mathbb{R}_+ \quad \text{com} \quad f(x) = \begin{cases} 12x; & \text{se } 0 \leq x \leq 44 \\ 15,6x - 158,40; & \text{se } x > 44 \end{cases}$$

- Momento II:

```
print('\nFUNÇÕES DEFINIDAS POR MAIS DE UMA SENTENÇA')
print('—=—'*14)
print('PROBLEMA — Salário de um metalúrgico.')

x = float(input('\nDigite a quantidade de horas de trabalho semanal: '))

if x < 0:
   print('Tempo inválido!')

elif 0 <= x <= 44:
   s = 12*x
   print('Ao trabalhar por {} horas numa semana, o salário do metalú
    rgico será R$ {:.2f}.'.format(x, s))

elif x > 44:
   s = 15.6*x — 158.40
   print('Ao trabalhar por {} horas numa semana, o salário do metalú
    rgico será R$ {:.2f}.'.format(x, s))
```

Código-fonte 5.8 – Problema - Salário de um metalúrgico

5.1.7 Aula 7 - Consumo residencial de água

- Tempo estimado: 2 aulas de 50 min.

  - Momento I: 40 min.
  - Momento II: 60 min.

- Desenvolvimento:

  - Momento I (Linguagem Matemática): Apresenta-se para a turma o enunciado de um problema matemático que envolve uma relação entre duas grandezas por meio de uma função definida por mais de uma sentença. Neste problema, solicita-se a obtenção da lei de formação da função apresentada no problema e, na sequência, a representação do seus conjuntos Domínio, Contradomínio e Imagem.

- Momento II (Linguagem de Programação): Apresenta-se para turma a solicitação da elaboração de um programa que forneça um valor numérico da função discutida no problema, isto é, um programa que forneça ao usuário o valor do salário que um metalúrgico deve receber de acordo com o tempo trabalhado.

Apresenta-se abaixo o enunciado do problema sugerido.

Tabela 10 – Aula 7: Problema - Consumo residencial de água.

Problema - Consumo residencial de água.

A) Linguagem Matemática.
Uma empresa presta serviço de abastecimento de água em uma cidade. O valor mensal a pagar por esse serviço é determinado pela aplicação de tarifas, por faixas de consumo de água, sendo obtido pela adição dos valores correspondentes a cada faixa.
• Faixa 1: para consumo de até 6 m^3, valor fixo de R$ 12,00;
• Faixa 2: para consumo superior a 6 m^3 e até 10 m³, tarifa de R$ 3,00 por metro cúbico ao que exceder a 6 m^3;
• Faixa 3: para consumo superior a 10 m^3, tarifa de R$ 6,00 por metro cúbico ao que exceder a 10 m^3.
Sabe-se que nessa cidade o consumo máximo de água por residência é de 15 m^3 por mês.
Determine a lei de associação entre o valor $f(x)$, em reais, a ser pago em função da quantidade x de água consumida mensalmente, em m^3. Determine, também, os conjuntos Domínio, Contradomínio e Imagem dessa função.

B) Linguagem de Programação.
Crie um programa que solicite ao usuário a quantidade de água consumida mensalmente, para então, fornecer o valor, em reais, a ser pago pelo consumo.

Fonte: ENEM 2019, adaptado pelo autor (INEP, 2022).

- Resultados esperados dos alunos:

  - Momento I:

  Conjunto Domínio da função: $Dm(f) = [0, 15]$.
  Conjunto Contradomínio da função: $CDm(f) = \mathbb{R}_+$.
  Conjunto Imagem da função: $Im(f) = [12, 54]$.
  O valor f(x), em reais, a ser pago em função da quantidade x de água consumida, em m^3, para:
  1) $0 \leq x \leq 6$; é $f(x) = 12$
  2) $6 < x \leq 10$; é $f(x) = 12 + 3 \cdot (x - 6) = 12 + 3x - 18 = 3x - 6$

3) $10 < x \leq 15$; é $f(x) = 12 + 3 \cdot (10 - 6) + 6(x - 10) = 24 + 6x - 60 = 6x - 36$
A lei de formação da função pode ser expressa por:

$$f : [0, 15] \to \mathbb{R}_+ \quad \text{com} \quad f(x) = \begin{cases} 12; & \text{se } 0 \leq x \leq 6 \\ 3x - 6; & \text{se } 6 < x \leq 10 \\ 6x - 36; & \text{se } 10 < x \leq 15 \end{cases}$$

- Momento II:

```
print('\nFUNÇÕES DEFINIDAS POR MAIS DE UMA SENTENÇA')
print('—=—'*14)
print('PROBLEMA — Consumo residencial de água.')

x = float(input('\nDigite o volume de água consumido (m^3): '))

if x < 0:
   print('Volume inválido!')

elif 0 <= x <= 6:
   p = 12
   print('Ao consumir {} m^3 de água em um mês, o valor a ser pago será
    R$ {:.2f}.'.format(x, p))

elif 6 < x <= 10:
   p = 3*x — 6
   print('Ao consumir {} m^3 de água em um mês, o valor a ser pago será
    R$ {:.2f}.'.format(x, p))

elif 10 < x <= 15:
   p = 6*x — 36
   print('Ao consumir {} m^3 de água em um mês, o valor a ser pago será
    R$ {:.2f}.'.format(x, p))

elif x > 15:
   print ('Limite de consumo excedido! A empresa não fornece um
    abastecimento superior a 15 m^3.')
```

Código-fonte 5.9 – Problema - Consumo residencial de água

5.1.8 Aula 8 - Imposto de Renda (cálculo anual)

- Tempo estimado: 2 aulas de 50 min.

  - Momento I: 40 min.

  - Momento II: 60 min.

- Desenvolvimento:

  - Momento I (Linguagem Matemática): Apresenta-se para a turma o enunciado de um problema matemático que envolve uma relação entre duas grandezas por meio de uma função definida por mais de uma sentença. Neste problema, solicita-se a obtenção da lei de formação da função apresentada no problema e, na sequência, a representação do seus conjuntos Domínio, Contradomínio e Imagem.

  - Momento II (Linguagem de Programação): Apresenta-se para turma a solicitação da elaboração de um programa que forneça um valor numérico da função discutida no problema, isto é, um programa que forneça ao usuário o valor, em reais, do imposto de renda a ser pago por um contribuinte em função da sua renda.

  Apresenta-se a seguir o enunciado do problema sugerido.

Tabela 11 – Aula 8: Problema - Imposto de Renda (cálculo anual).

Problema - Imposto de Renda (cálculo anual).

A) Linguagem Matemática.
No Brasil, os impostos são arrecadados pela Secretaria da Receita Federal e são aplicados na manutenção da estrutura pública e em políticas sociais, econômicas e culturais do Estado. O Imposto de Renda (IR) é o valor que um cidadão contribuinte paga, anualmente, sobre a sua renda adquirida. A tabela progressiva para o cálculo anual do Imposto de Renda de Pessoa Física, representada abaixo, exibe que esse tipo de imposto é calculado em função da renda de cada cidadão. Os valores expostos nesta tabela estão em conformidade com a Instrução Normativa da RFB definida em 2017.

Figura 15 – Imposto de Renda (cálculo anual)

Receita Federal

Base de cálculo (R$)	Alíquota (%)	Parcela a deduzir do IRPF (R$)
Até 22.847,76	-	-
De 22.847,77 até 33.919,80	7,5	1.713,58
De 33.919,81 até 45.012,60	15	4.257,57
De 45.012,61 até 55.976,16	22,5	7.633,51
Acima de 55.976,16	27,5	10.432,32

Fonte: Receita Federal do Brasil (BRASIL, 2017).

Por exemplo, uma pessoa que recebeu R$ 30.000,00, em um determinado ano, deverá pagar R$ 536,42 de imposto de renda no ano subsequente, conforme exibem os cálculos:

$$7,5\% \cdot 30.000 - 1.713,58 = 2.250 - 1713,58 = 536,42.$$

Dessa forma, determine a lei de formação que relaciona o imposto de renda anual $f(x)$ a se pagar em função da renda anual x de um cidadão. Determine, também, os conjuntos Domínio, Contradomínio e Imagem dessa função.

B) Linguagem de Programação.
Crie um programa que informe ao usuário o valor do Imposto de Renda a ser pago em função da sua renda anual.

Fonte: Elaborado pelo autor.

- Resultados esperados dos alunos:

  - Momento I:

  Conjunto Domínio da função: $Dm(f) = \mathbb{R}_+$.
  Conjunto Contradomínio da função: $CDm(f) = \mathbb{R}_+$.

Conjunto Imagem da função: $Im(f) = \mathbb{R}_+$.
A lei de formação da função pode ser expressa por:

$$f : \mathbb{R}_+ \to \mathbb{R}_+ \quad \text{com} \quad f(x) = \begin{cases} 0; & \text{se } x \leq 22.847,76 \\ 0,075x - 1.713,58; & \text{se } 22.847,77 \leq x \leq 33.919,80 \\ 0,15x - 4.257,57; & \text{se } 33.919,81 \leq x \leq 45.012,60 \\ 0,225x - 7.633,51; & \text{se } 45.012,61 \leq x \leq 55.976,16 \\ 0,275x - 10.432,32; & \text{se } x > 55.976,16 \end{cases}$$

- Momento II:

```
print('\nFUNÇÕES DEFINIDAS POR MAIS DE UMA SENTENÇA')
print('-=-'*14)
print('PROBLEMA - Imposto de renda (cálculo anual).')

x = float(input('\nDigite o valor (em reais) da renda anual: R$ '))

if x < 0:
    print('Valor inválido!')

elif 0 <= x <= 22847.76:
    ir = 0
    print('Com uma renda anual de R$ {:.2f}, o valor do imposto de renda
     a ser pago será R$ {:.2f}.'.format(x, ir))

elif 22847.77 <= x <= 33919.80:
    ir = 0.075*x - 1713.58
    print('Com uma renda anual de R$ {:.2f}, o valor do imposto de renda
     a ser pago será R$ {:.2f}.'.format(x, ir))

elif 33919.81 <= x <= 45012.60:
    ir = 0.15*x - 4257.57
    print('Com uma renda anual de R$ {:.2f}, o valor do imposto de renda
     a ser pago será R$ {:.2f}.'.format(x, ir))

elif 45012.61 <= x <= 55976.16:
    ir = 0.225*x - 7633.51
```

```
24    print('Com uma renda anual de R$ {:.2f}, o valor do imposto de renda
      a ser pago será R$ {:.2f}.'.format(x, ir))
25
26 elif x > 55976.16:
27    ir = 0.275*x - 10432.32
28    print('Com uma renda anual de R$ {:.2f}, o valor do imposto de renda
      a ser pago será R$ {:.2f}.'.format(x, ir))
```

Código-fonte 5.10 – Problema - Imposto de Renda (cálculo anual)

5.1.9 Aula 9 - Considerações e Avaliações

- Tempo estimado: 2 aulas de 50 min.

  - Momento I: 1 aula
  - Momento II: 1 aula

- Desenvolvimento:

  - Momento I: Conduzir um debate, por meio de indagações, que permitam analisar as necessidades de declarar as variáveis para manipular números inteiros e racionais, bem como sobre a necessidade de definir os conjuntos domínio e imagem da função antes de realizar as codificações dos problemas. Recomenda-se algumas perguntas como: "Que tipos de variáveis foram declaras nos programas?", "Vocês conseguiram perceber que alguns problemas só permitiam a manipulação de números inteiros, enquanto outros operavam com racionais?", "Por que a necessidade de analisar os conjuntos domínio e imagem da funções?", "Existem outras maneiras de representar os intervalos por meio dos operadores de comparação?", "Você sentiu alguma diferença ao codificar programas diferentes? Quais?".

  - Momento II: Revisitar todos os programas elaborados e discutir a importância na distinção do uso das variáveis `int` e `float` de acordo com os conjuntos domínio e imagem das funções. Sugere-se uma analise detalhada do porquê da necessidade de decidir usar o `int` ou `float`. Além disso, analisar a importância das estruturas de decisão na definição dos diferentes intervalos do conjunto domínio, onde os condicionais `if` e `else` favoreciam essa separação. Inclusive, é valioso discutir, também, sobre as várias formas de se definir os intervalos utilizando os operadores de comparação como `<` e `<=`.

5.2 Ponderações

Considera-se importante, analisar os conjuntos Domínio e Imagem das funções, pois, ao declarar as variáveis numéricas na codificação, é necessário defini-las como um número inteiro ou flutuante. Dessa forma, os alunos serão criteriosos ao elaborar códigos de relações matemáticas que manipulam números decimais ou não, o que pode ficar evidente ao analisar a natureza dos conjuntos Domínio e Imagem das funções.

Ressalta-se que todos os problemas de programação propostos nessa Sequência Didática solicitam o valor numérico de uma função, isto é, nas execuções dos programas sempre é informado o valor de $f(x)$ para um determinado x fornecido pelo usuário. No entanto, o contrário também pode ocorrer. O professor pode moldar o enunciado do problema e solicitar aos alunos que codifiquem um modelo alternativo. Por exemplo, na Aula 5 (Subseção 5.1.5), o programa elaborado fornece o salário do vendedor ao solicitar a quantidade de produtos vendidos, porém, estrategicamente, o professor pode indagar os alunos sobre como proceder para que o programa realize o oposto. Isto significa que o problema agora consiste um codificar um programa que forneça a quantidade de produtos vendidos ao solicitar o valor do salário recebido por um vendedor, dessa maneira, alguns detalhes da programação ganharão uma nova roupagem.

Com isso, aponta-se que toda a Sequência Didática pode se tornar mais requintada e, consequentemente, mais enriquecedora para os alunos. Ao aprimorar os enunciados dos problemas, o professor pode contribuir ainda mais para o aperfeiçoamento dos alunos acerca dos conceitos estudados.

Além disso, os coeficientes das funções também podem ser modificados, de acordo com vários contextos e realidades, como exemplo, cita-se que os valores da tabela para o cálculo anual do Imposto de Renda podem mudar conforme o passar dos anos, fornecendo assim, novos coeficientes para a função. Desse modo, o professor tem autonomia para mudar os coeficientes das funções conforme a realidade dos alunos. Por exemplo, nos problemas sobre o consumo de energia elétrica e de água apresentados, respectivamente, na Avaliação Diagnóstica (Tabela 6) e no problema da Aula 7 (Subseção 5.1.7), o professor poderia investigar com os alunos as tarifas locais, de acordo com a distribuição de energia e água de seus estados. Tudo isso pode tornar as aulas ainda mais fomentadoras e estimulantes.

Aspira-se que a sequência didática apresentada possa ser de grande valia para os professores, como também para os alunos. Almeja-se contribuir efetivamente para o desenvolvimento do letramento matemático dos discentes. Espera-se também fornecer aos professores uma metodologia eficiente ao propor o uso desse recurso didático tecnológico.

6 Conclusões

Constata-se que a utilização da Linguagem de Programação como um recurso digital e tecnológico em sala de aula favorece a formação educacional dos estudantes e contribui com o trabalho pedagógico do professor. A sua exploração nas aulas de Matemática do Ensino Médio possibilita a abordagem de conceitos, como também, a discussão de estratégias para a solução de problemas.

Dessa forma, conclui-se que o uso da linguagem de programação associada à linguagem matemática corrobora com o aprimoramento do letramento matemático dos estudantes. Isto se dá por meio da contribuição do desenvolvimento das competências e habilidades de raciocinar, representar, comunicar e argumentar matematicamente, de modo a favorecer o estabelecimento de conjecturas, a formulação e a resolução de problemas, utilizando conceitos, procedimentos, fatos e ferramentas matemáticas.

De modo particular, o Python é uma linguagem de programação que facilita essa abordagem devido a sua sintaxe simples, sua clareza e objetividade. Sendo assim, infere-se que esta linguagem de programação específica possui um grande potencial pedagógico.

A composição desta dissertação promove uma assistência completa aos professores, sendo um tanto autossuficiente neste quesito, uma vez que oferece um tutorial com exemplificações, uma variedade de problemas e aplicações, bem como possibilidades reais de aplicação.

Portanto, com o desfecho deste trabalho, pretende-se auxiliar professores de Matemática do Ensino Médio na execução de estratégias pedagógicas que utilizam a Linguagem de Programação Python, além de motivá-los a desfrutarem com frequência deste poderoso recurso digital e tecnológico. Espera-se que o seu manuseio na vivência docente contribua para a melhoria do ensino de matemática e proporcione a compreensão de conceitos, facilitando a aprendizagem desta disciplina.

Referências

BERTOLINI, C. et al. Linguagem de programação i. Brasil, 2019. Citado na página 27.

BRASIL. *Matriz de Referência em Avaliação Matemática*. 2012. Disponível em: <https://download.inep.gov.br/acoes_internacionais/pisa/marcos_referenciais/2013/matriz_avaliacao_matematica.pdf>. Acesso em: 29 set 2022. Citado na página 28.

BRASIL. Base nacional comum curricular. *Brasília*, 2018. Citado 6 vezes nas páginas 19, 20, 21, 22, 23 e 24.

BRASIL. *Referenciais Curriculares para a elaboração de Itinerários Formativos*. Brasília, 2020. Citado 2 vezes nas páginas 25 e 26.

BRASIL, R. F. D. *Imosto de Renda*. 2017. Disponível em: <https://www.gov.br/receitafederal/pt-br/assuntos/orientacao-tributaria/tributos/irpf-imposto-de-renda-pessoa-fisica#c-lculo-anual-do-irpf>. Acesso em: 13 dez 2022. Citado na página 114.

CARNEIRO, S. d. S. et al. Uso da linguagem de programação python na formação de professor para o ensino de matemática. Universidade do Estado do Amazonas, 2022. Citado na página 27.

GIRALDO, V.; CAETANO, P.; MATTOS, F. Recursos computacionais no ensino de matemática. *Rio de Janeiro: SBM*, p. 127, 2012. Citado na página 27.

HAVERBEKE, M. *Eloquent JavaScript: a modern introduction to programming*. [S.l.]: No Starch Press, 2018. Citado na página 27.

INEP. *Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira - Provas e Gabaritos do ENEM*. 2022. Disponível em: <https://www.gov.br/inep/pt-br/areas-de-atuacao/avaliacao-e-exames-educacionais/enem/provas-e-gabaritos>. Acesso em: 13 dez 2022. Citado 2 vezes nas páginas 107 e 111.

LORENSATTI, E. J. C. Linguagem matemática e língua portuguesa: diálogo necessário na resolução de problemas matemáticos. *Conjectura: filosofia e educação*, Editora da Universidade de Caxias do Sul, v. 14, n. 2, p. 89–99, 2009. Citado na página 26.

MARCONDES, G. A. B. *Matemática com Python um guia prático*. [S.l.]: Novatec, 2018. Citado na página 33.

MENEZES, N. N. C. Introdução a programação com python. *São Paulo: Novatec*, 2010. Citado 4 vezes nas páginas 29, 32, 54 e 55.

PERNAMBUCO, G. d. E. Currículo de pernambuco ensino médio. *Secretaria de Educação e Eportes*, 2021. Citado na página 30.

RESNICK, M. Learn to code, code to learn. *EdSurge, May*, v. 54, 2013. Citado na página 28.

ZABALA, A. A prática educativa: como ensinar. trad. *Ernani F. da F. Rosa. Porto Alegre: ArtMed*, 1998. Citado na página 96.

Apêndices

APÊNDICE A – TRIÂNGULOS: Existência, Classificação e Área.

```
1  print('\nTRIÂNGULOS — Existência, Classificação e Área.')
2  print('—=—'*15)
3
4  print('\nVamos calcular a área, analisar a existência de um triângulo e a sua
       classificação?')
5  print('Informe as medidas dos seus lados.\n')
6
7  m1 = float(input('Medida do primeiro lado: '))
8  m2 = float(input('Medida do segundo lado: '))
9  m3 = float(input('Medida do terceiro lado: '))
10
11 #Definindo a sequência (m1 > m2 > m3)
12 maior = m1
13 meio = m2
14 menor = m3
15
16 if (m2 <= m3 <= m1):
17   maior = m1
18   meio = m3
19   menor = m2
20
21 if (m2 <= m1 <= m3):
22   maior = m3
23   meio = m1
24   menor = m2
25
26 if (m1 <= m2 <= m3):
27   maior = m3
28   meio = m2
29   menor = m1
30
31 if (m1 <= m3 <= m2):
```

```
32   maior = m2
33   meio = m3
34   menor = m1
35
36 if (m3 <= m1 <= m2):
37   maior = m2
38   meio = m1
39   menor = m3
40
41 if (m1 <= m2 <= m3):
42   maior = m3
43   meio = m2
44   menor = m1
45
46
47 if m1 < m2 + m3 and m2 < m1 + m3 and m3 < m1 + m2:
48    print('SIM! Os segmentos acima podem formar um triângulo ', end='')
49    if m1 == m2 == m3:
50       print('EQUILÁTERO e ACUTÂNGULO!')
51
52    elif m1 != m2 and m1 != m3 and m2 != m3:
53       print('ESCALENO ', end='')
54       if maior**2 < menor**2 + meio**2:
55          print('e ACUTÂNGULO!')
56       elif maior**2 > menor**2 + meio**2:
57          print('e OBTUSÂNGULO!')
58       elif maior**2 == menor**2 + meio**2:
59          print('e RETÂNGULO!')
60
61    elif (m1 == m2 and m1 != m3 and m2 != m3) or (m2 == m3 and m1 != m3 and m1
      != m2) or (m1 == m3 and m2 != m3 and m2 != m1):
62       print('ISÓSCELES ', end='')
63       if maior**2 < menor**2 + meio**2:
64          print('e ACUTÂNGULO!')
65       elif maior**2 == menor**2 + meio**2:
66          print('e RETÂNGULO!')
67       elif maior**2 > menor**2 + meio**2:
68          print('e OBTUSÂNGULO!')
```

```
69
70    p = (m1+m2+m3)/2
71    A = (p*(p-m1)*(p-m2)*(p-m3))**(1/2)
72    print('A área do triângulo é {:.2f}.'.format(A))
73
74 else:
75    print('NÃO! As medidas dos segmentos acima NÃO podem formar um triângulo.')
```

Código-fonte A.1 – TRIÂNGULOS (Codificação)