Avaliação de eficiência do curso introdução prática à programação de computadores

Jasson Marques Fontoura Júnior[1], Pedro Vinícius da Silva Ribeiro, Lucas Bessa Façanha Pereira, Kevin Willyn Conceição Barros, Ozéias Silva Souza, Rafael Nóbrega de Lima, Natália Ribeiro de Almada, Maely da Silva Moraes, Marcelo Henrique Oliveira Henklain

{jassonjr5, silvapv7, lucas.bessafp72, o.souzati, rafaelnobregadelima2809, almada.compsci}@gmail.com, {kw.willyn}@outlook.com

{maely.moraes, marcelo.henklain}@ufrr.br

Resumo. *Investigamos a eficiência de curso para o ensino do comportamento de "escrever programas de computador em Python". Participaram sete calouros de computação cuja tarefa foi (1) responder a prova de conhecimentos sobre programação no início e ao final do curso, e, ao final, uma escala de satisfação, (2) realizar exercícios ao longo do curso, (3) assistir a videoaulas e (4) participar de atividades presenciais para sanar dúvidas. Os resultados foram promissores em termos de aprendizado, autoconfiança e satisfação. Dos seis objetivos de aprendizagem, em cinco notamos no grupo aumento de percentual de acertos. Esperamos com este estudo contribuir com a superação dos desafios no ensino de programação.*
Palavras-chave: *Ensino Superior; Estudantes de computação; Ensino-aprendizado; Introdução à programação.*

Abstract. *We investigate the course efficiency for teaching the behavior of "writing computer programs in Python". Seven freshmen computer science majors participated, whose task was (1) to answer a test of programming knowledge at the beginning and end of the course, and, at the end, a satisfaction scale, (2) do exercises throughout the course, (3) watch video classes, and (4) participate in class activities to solve doubts. The results were promising in terms of learning, self-confidence e satisfaction. Of the six learning objectives, in five we noticed an increase in the group's percentage of correct answers. We hope with this study to contribute to overcoming the challenges in teaching programming.*
Keywords: *College education; Computer students; Teaching-learning; Introduction to programming.*

1. Introdução

O futuro da humanidade está marcado pela ampla disseminação no uso de tecnologias da computação. Com isso, é cada vez mais fundamental compreender como um computador

[1] Agradecemos ao apoio da Universidade Federal de Roraima, no âmbito do Edital 44/2022-PRAE/UFRR, na forma de bolsa de extensão, para a realização deste trabalho.

funciona e saber como programá-lo, assim como é vital conseguir ler, escrever e realizar as quatro operações [Machado, 2015; Unesco, 2014]. A programação é importante porque nos leva a poder intervir de modo intencional no mundo digital, criando soluções para problemas do cotidiano ou, pelo menos, compreendendo como funcionam as soluções computacionais que utilizamos. Além disso, pode melhorar nosso raciocínio lógico e criatividade, na medida em que aprendemos estratégias, a partir dos algoritmos, para lidar com diferentes tipos de problemas [Gomes, Teles, & Ramos, 2022].

Apesar de sua importância, estudantes de diversos níveis de ensino, inclusive de cursos específicos da área de computação, no ensino superior, podem experimentar dificuldades no aprendizado do comportamento de "programar computadores". Essas dificuldades podem levar a um baixo desempenho e à desmotivação dos estudantes, contribuindo para o aumento da taxa de evasão em cursos de Computação e Informática [Silva, 2016]. As principais dificuldades enfrentadas pelos alunos são relacionadas à compreensão da lógica de programação e ao emprego adequado da sintaxe das linguagens de programação [Holanda, Freire, & Coutinho, 2019]. Os professores de programação também enfrentam obstáculos para ensinar, os quais podem estar relacionados à baixa capacitação didática, falta de recursos pedagógicos e de pessoal para lidar com turmas grandes e com diversos tipos de demandas específicas de cada aluno [Silva, Brito, & Vaz, 2019]. Com efeito, Linhares, Pereira, Ribeiro, Moraes e Henklain (2021) reportaram, em um relato de experiência profissional, que o colegiado do curso de Ciência da Computação da Universidade Federal de Roraima, sentiu a necessidade de criar um projeto de extensão na forma de curso, que servisse como introdução à disciplina de algoritmos, justamente para tentar reduzir reprovações já no primeiro semestre da graduação. Nesse curso de extensão, os alunos começam a aprender o que significa programar e como isso pode ser feito por meio de uma linguagem de programação específica.

Ao considerarmos esse cenário, verificamos a relevância de que se desenvolvam capacitações de pessoas para que aprendam a "programar computadores" ou comportamentos mais básicos que constituem esse comportamento mais geral. Aliado a isso, notamos na literatura uma lacuna no sentido de falta de explicitação de quais são os comportamentos que estão sendo ensinados como parte de uma capacitação na área de programação, bem como nem sempre fica evidente de que forma (no sentido de critérios) o sucesso do ensino está sendo examinado em relação ao desenvolvimento de cada comportamento definido como alvo para ser aprendido. Esses cuidados são compatíveis com o manifesto de Bittencourt e Isotani (2018) em favor de uma informática na educação (ou, no caso, de uma educação em computação) baseada em evidências.

Assim, o objetivo deste estudo foi investigar a eficiência de um curso em relação ao ensino do comportamento de "escrever programas de computador simples por meio da linguagem de programação Python"[2], de modo que os seus comportamentos constituintes fossem explicitados e o processo avaliativo adotado, para inferir sobre a sua presença, aperfeiçoamento ou aquisição, fosse demonstrado. Consideramos que, desse modo, poderemos trazer contribuições para a área de Educação em Computação, pois

[2] A simplicidade dos programas reside no fato de terem poucas linhas de código, baixa demanda de conhecimento de informações externas àquelas explicitamente apresentadas nos enunciados dos exercícios de programação, que envolvem poucas estruturas de controle de fluxo, poucas entradas e poucas possibilidades de saída. Em estudos futuros, pretendemos definir critérios objetivos do que significa "simplicidade".

começaremos a examinar com maior precisão em que medida as condições de ensino planejadas funcionam na promoção de cada comportamento que se almeja desenvolver.

Este trabalho está organizado nas seguintes seções: Fundamentação teórica, Trabalhos relacionados, Método, Resultados e discussão e Conclusão. Na fundamentação teórica, são apresentados os principais conceitos e a teoria que adotamos para planejamento do curso e, ainda, para a interpretação dos dados deste estudo. Na seção de trabalhos relacionados, são apresentados artigos com problemas de pesquisa semelhantes ao que é desenvolvido neste estudo. Esses artigos são a base para compreender a lacuna existente na literatura e mostrar o diferencial deste trabalho. No final desta seção, explicitamos as perguntas que iremos responder por meio desta pesquisa. No método, apresentamos o curso como solução proposta para o desafio de ensinar programação, bem como a estratégia que adotamos para avaliação de eficiência desse programa de aprendizagem. Na seção de resultados e discussão, descrevemos os nossos achados e a argumentação para sustentar a conclusão do estudo, assim como as limitações da nossa pesquisa. Por fim, finalizamos com a seção de conclusão.

2. Fundamentação teórica

Adotamos neste estudo a teoria analítico-comportamental como fundamento científico para a nossa visão sobre Psicologia humana e, mais especificamente, sobre os processos de ensinar e aprender. Destacamos dois motivos que nos levaram à escolha dessa teoria: (1) seus conceitos são produtos de um programa sólido e sistemático de pesquisa experimental, de natureza básica e aplicada, com dados sobre seres humanos em diferentes países e condições, além de tais conceitos serem objeto de investigação filosófica criteriosa; e (2) essa teoria preza pelo princípio científico da parcimônia, buscando por explicações simples (jamais simplistas) e concretas para os fenômenos naturais [Skinner, 2005/1953], aspecto relevante para orientar o cientista no seu trabalho e evitar confusões conceituais. Adicionalmente, a Análise do Comportamento nos oferece um conjunto de ferramentas conceituais aplicáveis a toda diversidade do comportamento humano, seja ele interpretado pela sociedade como típico, atípico, saudável ou patológico, esteja ele ocorrendo em contexto de trabalho, lazer ou educacional [Todorov, 2007].

De acordo com pesquisadores analistas do comportamento, que trabalham aplicando a Análise do Comportamento ao contexto educacional, ensinar é o nome que sintetiza um conjunto complexo de comportamentos apresentados por professores e que têm por finalidade a produção de aprendizado [Kubo & Botomé, 2001]. Aprender, por sua vez, é o nome dado ao processo de mudança duradoura no repertório comportamental de uma pessoa, o que ocorre seja pela aquisição de um novo comportamento, que a pessoa não conseguia apresentar anteriormente, ou pelo aperfeiçoamento de um comportamento já adquirido previamente, mas cujo grau de competência foi ampliado [Kubo & Botomé, 2001].

Desse ponto de vista, o ensino é um trabalho que se realiza em função de um fim e isso marca a sua natureza enquanto um processo planejado. É preferível quando esse planejamento é feito de modo técnico e gradual, mas podem ocorrer situações em que o professor precise lidar com imprevistos, sendo necessário improvisar. De todo modo, independentemente do grau de planejamento possível, o fato é que ensinar pressupõe que

o professor, antes da sua aula, conheça ou formule de modo inequívoco o que espera que seja aprendido [Cortegoso & Coser, 2013]. Mesmo em um contexto de sala de aula, em que várias situações impeçam que o professor realize o que planejou e até o obriguem a mudar completamente de foco, é preciso que o educador tenha uma referência de quais são os comportamentos mais promissores a serem desenvolvidos. Isso nos conduz à noção de objetivos de aprendizagem.

Objetivos de aprendizagem são descrições dos comportamentos que o professor espera que os seus alunos adquiram e/ou aperfeiçoem até o final de um programa de aprendizagem (curso, disciplina etc.) para que, então, eles possam lidar de modo mais efetivo com a sua realidade social [Cortegoso & Coser, 2013]. Objetivos de aprendizagem refletem, portanto, comportamentos que, ao serem aprendidos, ampliam as capacidades do aprendiz para viver melhor em sua realidade e, justamente por isso, são relevantes.

É por conta dos objetivos de aprendizagem que os processos avaliativos adquirem maior propósito em termos de desenvolvimento dos alunos. Isso ocorre porque avaliamos não só para decidir pela aprovação/reprovação, mas, fundamentalmente, para examinar em que medida os objetivos propostos em um programa de aprendizagem estão sendo ou foram atingidos. Saber disso nos informa em que medida as condições de ensino[3] foram efetivas para que o aprendizado dos objetivos de aprendizagem ocorresse [Botomé & Rizzon, 1997].

Entendemos que um aspecto nuclear do que chamamos de "educação de qualidade" diz respeito justamente à demonstração inequívoca de que condições de ensino planejadas pelo professor promoveram ou, pelo menos, facilitaram o alcance dos objetivos de aprendizagem planejados. Por esse motivo, apresentaremos em nossa seção de 'Solução proposta', os objetivos de aprendizagem adotados neste estudo, que constituíram os alvos da nossa capacitação no campo da programação de computadores. Também descreveremos de que modo avaliamos as condições de ensino que organizamos, para examinar em que medida foram efetivas.

Nesse contexto do que são objetivos de aprendizagem, importa lembrar que programar computadores, sob a ótica analítico-comportamental, também é interpretado como um comportamento [Lazzari, 2013] e, por isso mesmo, pode ser proposto como objetivo de aprendizagem. De acordo com Lazzari (2013), programar computadores é um processo comportamental constituído por diversas outras classes de comportamento em diferentes graus de especificidade. No primeiro nível de especificidade, a autora propõe que "programar computadores" é constituído por (1) "avaliar argumentos de acordo com regras lógicas"; (2) "caracterizar funcionamento de computadores"; (3) "resolver problemas"; (4) "construir algoritmos"; (5) "formalizar algoritmos"; (6) "escrever programas de computador"; e (7) "avaliar programas de computador". Cada uma dessas sete classes de comportamento pode, por sua vez, ser decomposta em outras classes ainda mais específicas. Vale o esclarecimento de que um processo de decomposição, tipicamente, é finalizado quando se chega a comportamentos muito simples que, no contexto de ensino, correspondem ao que estudantes provavelmente já aprenderam ou que poderiam aprender com facilidade [Kubo & Botomé, 2001]. Esse processo de decomposição é útil porque permite ao professor se planejar para organizar condições de

[3] Exemplos de condições de ensino: aulas ministradas, slides exibidos, visitas realizadas a uma comunidade ou museu, jogos educativos utilizados, vídeos apresentados, textos indicados, palestras de convidados proferidas, problemas discutidos em sala, seminários conduzidos, listas de exercícios requeridas etc.

ensino que partam do que o aluno já sabe ou que facilmente aprenderá para, então, avançar em direção a comportamentos mais complexos.

O aprendizado desse comportamento de "programar computadores" constitui parte fundamental do repertório de um profissional da computação, sendo necessário que seja aprendido por quem está buscando atuar profissionalmente nessa área. Para além disso, conforme defendem Linhares et al. (2021), o aprendizado desse comportamento pode ser útil para qualquer pessoa, afinal vivemos em um mundo amplamente afetado e marcado pelas tecnologias digitais. Adicionalmente, achados na literatura científica têm sugerido que, por meio da programação, entre outras aprendizagens, começamos a desenvolver um repertório de resolução de problemas, denominado de pensamento computacional, que pode ser aplicável às mais diversas situações cotidianas, não só aquelas que envolvem problemas técnicos da computação.

Um exemplo de pensamento computacional é a estratégia, amplamente usada na programação, de "Dividir para conquistar". Essa estratégia orienta muitos algoritmos, como os de ordenação (exemplo: *mergesort* e *quicksort*), e consiste na resolução de um problema a partir de três etapas: (1) divisão do problema principal em vários pequenos problemas, mais fáceis de serem solucionados; (2) resolução dos pequenos problemas; e (3) junção de todas as soluções para que, então, o problema inicial seja resolvido. Ao aprender isso, podemos generalizar para a vida, pois muitos problemas do nosso cotidiano podem ser solucionados quando divididos em problemas menores, mais simples ou com solução já conhecida. Assim, notamos que o pensamento computacional é uma habilidade criativa, crítica e estratégica, que utiliza os fundamentos da computação para resolver problemas de maneira eficaz de modo que tanto pessoas quanto computadores possam executar a solução proposta [Scaico et al., 2013; Brackmann, 2017].

Em face desse cenário que descrevemos, destacamos que, apesar da relevância da programação, lidamos com um paradoxo: embora muito importante, estudantes de diversos níveis de ensino, inclusive de cursos específicos da área de computação, no ensino superior, experimentam dificuldades no aprendizado do comportamento de "programar computadores". Por vezes, isso está associado ao fato de que o aluno não é ensinado a analisar os requisitos do problema, antes de codificar uma solução [Franzen, Bercht, & Dertzbacher, 2017]. Isso revela que precisamos conhecer melhor sobre os comportamentos específicos que constituem o processo de programar computadores, pois nem sempre ensinamos todos os comportamentos pré-requisito para que a pessoa torne-se apta a programar computadores.

Assim, os educadores enfrentam o desafio de entender como podem ajudar os alunos a programar, de modo que as dificuldades encontradas nesse processo não sejam impeditivas para o avanço e conclusão do curso [Júnior & Bogea, 2020]. Para lidar com esse desafio, partimos do fato de que as estratégias de ensino utilizadas pelos professores são um fator determinante para ajudar os alunos a superar as dificuldades no processo de aprendizagem [Zanin, Sparremberger, & Barbosa, 2019]. Tais fatos, portanto, que são a necessidade de descrição dos comportamentos que constituem o processo de programar computadores e de construção de estratégias mais eficazes para ensinar esse comportamento, ampliam a demanda por estudos nessa área da Educação em Computação. Examinaremos melhor essa literatura sobre ensino de programação e suas dificuldades na seção de trabalhos relacionados.

Neste ponto da seção, entendemos ser importante esclarecer que, por meio dos conceitos apresentados e da estrutura rigorosamente planejada de nosso processo de ensino, não queremos dar a impressão de que, baseados na teoria da Análise do Comportamento, consideramos os processos de ensinar e aprender a partir de uma ótica simplista. Ou, ainda, que por meio de um único parâmetro avaliativo, baseado em acertos e erros, podemos falar em ensino de qualidade. Pelo contrário, estudos em Análise do Comportamento revelam múltiplas variáveis que podem interferir nos comportamentos de ensinar de professores e nos comportamentos de estudar de alunos, explicitando, desse modo, a complexidade inerente à produção de aprendizagens [Todorov, Moreira, & Martone, 2009; Skinner, 2003/1968]. Por exemplo, variáveis motivacionais podem dificultar o aprendizado do estudante. Além disso, a falta de formação adequada para o docente e de recursos para o trabalho (inclusive, de tempo de qualidade para preparação de condições de ensino e de avaliação formativa), podem reduzir as suas chances de sucesso no ensino. Condições sociais, econômicas e de saúde pública de um país também podem afetar processos de ensino-aprendizagem.

Nessa perspectiva analítico-comportamental de fugir à supersimplificação, adotaremos como parâmetros de qualidade do nosso curso, três resultados esperados: (1) aprendizado dos comportamentos propostos como objetivos, o que será inferido a partir das percepções dos próprios alunos e de seus acertos em um instrumento desenhado para esta pesquisa, que será apresentado em detalhes no método; (2) satisfação com o curso, afinal o aprendizado não deveria ser alcançado por meio de estratégias aversivas, uma vez que isso pode enfraquecer o comportamento de estudar das pessoas, algo que não seria desejável como subproduto do processo de ensino; e (3) aumento da percepção de autoconfiança em relação ao próprio desempenho. Vamos explicar a lógica dessas três expectativas teóricas de resultado.

Tecnicamente, a teoria analítico-comportamental mostra que, parte das importantes aprendizagens que adquirimos ao longo da vida, têm como mecanismo explicativo básico uma relação de condicionamento ou dependência entre ações de uma pessoa (apresentadas em determinado contexto) e consequências chamadas de reforçadoras [Skinner, 2003/1968]. Uma consequência é denominada de reforçadora quando a sua ocorrência, após uma ação, entre outros efeitos, torna essa ação mais provável de recorrer no futuro. Portanto, não são reforçadores aquilo que achamos ser reforçador ou o que gostaríamos que fosse e, tampouco, o que uma pessoa afirma ser reforçador para si própria, pois ela pode estar equivocada. Para falarmos em reforço, precisamos verificar que os efeitos do reforçamento ocorreram.

Aplicando isso ao ensino, o que esperamos que os professores façam é, partindo dos objetivos de aprendizagem e das condições de ensino planejadas, organizar em sala de aula situações nas quais os estudantes apresentem ações, diante de determinado contexto, para que, então, se os parâmetros dessas ações-em-determinado-contexto estiverem adequados, sejam seguidas por consequências potencialmente reforçadoras. Um exemplo pode ajudar a ilustrar o que foi explicado. Um professor de programação pode querer ensinar o comportamento de “exibir mensagens na tela do computador por meio da linguagem de programação Python”. Para isso, ele pode explicar a finalidade da função “print()” em Python e pedir que (1) no contexto do Jupyter Notebook, já aberto na máquina que cada estudante tem diante de si, (2) o aluno digite `print("Hello world!")` e, em seguida, (3) o professor pode orientar os alunos a executar esse código

para que tenham acesso à consequência, que é ver o texto “Hello world!” exibido na tela do computador.

Se as consequências planejadas pelo professor forem de fato reforçadoras para o estudante, espera-se, com isso, “ativar” o mecanismo de aprendizado pelas consequências, tornando mais provável que o estudante volte a apresentar a ação, no mesmo contexto e, ainda, em situações similares, em função da consequência que produziu. Geralmente, as primeiras consequências planejadas por um professor podem ser artificiais, tais como pontos ou reconhecimento pelo bom trabalho realizado. Espera-se, porém, que conforme o estudante se comporte mais e mais, as consequências naturais de, por exemplo, resolver um problema, passem a ser determinantes para o aprendizado e manutenção do comportamento.

Em síntese, por meio desses arranjos organizados pelo professor, para que o estudante se comporte e receba consequências, pode ocorrer aprendizado. Essa mudança de repertório passa, então, a ser possível de identificar sob a forma de aumento de acertos em tarefas que demandem que o aluno apresente os comportamentos que aprendeu. Conforme o estudante apresenta um comportamento, obtendo reforço, amplia-se, segundo a teoria, o sentimento de autoconfiança em relação a esse comportamento específico que foi reforçado. Isso pode ser identificado pela indicação do estudante de que está seguro da resposta que fornece em um teste. Finalmente, se o professor organizou condições de ensino amenas e agradáveis, evitando o uso de coerção, é provável que o aluno se sinta satisfeito com a experiência educacional, o que pode ser detectado por instrumentos de satisfação.

Assim, a teoria da Análise do Comportamento nos orienta em relação a como planejar o ensino e, ainda, evidencia o que podemos mensurar para avaliação de sua efetividade. Em nosso estudo, trabalharemos com três parâmetros, mas poderíamos usar muitos outros. Contudo, consideramos que temos muito o que aprender e aperfeiçoar em relação à medida de aprendizado, de autoconfiança e de satisfação. Por isso, temos trabalhado, por enquanto, apenas com esses três indicadores.

3. Trabalhos relacionados

Nesta seção apresentamos investigações sobre o ensino de programação, para identificação das principais lacunas existentes na literatura. Essas pesquisas serão apresentadas considerando a semelhança entre os problemas que investigam e aqueles que são examinados no presente estudo.

Medeiros, Falcão e Ramalho (2020) conduziram estudo cujo objetivo foi caracterizar os desafios de ensino e aprendizagem nas disciplinas de introdução à programação em nível universitário, por meio de uma revisão de literatura. Eles conduziram uma busca manual de artigos brasileiros, entre os anos de 2010 a 2016, sobre esse tema. Os pesquisadores identificaram na literatura a existência de barreiras para o aprendizado, como a falta de habilidades de interpretação de texto ou déficits em matemática, que dificultam o processo de abstração. Os problemas de escalabilidade, como salas com muitos alunos, poucos professores e monitores e, ainda, laboratórios depreciados e ultrapassados, também foram aspectos identificados como dificuldades

para o ensino de programação. Os pesquisadores concluíram que o ensino superior brasileiro necessita de mais atenção pela comunidade de Educação em Computação.

Fiss et al. (2018), por sua vez, investigaram o ensino de programação em conjunto com o pensamento computacional. A tarefa dos participantes, distribuídos em dois grupos (A e B, ambos com alunos do 3º e 5º semestres do curso de engenharia de computação, sendo que no B havia um aluno do 7º), consistia em cursar uma oficina, com quatro horas de atividades presenciais e oito de atividades à distância, no Moodle. As atividades presenciais envolviam exercícios de programação apresentados na forma de uma técnica de dinâmica de grupo. Nas atividades à distância, que eram individuais, os alunos tinham que construir um algoritmo para quatro problemas distintos. Os pesquisadores verificaram que o Grupo A teve um pouco mais de dificuldade em realizar as atividades. Eles perceberam que o conhecimento prévio dos alunos ajudou na realização das atividades e notaram também que os alunos tiveram dificuldade em acompanhar o raciocínio durante a execução das soluções abstratas, algo semelhante ao que foi identificado por Medeiros, Falcão e Ramalho (2020). Por meio das atividades à distância, os autores puderam perceber que existiu uma falta de verificação de erros na construção dos algoritmos e, com isso, eles concluíram que um acompanhamento continuado dos alunos poderia produzir um melhor efeito no ensino de programação. Em razão de dificuldades como as encontradas nesse estudo, ocorrem muitas reprovações em disciplinas de algoritmos e lógica de programação, o que favorece a evasão.

Considerando esse cenário de dificuldades no ensino de programação, muitos estudos buscaram desenvolver estratégias para superar esse desafio. Silva et al. (2019), por exemplo, conduziram estudo cujo objetivo foi identificar, a partir de percepções dos professores, formas de amenizar os índices de reprovação nas disciplinas de programação. Para coletar esses dados, os pesquisadores utilizaram um questionário do Google aplicado para sete professores. Os participantes apontaram que, geralmente, parte das dificuldades que os alunos enfrentam estão associadas às metodologias utilizadas pelos professores, tais como aulas cansativas e tradicionais, a mesma forma de ensino do início ao fim, sem variar ou buscar novas estratégias de ensino. Além disso, listaram várias ferramentas de ensino que poderiam auxiliar os professores a diminuir os índices de reprovação, a saber: Juízes Online, Ambientes de desenvolvimento visual e Compiladores de Portugol.

Ainda sobre reprovações e evasão, Ribas, Bianco e Lahm (2016) destacaram que existe grande evasão de alunos em cursos de tecnologia, que está correlacionada às taxas de reprovações em disciplinas iniciais de algoritmos e lógica de programação. Tal como concluído no estudo de Silva et al. (2019), Ribas et al. avaliam que um dos maiores problemas existentes no ensino de programação são as metodologias utilizadas. Por isso, esses pesquisadores propõem um ensino que estimule o estudo de lógica de programação por meio de linguagens visuais. Essa hipótese foi testada por meio da aplicação de uma série de problemas relacionados à lógica de programação, divididos em três níveis, com aumento gradual de dificuldade, para 37 alunos do 1° semestre do curso de ciência da computação. Por meio de comparação quantitativa das notas entre semestres que utilizaram metodologia tradicional e aqueles expostos à metodologia visual, verificou-se que, utilizando o meio tradicional, obteve-se cerca de 23% de aprovados, 43% de desistentes e 34% de reprovados. Em contrapartida, por meio das linguagens visuais, obtiveram-se 46% de aprovados, 46% de reprovados e 8% de desistentes. Nota-se que houve uma melhora sutil, tanto em relação ao percentual de desistentes quanto ao percentual de aprovados, com o uso de metodologias visuais.

Daly (2011) também mostra em seu estudo como a metodologia de ensino pode impactar no aprendizado. Dois grupos foram expostos a uma disciplina de introdução a lógica de programação, da qual participaram pessoas de diferentes áreas de graduação. Um grupo de 11 participantes utilizou a plataforma Alice por seis semanas para aprender de maneira mais amigável, sem preocupação com pontos e vírgulas. Só então foram introduzidos à linguagem Java. Já o outro grupo, com 18 participantes, utilizou Java desde o início. Todos completaram as mesmas tarefas ao longo do mesmo intervalo de tempo. O pesquisador observou que o grupo que utilizou a plataforma Alice se mostrou mais confiante, com maior ganho de desempenho e mais propenso a seguir estudando programação, mesmo esse grupo tendo menos pessoas graduadas na área de Tecnologia.

Assim como nas pesquisas anteriores, Vidotto, Pozzebon, Tarouco e Silva (2021) conduziram estudo com o objetivo de aumentar a aprovação de alunos em matérias de programação e, assim, diminuir a taxa de evasão. Participaram 55 alunos, sendo 20 de Engenharia Mecatrônica e 35 de Engenharia Civil, da disciplina de Programação I. A tarefa desses alunos consistia em formar duplas e, assim, construir jogos digitais com realidade aumentada a partir do que haviam aprendido em sala. No final das atividades, cada dupla teria que mostrar o jogo criado. Antes de começar a intervenção, foi aplicado um Pré-teste para identificar se os alunos já haviam utilizado programação em blocos, se tinham experiência em programação de jogos digitais e se conheciam a ferramenta que eles iriam utilizar, no caso, o Scratch. Os pesquisadores notaram, baseado no pré-teste, que a maioria dos alunos não tinha muita experiência com programação. Após o fim de todas as atividades, foi aplicado um Pós-teste para identificar os ganhos de repertório em programação e para avaliar a satisfação dos alunos. Nessa avaliação, a maioria dos participantes afirmou que tinha sido divertido, sendo que 68% dos alunos de Engenharia Mecatrônica e 64% dos alunos de Engenharia Civil afirmaram que aprenderam os conceitos básicos de programação. Após o projeto, os alunos de ambos os cursos fizeram uma transição para uma linguagem de programação tradicional, e com isso a maioria progrediu com facilidade e alcançou a aprovação na disciplina de Programação I, diferentemente do que ocorria antes dessa intervenção.

A partir dos trabalhos abordados, podemos concluir que existem aspectos de repertório, cuja descrição ainda é pouco clara (exemplo: raciocínio, abstração), que precisam ser aprendidos para que os alunos progridam em disciplinas de programação. Notamos, ainda, que a metodologia de ensino adotada pelo professor é importante em termos de aprendizado e motivação. Os estudos revisados buscaram alternativas de ensino em linguagens visuais. Nos questionamos, porém, se é possível obter bons resultados em termos de aprendizado autoconfiança e satisfação, mesmo começando o ensino com linguagens tradicionais. Neste estudo, consideramos esses dois aspectos. O nosso objetivo foi investigar a eficiência de um curso em relação ao ensino do comportamento de "escrever programas de computador simples por meio da linguagem de programação Python". Buscamos responder, especificamente, às seguintes perguntas:

P01. O curso foi eficiente na promoção de aprendizagens para os objetivos de aprendizagem formulados?

P02. O curso foi eficiente na promoção de aprendizagens para todos os alunos?

P03. Os alunos aumentaram o seu grau de autoconfiança em relação à programação?

P04. O curso foi uma experiência agradável para os aprendizes?

4. Método

4.1. Solução proposta

Objetivos de aprendizagem. A Tabela 1 exibe o conjunto de objetivos de aprendizagem propostos neste estudo para o ensino do comportamento de "escrever programas de computador simples por meio da linguagem de programação Python". Na tabela, especificamos com os números à esquerda de cada objetivo, a relação hierárquica (e de pertencimento) existente entre eles. Nota-se, então, que existem objetivos de aprendizagem que refletem comportamentos mais gerais e objetivos que correspondem a comportamentos mais específicos, constituintes dos comportamentos mais gerais.

Tabela 1. Objetivos de aprendizagem propostos para capacitar estudantes a "escrever programas de computador simples por meio da linguagem de programação Python".

Objetivos de aprendizagem
1.1. Diferenciar algoritmo de programa de computador. **1.1.1.** Definir o conceito de algoritmo. **1.1.1.1.** Caracterizar a finalidade de um algoritmo. **1.1.2.** Definir o conceito de programa de computador. **1.1.2.1.** Definir o conceito de programação. **1.1.3.** Descrever o funcionamento do computador a partir do conceito de programa. **1.1.3.1.** Definir o conceito de computador.
1.2. Definir o conceito de variável. **1.2.1.** Caracterizar tipos de dados tipicamente usados em programas de computador. **1.2.1.1.** Caracterizar dados inteiros. **1.2.1.2.** Caracterizar dados decimais. **1.2.1.3.** Caracterizar dados booleanos. **1.2.1.4.** Caracterizar dados *string*.
1.3. Caracterizar contextos e regras de uso de operadores típicos da programação de computadores. **1.3.1.** Caracterizar contextos e regras de uso de operadores aritméticos. **1.3.2.** Caracterizar contextos e regras de uso de operadores relacionais. **1.3.3.** Caracterizar contextos e regras de uso de operadores lógicos.
1.4. Caracterizar estruturas de controle de fluxo. **1.4.1.** Caracterizar estrutura do tipo seleção. **1.4.1.1.** Caracterizar estrutura *if-then-else*. **1.4.2.** Caracterizar estrutura do tipo laço de repetição. **1.4.2.1.** Caracterizar estrutura do tipo *while*. **1.4.2.2.** Caracterizar estrutura do tipo *for*.
1.5. Formalizar algoritmos na linguagem de programação Python. **1.5.1.** Construir algoritmos. **1.5.1.1.** Traduzir algoritmo em termos da linguagem Python.

1.5.1.1.1. Identificar recursos básicos para programação em Python.

1.6. Identificar finalidade, resultados e erros em programas com base no código.

Vale destacar que os objetivos de aprendizagem propostos neste estudo são compatíveis com alguns dos 10 principais tópicos mais abordados no ensino de programação no Brasil, identificados por Silva et al. (2022). No caso, os tópicos que contemplamos em nossos objetivos são os seguintes: "1º - variáveis, constantes e atribuições", "2º - comandos condicionais", "3º - comandos de repetição" e "5º - expressões aritméticas, lógicas e relacionais". Além disso, os objetivos relativos à formalização de algoritmos e de leitura de código constituem comportamentos com alta complexidade e que os estudantes, tipicamente, erram mais [Araújo et al., 2021]. Finalmente, cumpre destacar que os objetivos de aprendizagem propostos são compatíveis com a análise conduzida por Lazzari (2013).

Condições de ensino. Para alcançar os objetivos de aprendizagem, foi elaborado planejamento de ensino que consistiu em sete aulas, organizadas da seguinte forma: a cada semana era disponibilizada uma videoaula com duração de até 20 minutos, em conjunto com uma atividade relacionada aos conceitos apresentados na videoaula, ambas disponibilizadas pelo sistema Moodle. No total, foram produzidas sete videoaulas e seis atividades *on-line* de verificação de conhecimento (a última atividade era referente às Aulas 06 e 07). Adicionalmente, foram realizados nove encontros presenciais com os alunos para responder a dúvidas, gerar engajamento em relação ao curso e proporcionar treino adicional sobre os conceitos estudados, utilizando nesse contexto a plataforma *Kahoot*.

Antes do início das aulas, houve um encontro presencial com os alunos, em que foi apresentado o plano de ensino e aplicado o pré-teste, para coletar os dados dos conhecimentos que eles tinham antes da realização do curso. Na primeira aula foi abordado o funcionamento dos computadores e a diferença entre algoritmo e programas. Nesse momento, os alunos tiveram como atividade *on-line* questões de verdadeiro ou falso, em que tinham que identificar os conceitos corretos de algoritmo, programação e computador. Para a segunda aula, foram abordados os tipos de dados, o conceito de variável e foi apresentada uma atividade *on-line* correspondente, com perguntas de verdadeiro ou falso. A terceira aula abordou o uso dos operadores aritméticos na programação, seguido de atividade *on-line* que pedia dos alunos a apresentação dos resultados de operações, seguindo as regras e ordem de prioridade aritméticas. Na quarta aula, foram apresentados conceitos sobre lógica de programação, mostrando a estrutura de controle de fluxo *if-then-else*. Na atividade *on-line*, os alunos responderam perguntas teóricas de verdadeiro ou falso. Na quinta aula, ainda sobre lógica de programação, foi apresentado o conceito de laço de repetição, a partir das estruturas *while* e *for*. Na atividade *on-line*, os alunos tiveram que criar um pseudocódigo, usando o *while* ou *for*, que lesse dois números e escrevesse na tela o resultado da divisão de ambos. Caso o segundo número fosse zero, o programa não deveria permitir a divisão.

Quanto à sexta e sétima aulas, ambas foram realizadas na mesma semana e tiveram a mesma atividade *on-line*, tendo abordado a linguagem Python. Os alunos viram e praticaram o funcionamento do Python, aplicando todos os conceitos vistos anteriormente. Como atividade *on-line*, tiveram que criar um algoritmo que lesse números

de um usuário e imprimisse o maior e menor valor de todos eles, e que depois parasse de receber valores quando o valor de entrada fosse zero. Após esses encontros, foi realizado o último, no qual houve conversa sobre os assuntos tratados durante todo curso e como os alunos se sentiam em relação a eles. Logo depois, ocorreu a premiação dos alunos que mais acertaram questões nas atividades *on-line* e nas competições de *Kahoot* realizadas nos encontros presenciais. Para encerrar a participação dos alunos, foi realizada a aplicação do Pós-teste para averiguar a segurança deles ao responder sobre os conhecimentos abordados no curso e o grau de acertos. Aplicamos, ainda, instrumento para avaliação do nível de satisfação com o curso.

4.2. Avaliação Experimental

4.2.1. Participantes

Participaram sete estudantes calouros de um mesmo curso de computação de universidade pública, P1, P2, P3, P4, P5, P6 e P7, que ainda não haviam sido expostos a nenhuma disciplina da graduação. Desse grupo, seis participantes eram homens e uma era mulher. A idade média foi 18,65 anos (*DP* = 0,43), variando de 18 a 19. Um participante classificou-se como branco, um como preto e os demais como pardos. Não participaram dessa amostra alunos com deficiência. Nenhum participante estava trabalhando ou já havia tido essa experiência, fosse dentro ou fora da área de computação. Isso é coerente com o fato de que todos reportaram serem dependentes financeiramente de seus pais/responsáveis. Em termos de renda familiar, quatro reportaram rendimento entre dois a cinco salários-mínimos, e os demais indicaram rendimentos entre cinco e oito salários-mínimos.

No que diz respeito aos conhecimentos de programação, verificamos que cinco participantes disseram que já haviam realizado algum curso na área (P1, P4, P5, P6 e P7), mas apenas três deles afirmaram conhecer uma ou mais linguagens de programação (P1, P6 e P7). As linguagens citadas foram: Python (2 vezes), JavaScript (2 vezes) e com uma ocorrência apareceram outras linguagens, citadas como conhecidas por P7: C, C#, C++, Cobol, Go, Java, Kotlin, PHP e Ruby.

Destacamos que todos os participantes eram maiores de idade e tiveram que indicar em um Termo de Consentimento Livre e Esclarecido (TCLE) digital o seu aceite em participar da pesquisa, para que pudessem ter acesso ao Pré-teste e, na sequência, ao curso. Este projeto foi aprovado por Comitê de Ética em Pesquisa, Parecer n. 2.133.013 e CAAE n. 66249317.4.0000.5302.

4.2.2. Instrumentos

Avaliações realizadas no início e ao final do processo de ensino. Perguntamos aos participantes, antes do ensino e após, a percepção que tinham em relação ao próprio conhecimento acerca de: (1) programação de computadores; (2) conceito de algoritmo; (3) conceito de linguagem de programação; (4) conceito de computador; (5) conceito de programa de computador; (6) conceito de variável; (7) conceito de tipos de dados; (8) conceito de estrutura de controle de fluxo; (9) conceito de laço de repetição; (10)

programação por meio da linguagem Python. A primeira pergunta era respondida por meio de uma escala de quatro pontos: 1 - Nulo, 2 - Básico, 3 - Intermediário e 4 - Avançado. As demais questões eram respondidas com Sim ou Não. Além disso, criamos um instrumento para avaliação do repertório de entrada e saída dos participantes do estudo. Em outros termos, o que já conheciam e o que passaram a conhecer (o instrumento construído está disponível no Apêndice 1).

Todos os itens desse instrumento de avaliação do repertório dos participantes foram adaptados para um formato em que pudessem ser respondidos como Verdadeiro ou Falso. Utilizamos uma estratégia para tornar a resposta do participante mais informativa. Criamos uma escala de quatro pontos para ser usada em relação a cada um dos 30 itens, a saber: 1 - Totalmente seguro de que é Falso; 2 - Parcialmente seguro de que é Falso; 3 - Parcialmente seguro de que é verdadeiro; 4 - Totalmente seguro de que é verdadeiro. Com essa estratégia, para cada resposta, conseguimos examinar acerto/erro e grau de segurança na resposta fornecida. Esperamos, portanto, aumento no número de acertos, refletindo ganho de repertório entre o momento anterior e posterior ao ensino, bem como desenvolvimento de um sentimento de segurança em relação ao que está sendo respondido.

Avaliações realizadas ao longo do processo de ensino. O curso teve um total de sete aulas e, após cada uma, era disponibilizado um questionário na plataforma Moodle, totalizando seis questionários (sendo o último referente às Aulas 06 e 07), relativos às aulas ministradas, para avaliar a efetividade do ensino que estava sendo realizado e, ainda, para favorecer o comportamento de estudo do aluno. Além dos questionários, era feita a interação com a turma utilizando o *Kahoot*, que é uma plataforma que utiliza a gamificação para incentivar a participação dos alunos em relação às condições de ensino. Nessa plataforma, criamos três questionários, após as Aulas 1, 2 e 3. Trabalhamos nessas atividades o reconhecimento dos nomes corretos de vários tipos de dados, e a identificação de usos corretos dos diversos tipos de operadores existentes. Ao final do curso, foi conduzida uma atividade no *Kahoot* que contemplava todos os comportamentos abordados no curso.

Avaliação de satisfação. Criamos 12 itens em relação aos quais avaliamos a satisfação dos alunos, a saber: 1 - Método de aviso (confirmação) sobre a participação do curso; 2 - Suficiência das informações divulgadas sobre o curso para orientar a tomada de decisão sobre fazer ou não a inscrição; 3 - Infraestrutura do laboratório no qual o curso foi realizado (ar-condicionado, *datashow*, cadeiras, mesas, iluminação, etc.); 4 - Infraestrutura do local no qual o curso foi realizado (acesso, estacionamento, banheiros, etc.); 5 - Horários de início/término do curso; 6 - Relevância do conteúdo apresentado; 7 - Velocidade da apresentação do conteúdo; 8 - Assistência/disponibilidade do(a) monitor(a) diante das dúvidas ou outras necessidades ao longo do curso; 9 - Clareza da comunicação do(a) monitor(a); 10 - Domínio que o(a) monitor(a) demonstrou sobre o tema; 11 - Recursos utilizados para o ensino; 12 - Minha satisfação com o curso de um modo geral. Todos esses itens foram respondidos em uma escala tipo Likert de quatro pontos: 0 - Totalmente insatisfeito; 1 - Parcialmente insatisfeito; 2 - Parcialmente satisfeito; 3 - Totalmente satisfeito.

4.2.3. Procedimento de coleta e análise de dados

Para a divulgação, entramos em contato com os calouros do curso de ciência da computação por meio do "WhatsApp". A coleta de dados ocorreu em duas etapas de avaliações utilizando a plataforma "Google formulários". As duas etapas consistiram na aplicação do mesmo formulário, um ao início do curso e outro ao final. O instrumento avaliativo continha 30 questões de verdadeiro ou falso, em relação as quais mensuramos acertos e o grau de confiança dos alunos a cada resposta fornecida. Ao final do curso, avaliamos também a satisfação dos alunos.

Durante a aula de abertura, foi iniciada a primeira etapa da coleta de dados para obter informações sobre o nível de conhecimento dos alunos em lógica de programação. Algumas outras perguntas foram relacionadas à caracterização do indivíduo, tais como sexo, data de nascimento e renda familiar. Foram tomadas medidas de garantia de proteção e sigilo dos dados. Assim, todos os participantes tiveram que assinar o Termo de Consentimento Livre e Esclarecido (TCLE), para que pudessem ser incluídos no estudo. Após a realização das aulas e atividades práticas do projeto, iniciou-se a segunda etapa da coleta de dados, para que pudéssemos verificar a diferença entre o desempenho dos alunos antes e depois da realização do curso.

Os dados foram, predominantemente, analisados a partir de estatísticas descritivas, tais como porcentagem, média e desvio-padrão. Utilizamos, ainda, o teste de postos com sinais de Wilcoxon para avaliar se houve diferença entre o desempenho e a segurança entre o início e ao final do curso, e o *g* de Hedges para examinar o tamanho do efeito.

5. Resultados e Discussão

Começamos pela explicitação da comparação dos dados antes e após o ensino. A Tabela 2 exibe os resultados dos participantes antes e após o ensino, em relação à percepção que manifestaram sobre os próprios conhecimentos acerca de programação, cujas respostas poderiam variar de 0 (nulo) a 4 (avançado), e o conhecimento sobre a linguagem Python, cuja resposta podia ser apenas sim (1) ou não (0).

Tabela 2. Resultados relativos à percepção dos participantes acerca do próprio conhecimento em relação à programação e ao Python, antes e após o ensino.

Participante	Conhecimento					
	Programação			Linguagem Python		
	Antes	Após	Diferença	Antes	Após	Diferença
1	1	1	0	1	1	0
2	0	2	+2	0	0	0
3	1	1	0	0	1	+1

4	1	2	+1	0	1	+1
5	1	1	0	0	1	+1
6	1	1	0	1	1	0
7	2	1	-1	0	0	0

Observamos na Tabela 2 que a percepção sobre o conhecimento de programação mudou de forma positiva para dois participantes, manteve-se a mesma para quatro e piorou para um. Uma hipótese para justificar esse último dado negativo é o de que a capacitação ofertada consistia apenas em uma introdução à programação, tratando de desenvolver comportamentos mais básicos e em relação a uma linguagem específica, Python. Além disso, pode ser que o contato com as atividades de programação ao longo do curso tenha ampliado os critérios ou o rigor aplicado pelos participantes no momento de se autoavaliar. Quando examinamos, especificamente, a habilidade de programar em Python, verificamos que três participantes reportaram melhora e outros três mantiveram a percepção do início do curso.

A Tabela 3 exibe os resultados das oito perguntas sobre a percepção dos participantes em relação ao domínio de conceitos importantes no contexto do aprendizado de programação. Incluímos colunas de "Pt" para indicar quantos pontos, mínimo de zero e máximo de oito, cada participante apresentou antes e após o ensino. Ressaltamos que cada uma das oito perguntas se refere ao aprendizado de objetivo de aprendizagem apresentado na Tabela 1.

Tabela 3. Resultados da percepção dos participantes em relação ao próprio conhecimento antes e após o ensino.

P.	Perguntas Antes									Perguntas Após									Dif.
	1	2	3	4	5	6	7	8	Pt	1	2	3	4	5	6	7	8	Pt	
1	1	1	1	1	1	0	1	1	7	1	1	1	1	1	1	1	1	8	+1
2	0	0	1	1	0	0	0	0	2	1	1	1	1	1	1	1	1	8	+6
3	1	1	1	0	0	0	0	0	3	1	1	1	1	1	1	1	1	8	+5
4	1	1	1	1	1	0	0	1	6	1	1	1	1	1	1	1	1	8	+2
5	1	1	1	1	1	0	0	0	5	1	1	1	1	1	1	1	1	8	+3
6	0	0	0	0	1	1	0	1	3	1	1	1	1	1	1	1	1	8	+5
7	1	0	1	0	0	0	0	0	2	1	1	1	1	1	1	1	1	8	+6

Nota. P. = Participante; 1 = conceito de algoritmo; 2 = conceito de linguagem de programação; 3 = conceito de computador; 4 = conceito de programa de computador; 5 = conceito de variável; 6 = conceito de tipos de dados; 7 = conceito de estrutura de controle de fluxo; 8 = conceito de laço de repetição; Pt = Pontos.

A Tabela 3 apresenta resultados mais promissores do que a Tabela 2, afinal todos os participantes indicaram ganhos. Três reportaram ganhos em cinco a seis comportamentos, outros dois participantes relataram ganhos em dois a três comportamentos e um participante visualiza ganho em apenas um comportamento porque no pré-teste já havia avaliado que possuía conhecimentos em sete dos oito objetivos. Ressaltamos que esse resultado mais positivo pode estar associado ao fato de que essas oito perguntas tratam diretamente de comportamentos previstos entre os objetivos de aprendizagem trabalhados em nosso curso e, especificamente, daqueles que, embora importantes, são os mais simples, pois envolvem definições de conceitos.

A Tabela 4 exibe os resultados relativos ao instrumento com 30 itens de verdadeiro ou falso, que examinava os principais objetivos de aprendizagem do nosso curso. Por serem muitos dados, precisamos sintetizá-los para viabilizar a sua apresentação. Por isso trabalhamos com percentuais de segurança e de acerto, antes e após o curso, sem indicar os resultados de cada um dos 30 itens.

Tabela 4. Resultados nos 30 itens do instrumento avaliativo antes e após o ensino.

Participante	% Segurança nas respostas			% Acertos nas respostas		
	Antes	Após	Diferença	Antes	Após	Diferença
1	100,00	100,00	0,00	93,33	93,33	0,00
2	40,00	66,67	+26,67	60,00	63,33	+3,33
3	83,33	96,67	+13,33	63,33	76,67	+13,33
4	36,67	80,00	+43,33	73,33	66,67	-6,67
5	80,00	83,33	+3,33	60,00	83,33	+23,33
6	66,67	73,33	+6,67	56,67	70,00	+13,33
7	90,00	63,33	-26,67	66,67	66,67	0,00
Média	**70,95**	**80,48**	**+9,52**	**67,62**	**74,29**	**+6,67**
Mediana	**80,00**	**80,00**	**0,00**	**63,33**	**70,00**	**+6,67**
DP	**24,47**	**14,07**	**---**	**12,58**	**10,84**	**---**

Notamos na Tabela 4 que houve uma tendência dos participantes de aumento de segurança nas respostas fornecidas ao instrumento avaliativo, a exceção de P7 que, ao final do ensino, sentiu-se menos seguro nas respostas que forneceu, embora tenha acertado o mesmo número de itens antes e após o ensino. Em média, a melhora em termos de segurança foi sutil (9,52%). Já o valor da mediana sugere que não houve alteração. No

que diz respeito aos acertos, um participante apresentou uma pequena piora (-6,67%), dois participantes não apresentaram melhora e os outros quatro apresentaram aumento no percentual de acertos na comparação entre os momentos antes e após o ensino. Nesse caso, a melhora indicada por média e mediana foi igual (6,67%), sugerindo uma melhora sutil, similar àquela observada em relação à segurança.

Para que pudéssemos ter uma medida do tamanho do efeito da intervenção sobre ganhos em relação a acertos no teste criado, calculamos o *g* de Hedges, com correção para pequenas amostras. Chegamos ao valor de 0,53, que pode ser considerado médio [Espírito-Santo & Daniel, 2015]. No teste de postos com sinais de Wilcoxon, que compara grupos no caso de amostras dependentes, o resultado não foi significativo ($T = 2$; $p = 0,178$). Com relação à segurança, o *g* de Hedges foi de 0,42, que pode ser considerado pequeno, e, novamente, o teste estatístico não foi significativo ($T = 4,5$; $p = 0,248$).

Quando comparamos os resultados de segurança e de acerto, notamos que houve coerência nos dados de P1 (não apresentou ganhos ou perdas em ambos os critérios), P3 (cresceu pouco em ambos) e P6 (cresceu pouco em ambos). Já os dados de P2 (cresceu bem em segurança, mas pouco em acertos), P4 (cresceu muito em segurança, mas apresentou perda nos acertos), P5 (cresceu pouco em segurança, mas conseguiu acertar mais após o ensino) e P7 (piorou em segurança e não mudou em termos de acertos) apresentaram uma variabilidade maior, dificultando uma interpretação inequívoca. Teoricamente, com os reforços oferecidos para os comportamentos que estavam sendo ensinados, deveria ter aumentado o número de acertos e o sentimento de autoconfiança em relação às respostas fornecidas (para mais informações, consulte considerações sobre formação do repertório de autoconfiança em Guilhardi, 2002). Possivelmente, as condições de ensino organizadas não foram reforçadoras a ponto de estabelecer os comportamentos. Pode ser também que comportamentos não ensinados fossem necessários para que esses participantes apresentassem melhora no desempenho.

Ao comparar os dados de diferenças de segurança e de acertos da Tabela 4, com as percepções sobre conhecimento de conceitos da Tabela 3, observamos que P1 foi coerente (indicou pouco ganho de conhecimento e não apresentou ganhos de segurança e acertos). Todos os demais participantes apresentaram padrões diversos. P3 e P6, por exemplo, indicaram ganhos altos de conhecimento, mas isso não foi observado na medida de segurança e em termos de acertos. P2, por sua vez, percebeu-se como conhecendo bem mais sobre os conceitos e demonstrou mais segurança. Contudo, o seu crescimento em acertos não se deu no mesmo grau. Já P5, reportou pouco ganho de conhecimento e demonstrou baixo ganho de segurança, mas aumentou em um grau maior que nesses dois parâmetros os seus acertos. P4 (reportou pouco ganho de conhecimento, demonstrou elevado crescimento em segurança e obteve perda em termos de acertos) e P7 (reportou pouco ganho de conhecimento, demonstrou perda de segurança e não teve diferença de acertos ao final do ensino), por fim, exibiram padrões singulares de resultados.

A Tabela 5 exibe uma nova perspectiva sobre os dados da Tabela 4, pois destaca a diferença de percentuais de segurança e de acertos, nas condições antes e após o ensino, em função dos objetivos de aprendizagem propostos para essa formação. Buscamos reportar as classes mais amplas que pudemos de objetivos, equilibrando perda de informações com excesso de detalhes. Esses dados em função de objetivos de aprendizagem correspondem às médias de desempenho dos participantes.

Tabela 5. Desempenho em função dos objetivos de aprendizagem de maior grau de generalidade.

Objetivo de aprendizagem	% diferença segurança	% diferença acertos
1.1. Diferenciar algoritmo de programa de computador.	+12,70	+7,94
1.2. Definir o conceito de variável.	+12,24	+12,24
1.3. Caracterizar contextos e regras de uso de operadores típicos da programação de computadores.	+23,81	+19,05
1.4. Caracterizar estruturas de controle de fluxo.	-7,14	-7,14
1.5. Formalizar algoritmos na linguagem de programação Python.	+25,00	-10,71
1.6. Identificar finalidade, resultados e erros em programas com base no código.	-19,05	+19,05

No objetivo 1.5 foi identificado baixo desempenho nas avaliações. Uma das possíveis causas, seria falta de atenção por parte dos alunos ao responder o questionário. Essa hipótese é plausível porque, durante o curso, eles se mostraram à vontade e confiantes na formalização dos algoritmos, e demonstraram engajamento nas atividades e interações propostas nos encontros presenciais. Contudo, é possível que no dia da aplicação do pós-teste, os participantes estivessem com pressa em completar o questionário rapidamente, visto que o tempo dedicado por eles para responder foi curto, o que pode ter aumentado as chances de erro. Além disso, é importante lembrar que dados da literatura sugerem que são mais frequentes erros de sintaxe ou de avaliações lógicas mais básicas (podendo, por exemplo, envolver estruturas de controle de fluxo) (Araújo et al., 2021).

Em termos de satisfação, observamos resultados bastante promissores. Todos os participantes atribuíram nota 3 na Questão 12, que examinava a satisfação geral com o curso. Esse é o máximo da escala que adotamos. Quando consideramos as médias de satisfação calculadas a partir do conjunto de 12 itens para todos os participantes, verificamos que o valor foi de 2,83 (*DP* = 013), variando de 2,67 (no caso de P6) a 3,00 (segundo P1). Em todos os cenários, as pontuações foram elevadas, sugerindo que os participantes sentiram satisfação parcial ou total com o curso. O aspecto com menor média de satisfação entre os participantes, ainda que alta (2,43), foi a infraestrutura do laboratório usado para o curso. Os aspectos de destaque, com média de satisfação igual a 3,00, foram a suficiência de informações para decisão sobre participar no curso, relevância do conteúdo, disponibilidade do monitor, clareza da comunicação do monitor e os recursos utilizados para o ensino. Ou seja, a maior parte dos itens com nota máxima foram, justamente, aqueles mais relacionados com as condições de ensino planejadas.

No geral, os dados de percepção de aprendizado, acertos, segurança e satisfação, sugerem que o curso desenvolvido é promissor, embora certamente precise ser aperfeiçoado para ampliar o seu grau de eficiência em termos de ensino. Este estudo teve limitações que precisam ser consideradas em pesquisas futuras, a saber: (1) ampliar o número de participantes; (2) aumentar a diversidade de participantes, incluindo estudantes de dentro e fora da área de computação; (3) aperfeiçoar a descrição de comportamentos e a precisão dos conceitos de problemas de programação "simples" ou "com poucas entradas"; (4) aperfeiçoar o instrumento de medida de acertos, transformando-o em uma medida direta dos comportamentos a que se propõe mensurar. Falamos em medida direta por oposição ao fato de que alguns itens do nosso instrumento avaliam comportamentos como definir ou caracterizar a partir de situações avaliativas nas quais a pessoa deve selecionar uma opção correta e não, concretamente, definir ou caracterizar um conceito, por exemplo. Além disso, identificamos três itens do instrumento que tratam de outras linguagens de programação que não Python, o que pode trazer uma dificuldade desnecessária para a avaliação dos objetivos de aprendizagem propostos neste estudo. Identificamos, ainda, um item[4] que no pré e no pós-teste continha uma dica explícita da resposta correta. Notamos que seis alunos acertaram esse item tanto no Pré, quanto no Pós-teste; (5) aperfeiçoar o instrumento de medida de satisfação no sentido de controlar as respostas em função da desejabilidade social; (6) aperfeiçoar o curso para torná-lo mais eficiente no ensino da sintaxe de Python; e (7) tornar o curso mais interessante de modo a engajar mais as pessoas, afinal conseguimos atrair pouco os alunos calouros, tendo finalizado o curso com apenas sete participantes.

6. Conclusão

O objetivo deste estudo foi avaliar a eficiência de curso em relação ao ensino do comportamento de "escrever programas de computador simples por meio da linguagem de programação Python", buscando investigar se ele promoveu as aprendizagens dos objetivos propostos, se houve melhora na autoconfiança dos alunos em relação às suas respostas em um teste e se o curso foi uma experiência agradável para os aprendizes. Com relação à Pergunta de Pesquisa 1, verificamos que a percepção geral do conhecimento sobre conceitos de programação mudou positivamente para os participantes, sendo que a percepção de ganho de habilidades em programação e Python apresentou uma tendência de melhora para dois a três participantes, tendo permanecido a mesma para a maioria dos aprendizes. Com relação aos acertos no Pós-teste, observamos que houve melhora nos conhecimentos dos participantes. A partir desses dados, podemos responder à Pergunta de Pesquisa 2 no sentido de que o curso não foi eficiente para todos os participantes, um dos motivos pelos quais ele precisa ser aperfeiçoado em versões futuras e testado com mais pessoas, de diferentes áreas do conhecimento. Sobre a Pergunta de Pesquisa 3, observamos que a maioria dos participantes aumentou sutilmente o grau de autoconfiança em relação às respostas que forneceram sobre conceitos e técnicas de programação.

Finalmente, podemos responder à Pergunta de Pesquisa 4 no sentido de que os participantes demonstraram elevado grau de satisfação com o curso. Assim, concluímos que, certamente, o curso é promissor. No entanto, é necessário aperfeiçoar vários aspectos

[4] "1.2.1. Se considerarmos que "I" são os dados do tipo inteiro, "R" os reais, "L" os literais e "B" os lógicos, identifique a alternativa que preencha os espaços dos seguintes dados [...]".

(de planejamento e implementação, envolvendo desde formulação de objetivos de aprendizagem, qualidade psicométrica dos instrumentos de medida e estratégias de promoção de engajamento) para torná-lo mais eficiente para todos os alunos.

Esperamos com este estudo incentivar a condução de mais pesquisas sobre Educação em Computação, especificamente no âmbito do ensino de programação. Para trabalhos futuros, recomenda-se uma divulgação mais abrangente do curso, com um objetivo de atrair mais estudantes, tanto de dentro como de fora da área de computação. A divulgação pode ser feita com um certo tempo de antecipação antes do início do curso, e poderia utilizar as redes sociais como um canal de comunicação para alcançar esse público. Sugere-se também a incorporação de recursos, como juízes *on-line*, que permitam aumentar o volume e diversidade do treino de sintaxe.

Pode ser útil também, para alguns alunos, no intuito de evitar frustrações com erros sucessivos ou elevada dificuldade logo no primeiro contato com a programação, o uso de programação visual antes da inserção de uma linguagem de programação tradicional. Poderia ser usado, por exemplo, ferramentas como o *Blockly*, o Thunkable ou o Scratch. Por meio de um conceito de quebra-cabeça (ou montagem de blocos), essas ferramentas se propõem a facilitar o ensino de como codificar em uma sequência lógica, empregando corretamente sintaxes e, assim, dando vida funcional ao *script*. No *Blockly*, por exemplo, o aprendizado tende a ser divertido porque os alunos precisam guiar um personagem até o seu destino por meio de coordenadas em forma de algoritmos. Essa ferramenta, além de ser um auxílio no ensino, é uma forma de interação com a turma. Isso poderia contribuir para melhorar o engajamento da turma com o curso, outro aspecto que identificamos como necessário de ser aperfeiçoado.

Em suma, é importante que este curso seja continuamente aprimorado, levando em conta as necessidades e *feedbacks* dos alunos. Esse processo iterativo de aperfeiçoamento parece ser a estratégia mais promissora para que o ensino seja o mais eficiente possível, atendendo, assim, aos propósitos de promoção de aprendizagens, autoconfiança e satisfação.

7. Referências

Araujo, A., Filho, D., Oliveira, E., Carvalho, L., Pereira, F., & Oliveira, D. (2021). Mapeamento e análise empírica de misconceptions comuns em avaliações de introdução à programação. In: Anais do Simpósio Brasileiro de Educação em Computação (pp. 123-131). Porto Alegre: Sociedade Brasileira de Computação. https://doi.org/10.5753/educomp.2021.14478

Bittencourt, I. I., & Isotani, S., (2018). Informática na educação baseada em evidências: Um manifesto. *Revista Brasileira de Informática na Educação - RBIE, 26*(3), 108-119. http://dx.doi.org/10.5753/rbie.2018.26.03.108

Botomé, S. P., & Rizzon, L. A. (1997) Medida de desempenho ou avaliação da aprendizagem em um processo de ensino: Práticas usuais e possibilidades de renovação. *Chronos, 30(1)*, 7-34.

Brackmann, C. P. (2017). Desenvolvimento do pensamento computacional através de atividades desplugadas na educação básica (Programa de Pós-graduação em

Informática na Educação, Universidade Federal do Rio Grande do Sul). Recuperado de https://lume.ufrgs.br/handle/10183/172208

Cortegoso, E., & Coser, D. S. (2013). *Elaboração de programas de ensino: Material autoinstrutivo*. São Carlos: Edufscar.

Daly, T. (2011). Minimizing to maximize: an initial attempt at teaching introductory programming using Alice. *Journal of Computing Sciences in Colleges*, *26*(5), 23-30. https://dl.acm.org/doi/10.5555/1961574.1961578

Fiss, R. E., Pereira, T. F., Freitas, D. S., & Ferreira, A. P. L. (2018). A. Pensamento computacional aplicado ao ensino de programação no ensino superior. In: *Anais do Salão Internacional de Ensino, Pesquisa e Extensão* (sem página). Recuperado de https://periodicos.unipampa.edu.br/index.php/SIEPE/article/view/86444

Franzen, E., Bercht, M., & Dertzbacher, J. (2017). Problematização aplicada ao ensino e aprendizagem de algoritmos: Uma análise dos fatores associados à motivação dos estudantes. *RENOTE*, *15*(1), 2-10. https://doi.org/10.22456/1679-1916.75148

Guilhardi, H. J. (2002). Autoestima, autoconfiança e responsabilidade. In: Brandão, M.Z. da S. et al. (orgs.), *Comportamento Humano: Tudo (ou quase tudo) que você precisa saber para viver melhor* (pp. 63-98). Santo André, SP: ESETec Editores Associados.

Holanda D. W., Freire L. P., & Coutinho J. C. S. (2019). Estratégias de ensino-aprendizagem de programação introdutória no ensino superior: Uma Revisão Sistemática da Literatura. *RENOTE*, *17*(1), 527-536. https://doi.org/10.22456/1679-1916.95905

Júnior, D. L. G., & Bogea, D. T. R. (2020). Ensino de programação: uma revisão sistemática e as aplicações ao ensino profissional. *Cadernos da FUCAMP*, *19*(41), 14-30. Recuperado de https://bit.ly/3YGBzut

Kubo, O., & Botomé, S. P. (2001). Ensino-Aprendizagem: Uma interação entre dois processos comportamentais. *Interação*, *5*, 123-132. https://doi.org/10.5380/psi.v5i1.3321

Lazzari, C. L. (2013). *Características da classe de comportamentos 'programar computadores' como parte da capacitação de profissionais da computação* (Dissertação apresentada ao Programa de Psicologia, Universidade Federal de Santa Catarina). Recuperado de https://bit.ly/3m7GrXH

Linhares, L. M. S., Pereira, J. H. H., Ribeiro, P. V. S., Moraes, M. S., & Henklain, M. H. O. (2021). Programação de computadores: A importância da prática. *Cadernos de Extensão, 6, 2, (24-27)*. Recuperado de http://bit.ly/3tHExQF

Machado R. L. (2015). *Uma proposta para o ensino de programação de computadores na Educação Básica* (Trabalho de Conclusão de Curso apresentado à Especialização em Mídias da Educação, Universidade Federal do Rio Grande do Sul). Recuperado de https://lume.ufrgs.br/handle/10183/133856

Medeiros, R. P., Falcão, T. P., & Ramalho, G. L. (2020). Ensino e Aprendizagem de Introdução à Programação no Ensino Superior Brasileiro: Revisão Sistemática da Literatura. In: *Anais do Workshop sobre Educação em Computação (WEI)* (pp. 186-190). Porto Alegre: Sociedade Brasileira de Computação. https://doi.org/10.5753/wei.2020.11155

Ribas, E., Bianco, G. D., & Lahm, R. A. (2016). Programação visual para introdução ao ensino de programação na Educação Superior: Uma análise prática. *RENOTE, 14*(2), 1-10. https://doi.org/10.22456/1679-1916.70671

Scaico, P. D., Lima, A. A., Azevedo, S., Silva, J. B. B., Raposo, E. H., Paiva, L. F., Alencar, Y., Mendes, J. P., & Scaico, A. (2013). Ensino de Programação no Ensino Médio: Uma abordagem orientada ao design com a linguagem Scratch. *Revista Brasileira de Informática na Educação*, *21*(2), 92-103. http://dx.doi.org/10.5753/rbie.2013.21.02.92

Silva, A. C. R., Lima, S. T., & Santos J. C. G. (2022). A lógica de programação na sala de aula: Desafios e contribuições em uma escola pública no município de Serrinha. *Revista educação, linguagem e tecnologias*, *1*(4), 47-69. Recuperado de https://www.revistas.uneb.br/index.php/elite/article/view/15880

Silva, D. N., Brito, J. R., & Vaz, N. A. P. (2019). Lógica de Programação: Dificuldades de ensino-aprendizagem, métodos e ferramentas computacionais. In: Anais do X Simpósio de Tecnologia da Informação, *XI Semana de Iniciação Científica do curso de Sistemas de Informação e IV Colóquio de Estágio* (sem página). Goiás: Universidade Estadual de Goiás. Recuperado de https://www.anais.ueg.br/index.php/sti_sic/article/view/13982

Silva, E., Caceffo, R., & Azevedo, R. (2022). Análise dos tópicos mais abordados em disciplinas de introdução à programação em universidades federais brasileiras. In: *Anais do II Simpósio Brasileiro de Educação em Computação* (pp. 29-39). Porto Alegre: Sociedade Brasileira de Computação. https://doi.org/10.5753/educomp.2022.19196

Silva, T. S. C. (2016). *Um modelo para promover o engajamento estudantil e auxiliar o aprendizado de programação utilizando gamification* (Programa de Pós-graduação em Ciência da Computação, Universidade Federal de Pernambuco). Recuperado de https://repositorio.ufpe.br/handle/123456789/20056

Skinner, B. F. (2003). *The technology of teaching*. Cambridge, MA: The B. F. Skinner Foundation. (Original work published in 1968).

Skinner, B. F. (2005). *Science and human behavior*. Cambridge, MA: The B. F. Skinner Foundation. (Trabalho original publicado em 1953). Recuperado de https://goo.gl/D7yLsb

Todorov, J. C. (2007). A Psicologia como o estudo de interações. *Psicologia: Teoria e Pesquisa*, *23*(número especial), 57-61. Recuperado de http://bit.ly/2OmEqYU

Todorov, J. C., Moreira, M. B., & Martone, R. C. (2009). Sistema personalizado de ensino, educação à distância e aprendizagem centrada no aluno. *Psicologia: Teoria e Pesquisa*, *25*(3), 289-296. https://doi.org/10.1590/S0102-37722009000300002

Unesco. (2014). Learn by coding. Recuperado de https://en.unesco.org/news/learn-coding

Vidotto, K. N. S., Pozzebon, E., Tarouco, L. M. R., & Silva, P. F. (2021). Jogos digitais com realidade aumentada no Ensino Superior: Um projeto para introdução a programação. *Conjecturas*, *21*(3), 692-710. https://doi.org/10.53660/CONJ-183-506

Zanin, A., Sparremberger, A. S., & Barbosa, J. L. V. (2019). Uma proposta de boas práticas para o ensino colaborativo de programação de computadores. *RENOTE*, *17*(3), 41-50. https://doi.org/10.22456/1679-1916.99292

Apêndice 1

Itens do instrumento para avaliação de repertório dos participantes do estudo.

Código, Item e [Gabarito]

1.1. Programas de computador e algoritmos não são sinônimos, embora possuam uma relação entre si. [V]

1.1.1. Algoritmos podem ser definidos como programas que o computador executa. [F]

1.1.1. Sobre algoritmos, analise as afirmativas abaixo e atribua corretamente os valores Verdadeiro (V) ou Falso (F).
( ) São procedimentos precisos, não ambíguos, eficientes e corretos.
( ) Um algoritmo representa os passos necessários para realização de uma tarefa.
( ) São representados somente por fluxogramas.
Agora julgue se o que se afirma a seguir é Verdadeiro ou Falso: A resposta correta é V, V, F. [V]

1.1.1.1. Sempre que executado, sob as mesmas condições, o algoritmo produz os mesmos resultados, que, tipicamente, consistem na solução de um problema. [V]

1.1.2. Programas de computador são desenvolvidos a partir de algoritmos formalizados em uma linguagem de programação. [V]

1.1.2. Identifique a alternativa que apresenta a definição de Programa.
A) Softwares que funcionam como um conjunto de ferramentas e realizam tarefas e trabalhos específicos.
B) Conjunto de softwares cuja função é gerenciar os recursos do sistema.
C) São aqueles que, uma vez instalados, são impossíveis de serem removidos, somente com a formatação.
D) Softwares localizados na barra de ferramentas que restauram arquivos e criam os pontos de restauração.
E) São bibliotecas de códigos que auxiliam no manuseamento do computador.
Agora julgue se o que se afirma a seguir é Verdadeiro ou Falso: A resposta correta é a letra E. [F]

1.1.2.1. Programar consiste no processo de construção de algoritmos. [F]

1.1.3. Sobre a definição de hardware e software em um computador, analise:
I. Hardware é formado pelo conjunto de circuitos eletrônicos e partes eletromecânicas.
II. Software consiste somente nos programas aplicativos; não sendo considerados softwares os Sistemas Operacionais.
III. O Sistema Computacional é formado pelo conjunto de software básico e hardware.
Agora julgue se o que se afirma a seguir é Verdadeiro ou Falso: Estão corretas apenas as afirmativas: I, II. [F]

1.1.3.1. O computador pode ser tecnicamente definido como uma máquina capaz de ser programada para realizar, no contexto de um programa, operações aritméticas e lógicas com os dados fornecidos pelo usuário. [V]

1.2. Um dado é denominado como variável quando existe a possibilidade de ser alterado em algum instante no decorrer do tempo. Identifique a alternativa que apresenta a característica de uma variável.
A) São espaços na memória onde podemos guardar somente valores numéricos de 0 a 9.
B) Uma variável pode ser utilizada durante seu período de vida, de acordo com seu limite de regravações.
C) Identificadores são utilizados para nomear variáveis que representam valores em mutação.
D) Os tipos existentes de variáveis se restringem a variáveis numéricas e caracteres.
E) É possível declarar uma variável iniciando seu nome com um caractere numérico.
Agora julgue se o que se afirma a seguir é Verdadeiro ou Falso: A resposta correta é a letra A. [F]

1.2.1. A frase: "[...] deve caracterizar o conjunto de valores a que uma constante pertence, ou que podem ser assumidos por uma variável ou expressão, ou que podem ser gerados por uma função", de Ziviani (1999), refere-se a: A) Sintaxe de uma estrutura de repetição. | B) Sintaxe de uma estrutura de controle de fluxo. | C) Tipo de dado. | D) Comando de entrada de dados. | E) Comando de saída de dados.
Agora julgue se o que se afirma a seguir é Verdadeiro ou Falso: A resposta correta é a letra C. [V]

1.2.1. Se considerarmos que "I" são os dados do tipo inteiro, "R" os reais, "L" os literais e "B" os lógicos, identifique a alternativa que preencha os espaços dos seguintes dados: ( ) "JOSE" ( ) 0,2132 ( ) +33 ( ) sim | não
Agora julgue se o que se afirma a seguir é Verdadeiro ou Falso: A resposta correta é L - I - R - B. [F]

1.2.1.1. Identifique a alternativa que apresenta a opção que equivale aos dados do tipo Inteiro.

A) 69 | B) True | C) Inteiro | D) " C " | E) 0.111
Agora julgue se o que se afirma a seguir é Verdadeiro ou Falso: A resposta correta é a letra C. [F]

1.2.1.2. O tipo de dados *float* refere-se também aos dados do tipo:
A) caractere | B) inteiro | C) booleano | D) real | E) local
Agora julgue se o que se afirma a seguir é Verdadeiro ou Falso: A resposta correta é a letra E. [F]

1.2.1.3. Dados booleanos são aqueles que se apresentam de modo binário, tais como "sim ou não", "verdadeiro ou falso". [V]

1.2.1.4. Dados *string* são aqueles cuja natureza é numérica, mas que escrevemos com letras. [F]

1.3.1. Operadores aritméticos são aqueles usados para realizar operações matemáticas. Exemplos: < (menor) e > (maior). [F]

1.3.2. Operadores relacionais são aqueles que nos auxiliam a realizar comparações entre valores de variáveis. Exemplos: == (igual) e =! (diferente). [V]

1.3.3. Operadores lógicos são aqueles que nos permitem realizar operações booleanas entre variáveis. Exemplos: AND e OR. [V]

1.4. Identifique a alternativa que contém uma estrutura de controle que permite que a execução de um trecho de programa dependa do fato de uma condição ser verdadeira, isto é, vinculada à execução de um ou mais comandos ao resultado obtido na avaliação de uma expressão lógica (também denominada condicional).
A) Seleção simples. | B) Seleção dupla. | C) Comando composto. | D) Múltipla escolha. | E) Seleção aninhada.
Agora julgue se o que se afirma a seguir é Verdadeiro ou Falso: A resposta correta é a letra B. [F]

1.4.2.1. Nas linguagens de programação, existem diversos tipos de instrução. Analisando o pseudocódigo a seguir, identifique a alternativa que define o tipo de instrução que realiza sua operação. Pseudocódigo: "Enquanto houver maçãs podres no cesto, retire-as e jogue fora".
A) Instrução de seleção | B) Instrução de repetição. | C) Instrução de seleção múltipla. | D) Instrução de evolução. | E) Instrução de decremento.
Agora julgue se o que se afirma a seguir é Verdadeiro ou Falso: A resposta correta é a letra B. [V]

1.4.2.1. Relacione as estruturas de controle empregadas em algoritmos e programas de computador com suas respectivas características: repita... ate... fimrepita [REP] | enquanto ... faca... fimenquanto [ENQ]
( ) O teste de controle é realizado no fim da estrutura de controle.
( ) O teste de controle é realizado no início da estrutura de controle.
( ) A condição de saída do loop ocorre quando o teste é FALSO.
( ) A condição de saída do loop ocorre quando o teste é VERDADEIRO.
( ) Se o resultado do teste for FALSO, a execução do programa permanece no loop.
( ) Se o resultado do teste for VERDADEIRO, a execução do programa permanece no loop.
Agora julgue se o que se afirma a seguir é Verdadeiro ou Falso: A resposta correta é REP – ENQ – REP – ENQ – ENQ – REP. [F]

1.4.2.2. Um analista de sistema MJSP precisa codificar uma instrução 'for' que varie a variável de controle de 7 a 77 em incrementos de 7. Escolha a sentença correta referente a essa instrução 'for'.
A) `for (int i = 0; i <= 77; i += 7)` | B) `for (int i = 7; i <= 77; i += 7)` | C) `for (int i = 1; i == 77; i + 7)` | D) `for (int i = 0; i <= 77; i++)` | E) `for (int i = 7; i >= 0; i++)`
Agora julgue se o que se afirma a seguir é Verdadeiro ou Falso: A resposta correta é a letra A. [F]

1.5.1. Os passos que usamos para realizar uma operação de divisão entre dois números podem ser classificados como um algoritmo. [V]

1.5.1.1. Considere o seguinte algoritmo:
recebe valor; divide valor por 2; avalia se o resto da divisão é igual a 0; se o resto for igual 0, imprimir que o valor é par; caso contrário, imprimir que o valor é ímpar. Em Python, este algoritmo poderia ser traduzido da seguinte forma:

```
valor = int(input("Informe o valor a ser avaliado: "))
if valor%2 == 0:
       print("O número",valor,"é par.")
else:
       print("O número",valor,"é ímpar.")
```

[V]

1.5.1.1. Considere a tarefa de avaliar 3 números quaisquer para, então, imprimir o menor deles.
Em Python, este programa poderia ser escrito da seguinte forma:

```
n1 = int(input("n1 "))
n2 = int(input("n2 "))
n3 = int(input("n3 "))
if n1 > n2:
       menor = n2
else:
       menor = n1
if menor > n3:
       menor = n3
print(menor)
```

Alternativamente, também poderia ser escrito da seguinte forma:

```
n1 = int(input("n1 "))
n2 = int(input("n2 "))
n3 = int(input("n3 "))
if (n2 < n1):
       n1 = n2
if (n3 < n1):
       n1 = n3
print(n1)
```

[F]

1.5.1.1.1. Considere o seguinte algoritmo para descobrir se um ano é bissexto, lembrando que ele está testando apenas uma condição, a saber: que um ano pode ser bissexto se for divisível por 4 e não por 100.
recebe ano; divide ano por 4; avalia se o resto da divisão é igual a 0; divide ano por 100; avalia se o resto da divisão é diferente de 0; se o resto da divisão por 4 for 0 e o resto da divisão por 100 não for, imprimir que o ano é bissexto caso contrário, imprimir que o ano não é bissexto. Em Python, este algoritmo poderia ser traduzido da seguinte forma:

```
ano = int(input("Informe o ano: "))
if ano%4 == 0 or ano%100 != 0:
       print("O ano", ano, "é bissexto")
else:
       print("O ano", ano, "não é bissexto")
```

[F]

1.6. Siga as instruções abaixo e identifique o valor final que deve ser exibido nas células A[], B[], C[] e D[].
Instruções: 1 - Coloque o valor 3 na célula D; 2 - Coloque o valor 8 na célula A; 3 - Copie o valor da célula D para a célula B; 4 - Copie o valor da célula A para a célula C; 5 - Some os valores da célula B e da célula C e coloque o resultado na célula A; 6 - Subtraia 1 ao valor da célula D e guarde o resultado na célula D; 7 - Repita as instruções 3, 4, 5, 6 até que o valor da célula D seja menor ou igual a 0.
Avalie a afirmação como V ou F: Os valores finais em cada célula são A[14], B[1], C[13] e D[0]. [V]

1.6. Considere o algoritmo abaixo desenvolvido segundo a sintaxe utilizada no VisuAlg 3.0.

```
Algoritmo "Gera um valor"
Var
    dens : vetor [1..4] de inteiro
    i, f, s, x, o : inteiro
Inicio
    f <- 1
    para i de 1 até 4 faça
        dens[i] <- f
        f <- f * (i + 1)
    fimpara
    x <- 2
    s <- 0
    p <- x
    para i de 1 até 4 faça
        p <- p * x
        se i mod 2 = 1 então
            s <- s - p \ dens[i]
        senão
```

```
            s <- s + p \ dens[i]
        fimse
    fimpara
    escreva (s)
Fimalgoritmo
```

Agora julgue se o que se afirma a seguir é Verdadeiro ou Falso: A resposta correta é -1. [V]

1.6. Sendo A1 = 5, B1 =7 e C1 = 1, o valor de C1 nas seguintes linhas de comando será:

```
Se (A1 > B1) Ou (A1 = B1)
    Faca C1 = 0
Senao
    Se (A1 < B1) e (C1 > 0)
        Faca C1 = 1
```

Agora julgue se o que se afirma a seguir é Verdadeiro ou Falso: A resposta correta é 1. [V]

Nota. Os códigos que antecedem cada enunciado, correspondem aos números usados na Tabela 1 para identificação dos objetivos de aprendizagem. Isso indica que a questão busca avaliar o objetivo de aprendizagem indicado pelo código.