ISSN: 2358-8829

O ERRO COMO RECURSO PEDAGÓGICO: ANÁLISE DE ANTIPADRÕES NO ENSINO INICIAL DE PROGRAMAÇÃO EM PYTHON

Italo Fabricio Pereira de Souza [1]
Ana Liz Souto Oliveira [2]

INTRODUÇÃO

Aprender a programar é um desafio para estudantes, principalmente para aqueles que nunca tiveram contato com linguagem de programação antes. Escrever algoritmos exige o conhecimento de uma sintaxe nova e não tradicional, raciocínio lógico e habilidade de estruturar soluções para resolver problemas de uma forma clara e eficiente (BOSSE, 2020). Dessa forma, surge a motivação para estudar formas de ensinar e aprender programação, entendendo como as dificuldades enfrentadas pelos alunos contribuem para os altos índices de reprovação e desistência em disciplinas introdutórias (BENNEDSEN; CASPERSEN, 2019; WATSON; LI, 2014).

Outros obstáculos enfrentados no aprendizado de programação incluem cursos que não são bem estruturados, correções inadequadas de exercícios práticos, bem como a ausência de suporte (MHASHI; ALAKEEL, 2013). Além disso, conhecer os antipadrões, ou seja, os erros frequentes cometidos por estudantes em processo de aprendizado, pode fornecer uma percepção de como aprimorar o ensino de programação. A análise de antipadrões é uma atividade de pesquisa relevante, pois permite identificar obstáculos recorrentes no aprendizado e sugerir estratégias para superá-los (BOSSE, 2020; BROWN et al., 1998).

Compreender e analisar os erros cometidos pelos discentes ao programar é uma estratégia fundamental para aprimorar o ensino da disciplina. O erro, quando devidamente estudado, pode ser um recurso valioso para a aprendizagem, pois permite que estudantes e professores identifiquem padrões de dificuldades e encontrem abordagens mais eficazes para superá-las. Segundo Brown *et al.* (1998), um antipadrão é uma solução comum a um problema que, em vez de resolvê-lo adequadamente, gera consequências negativas, e surge,

[1]Graduando do Curso de Licenciatura em Ciência da Computação da Universidade Federal da Paraíba - UFPB,italofps65@gmail.com;
[2] Professora Doutora do curso de Licenciatura em Ciência da Computação da Universidade Federal da Paraíba - UFPB, analiz@dcx.ufpb.br.

ISSN: 2358-8829

muitas vezes, por falta de conhecimento ou pela aplicação inadequada de um conceito ou padrão correto dentro de um contexto errado.

No ensino de programação, a análise de antipadrões permite compreender erros recorrentes cometidos por iniciantes e desenvolver estratégias que ajudem a preveni-los. De acordo com Bosse (2020), um erro só pode ser considerado um antipadrão quando ocorre pelo menos três vezes em situações distintas. Dessa forma, a catalogação de antipadrões possibilita a construção de um repertório de más práticas que, quando compreendidas e corrigidas, podem evitar dificuldades futuras no aprendizado. Além disso, o estudo de antipadrões vai além da mera identificação de erros isolados, pois fornece informações sobre como esses equívocos afetam a qualidade do código e a compreensão dos alunos sobre os conceitos fundamentais da programação (BOSSE, 2020).

Nesse contexto, este estudo tem por objetivo identificar a frequência dos padrões de erros cometidos por graduandos ao escrever códigos Python na disciplina de Introdução à Programação de um curso de Ciência da Computação. Dessa forma, respondemos a questão de pesquisa: quais os erros mais comuns cometidos por estudantes quando estão aprendendo a programar em Python? Para isso, adotou-se como referência o catálogo de antipadrões de erros em Python proposto por Bosse (2020) e foram analisados e classificados os erros encontrados em três avaliações escritas de uma turma, aplicadas ao longo de um semestre letivo.

Nosso referencial teórico se baseia no estudo das dificuldades enfrentadas por estudantes que estão iniciando em cursos de programação, pois tem sido um campo de análise explorado para entender melhor os desafios e as possíveis melhorias no ensino e na aprendizagem de programação. Piwek e Savage (2020) investigaram os desafios enfrentados por estudantes do primeiro ano do ensino a distância ao aprender a programar em Python. Os autores coletaram dados ao longo do semestre por meio de um fórum online, no qual os alunos discutiam suas dificuldades em um ambiente de suporte entre pares, e observaram que o fórum era usado não apenas para esclarecer dúvidas técnicas, mas também para oferecer suporte emocional e motivacional. Isso reforçou a importância do aprendizado colaborativo e do suporte social no ensino de programação, especialmente a distância.

O estudo de Bosse (2020) teve como objetivo identificar padrões de dificuldades relacionadas ao aprendizado de programação nas linguagens C e Python, por meio de entrevistas, análise de diários de classe, questionários e análises de código. A pesquisa foi organizada em três etapas: (1) levantamento inicial da situação do ensino de programação introdutória na Universidade de São Paulo (USP); (2) validação, com alunos e professores, do

ISSN: 2358-8829

que entendem como problemas no ensino e na aprendizagem de programação; e (3) análise dos códigos de exercícios desenvolvidos por alunos durante seus cursos. Sua principal contribuição foi um conjunto de antipadrões de erros cometidos por estudantes na disciplina de introdução à programação, relacionados às linguagens C e Python.

Garner et al. (2005) investigaram os desafios enfrentados por estudantes em um curso introdutório de programação em Java, analisando os registros das solicitações de ajuda feitas durante as sessões de laboratório, nas quais os estudantes resolviam exercícios pré-definidos e podiam solicitar suporte aos monitores. O resultado apontou que os conceitos de classes auxiliares (*worker classes*), a manipulação de objetos e as estruturas de controle, como *loops* e condicionais, foram pontos críticos na aprendizagem.

Já a pesquisa de Mhashi e Alakeel (2013) buscou entender as dificuldades dos estudantes ao longo do aprendizado de programação em C#. Os autores aplicaram um questionário que coletou informações sobre o conteúdo do curso, o ambiente de programação, o ambiente do aluno, aspectos do ensino, materiais e questões relacionadas ao professor. Os resultados mostraram que os alunos não consideram a sintaxe da linguagem o maior problema, mas sim a fase de planejamento dos programas, e que o ambiente de aprendizado, os materiais didáticos e a didática do professor tiveram impacto significativo no desempenho.

METODOLOGIA

A metodologia deste estudo baseou-se na identificação e na categorização da frequência de erros comuns na aprendizagem de programação na disciplina de Introdução à Programação em Python de um curso de Ciência da Computação. Após uma revisão de literatura, foi selecionado um catálogo de antipadrões de erros em Python de aprendizes de programação (BOSSE, 2020). Em seguida, foram obtidos códigos em Python oriundos de provas de discentes que cursaram a disciplina em 2023. Os códigos analisados foram escritos à mão em provas escritas e, por esse motivo, os erros foram classificados manualmente de acordo com o catálogo. O processo foi organizado em quatro fases.

Fase 1 – Seleção e adaptação do catálogo de antipadrões. O catálogo adotado apresenta os padrões de dificuldades relacionadas ao aprendizado de programação (BOSSE, 2020), com 12 tópicos de equívocos de escrita de código. Como o estudo original abordava erros tanto da linguagem C quanto de Python, foram selecionados apenas os tópicos relacionados a Python e, ainda, limitados àqueles contemplados nas provas analisadas.

ISSN: 2358-8829

Fase 2 – Coleta e organização dos dados. Os padrões de erros foram organizados em uma tabela no *Google Sheets*, em que as colunas indicam os padrões individualmente e as linhas representam os discentes. Em vez de marcar "X" para indicar a presença de erros, como no estudo original de Bosse (2020), optou-se por atribuir valores binários, em que 0 representa a ausência de erro em uma categoria e 1 indica a sua ocorrência. Os dados foram obtidos de três avaliações aplicadas ao longo do semestre letivo de 2023-1. Para cada avaliação, havia quatro versões de prova, que se diferenciavam pela ordem das questões, pelos valores e pelos enunciados (ligeiramente) diferentes, mas seguiam o mesmo padrão de dificuldade e de assuntos abordados. Essa variação foi planejada pelo professor da disciplina para garantir a integridade das avaliações, evitando o compartilhamento de respostas.

Quanto ao conteúdo em Python, a avaliação 1 abordou tópicos introdutórios, incluindo Variável, Problemas de Raciocínio Lógico, Função de Entrada e Saída de Dados, Expressão, Expressão Aritmética, Expressão Relacional e Estrutura de Seleção. A avaliação 2 contemplou todos os assuntos anteriores e mais a Estrutura de Repetição *while*. A avaliação 3 abordou todos os assuntos anteriores e mais a Estrutura de Repetição *for*, Listas e Sublistas.

Fase 3 – Classificação e análise dos erros. As avaliações impressas com as respostas dos estudantes foram escaneadas e armazenadas no *Google Drive*. Cada questão foi estruturada para avaliar o conhecimento dos estudantes nos assuntos da respectiva prova, e a correção dos erros foi explicada pela professora da disciplina, de forma manual, na própria prova, o que permitiu uma análise detalhada dos tipos de erros cometidos e facilitou a classificação de cada um nos tópicos de equívocos. Para a análise, foi produzida uma rubrica específica, elaborada a partir das respostas dos estudantes que obtiveram nota máxima e validada por meio da correção das provas, da codificação das respostas e de testes com o interpretador Python quando cabível. A partir dessa rubrica, todas as demais respostas foram analisadas e categorizadas.

Nas questões que exigiam justificativa, caso o aluno marcasse a opção correta, mas não a justificasse adequadamente, a resposta foi considerada errada, por se entender que se tratava de palpite (acerto ao acaso). Nesses casos, o erro foi atribuído à categoria de equívoco equivalente ao tópico da questão. Para os estudantes que não responderam às avaliações, deixaram questões sem resposta ou as entregaram incompletas, foi feita uma marcação específica de ausência. O processo de correção seguiu as etapas de: aplicação da rubrica em cada questão; comparação das respostas com os critérios da rubrica; identificação dos erros por meio das explicações da professora e registro em planilha; e classificação dos erros não previstos nos tópicos do catálogo nos equívocos mais próximos.

ISSN: 2358-8829

Durante a análise, notou-se a necessidade de ajustar e especificar a categorização, adaptando algumas categorias e acrescentando outras. No tópico de Expressão Aritmética, foi acrescentada a categoria "Realizar cálculo utilizando variáveis erradas" e ajustada a categoria antes denominada "Tipo do resultado float em uma divisão de inteiros" para "Realizar cálculo utilizando tipos diferentes", como o tipo float em uma divisão de inteiros ou a soma de string e float. No tópico de Lista, foram incluídas as categorias "Escrita incorreta para exibir o tamanho", "Escrita incorreta para adicionar valor na lista e sublista" e "Escrita incorreta para exibir valores da lista". Para a avaliação 3, a estrutura *while* foi acrescentada à categoria "Argumentos errados" do tópico de Estrutura de Repetição, uma vez que essa avaliação abordou tanto while quanto for.

Fase 4 – Análise estatística e visualização. Após cada prova ser analisada individualmente, os erros foram registrados em uma planilha do *Google Sheets*, na qual cada aluno recebeu um código identificador único, garantindo a confidencialidade e o anonimato. Os erros foram categorizados conforme o tipo e o assunto da questão, e os dados foram organizados para permitir a análise estatística descritiva.

Quanto aos cuidados éticos, o estudo tem caráter documental e retrospectivo, sem coleta direta de dados sensíveis e sem identificação dos participantes. Os dados foram obtidos exclusivamente de provas escritas aplicadas há dois anos, no contexto regular de uma disciplina de Introdução à Programação, e tratados de forma anônima, apenas para fins acadêmicos e científicos e sem qualquer julgamento de desempenho individual. Os dados foram sistematizados em planilhas digitais de acesso restrito à equipe de pesquisa, sem intervenção no processo avaliativo dos estudantes e sem exposição pública de dados individuais.

RESULTADOS E DISCUSSÃO

Os dados analisados foram obtidos de três provas escritas aplicadas ao longo do semestre letivo de 2023. Ao todo, foram corrigidas 76 provas da Avaliação 1, 74 da Avaliação 2 e 35 da Avaliação 3. Os resultados estão organizados por assunto e pela classificação dos antipadrões de erro.

Variáveis e indentação. O erro mais comum nessa categoria foi "Falta de conversão de tipo", seguido de "Falta de indentação" e "Uso de variáveis inexistentes". Ao longo das avaliações, nota-se uma tendência geral de redução dos erros. No entanto, o antipadrão "Falta de conversão de tipo" teve uma leve variação, com redução na avaliação 2, mas aumento na

ISSN: 2358-8829

avaliação 3. Esse comportamento pode estar relacionado à dificuldade dos estudantes em interpretar corretamente os enunciados e compreender exatamente o que era solicitado quando o assunto envolvia Listas. A realização das provas no papel também pode ter contribuído, pois, diferentemente da execução de um programa no computador, os alunos não contavam com o retorno imediato de um interpretador.

Entrada e saída. O antipadrão "Escrita incorreta do *input*" foi o de maior frequência na avaliação 1, enquanto o erro "Não apresentar o resultado ao usuário" teve maior ocorrência nas avaliações 2 e 3. Isso pode estar relacionado ao fato de que as questões posteriores exigiam raciocínio mais elaborado, sendo mais extensas e complexas do que as da avaliação 1. Essa maior carga de raciocínio pode ter feito os estudantes focarem na lógica do problema e se esquecerem de incluir a saída de dados ao final da questão. Observou-se que mesmo estudantes que estruturaram corretamente a lógica do programa deixavam de exibir a resposta ao usuário, o que indica que esse erro pode ter decorrido da falta de revisão ao final da resolução.

Expressões lógicas e aritméticas. Os problemas mais frequentes foram "Operador aritmético errado", principalmente na avaliação 2, que mais cobrou o assunto de estrutura de decisão, e "Mistura de comparações e operadores", na avaliação 1. É possível perceber que os alunos apresentaram dificuldades ao construir expressões lógicas corretamente dentro da sintaxe da linguagem. Esse comportamento está alinhado com as observações de Bosse (2020), segundo as quais os estudantes frequentemente tentam escrever expressões lógicas utilizando a mesma estrutura da linguagem natural ou da matemática, como "x > 3 e < 5" em vez de "x > 3 e x < 5", o que gera erros de sintaxe e de lógica. Dificuldade semelhante foi apontada por Hristova et al. (2003), que identificaram confusão entre o operador de atribuição "=" e o de comparação "==" ao escrever condicionais e laços. Esse mesmo tipo de erro foi observado nas provas, principalmente nas ocorrências mais altas da segunda e da terceira avaliações.

Estrutura de decisão. Na avaliação 1, os erros mais frequentes foram "*if* sem condição ou com a condição em lugar errado", que ocorreu em 42,11% (32/76) das respostas, seguido por "O não uso do else onde seria adequado", em 34,21% (26/76) dos casos. Esses erros diminuíram nas avaliações seguintes. Os resultados sugerem que os alunos tinham dificuldades na estruturação correta dos comandos condicionais, principalmente na definição das condições de um *if* e na compreensão do papel do *else* dentro do fluxo de decisão. Foi possível perceber, ainda, que esses erros estavam conectados a dificuldades em outros tópicos, como o uso de variáveis erradas ou inexistentes nas condições, a escrita incorreta de

ISSN: 2358-8829

expressões aritméticas e lógicas e a confusão entre operadores como *or* e *and*, "=" e "==", e ">=" e "<=", o que reforça a importância de uma compreensão ampla e interligada dos conteúdos.

Estrutura de repetição. Conteúdo introduzido a partir da segunda avaliação (com while) e da terceira (com for). Na segunda avaliação, os erros relacionados ao while incluíram "Argumentos errados para o *while/for*" em 50% (37/74) das respostas, seguido por "Ordem incorreta dos comandos na estrutura", em 43,24% (32/74) dos casos. Os erros "Condição de parada incompleta" e "Não muda valor de variável dentro da repetição, necessário para chegar ao resultado" ocorreram com a mesma frequência, 31,08% (23/74) cada, e "Variável de controle não está sendo alterada" esteve presente em 18,92% (14/74) das respostas. Esses números indicam que a principal dificuldade foi compreender como estruturar corretamente um laço *while* para que funcionasse dentro das condições esperadas. Um ponto observado foi a ocorrência frequente de loops infinitos, pois os alunos esqueciam de atualizar a variável de controle ou definiam condições que nunca seriam satisfeitas.

Na terceira avaliação, com o uso do *for*, houve redução significativa em quase todas as categorias de erro. Os erros de "Argumentos errados para o for ou while" ainda apareceram com frequência alta, em 58,82% (20/34) das ocorrências, enquanto "Condição de parada incompleta" caiu para 14,71% (5/34) e "Não muda valor de variável dentro da repetição, necessário para chegar ao resultado" foi registrado em 20,59% (7/34) das respostas. Essa avaliação incluiu questões em que era preciso reescrever um código com *for* de forma equivalente com while e outras envolvendo listas e sublistas, o que tornou os exercícios mais desafiadores. Essa mudança pode ter contribuído para a alteração no perfil dos erros: em vez de *loops* infinitos, os alunos passaram a ter mais problemas com a passagem de argumentos corretos para os laços de repetição, principalmente ao definir intervalos e condições, especificamente nas questões com listas e sublistas.

Listas e sublistas. A terceira avaliação introduziu esses conteúdos, e os estudantes apresentaram dificuldades na manipulação desses elementos, o que resultou na criação de novas categorias de erro para melhor catalogação dos equívocos. O erro mais recorrente foi "Escrita incorreta para exibir valores da lista", em 67,65% (23/34) das respostas, seguido de "Criação errada das listas", em 58,82% (20/34), "Escrita incorreta para adicionar valor na lista ou sublista", em 52,94% (18/34), e "Escrita incorreta para exibir o tamanho", em 41,18% (14/34) dos casos. Uma possível explicação é que a manipulação de listas exige um entendimento integrado de conceitos prévios, como variáveis, estruturas de repetição e lógica condicional, de modo que os alunos que já tinham dificuldades nesses tópicos podem ter

ISSN: 2358-8829

encontrado ainda mais dificuldade ao aplicá-los de forma conjunta. Isso está alinhado com a observação de Bosse (2020) sobre a relação entre diferentes tópicos na programação, na qual muitos alunos relatam dificuldades em organizar a ordem correta das ações ao escrever um código.

A análise sugere que a combinação entre prática contínua e *feedback* detalhado foi crucial para ajudar os estudantes a superarem suas dificuldades iniciais. No entanto, os resultados também destacam a importância de trabalhar os fundamentos de forma mais integrada desde o início do curso, garantindo que os alunos desenvolvam não apenas o conhecimento técnico, mas também a capacidade de aplicá-lo de forma coerente e sistemática.

Nesse sentido, destaca-se a relevância da realização de provas escritas no papel, especialmente para alunos iniciantes. Esse tipo de avaliação os força a organizar mentalmente suas ideias antes de codificar, promovendo o desenvolvimento do raciocínio lógico e da capacidade de interpretação de enunciados. Ao estruturar algoritmos sem o apoio imediato de um interpretador, os estudantes são desafiados a refletir sobre a lógica por trás de suas soluções e a consolidar seu entendimento dos conceitos fundamentais, o que estimula uma abordagem mais consciente e estratégica da programação e favorece a construção de uma base sólida para conteúdos mais avançados.

Um aspecto relevante foi a dificuldade dos estudantes em questões que exigiam raciocínio lógico mais complexo. Ao examinar as resoluções, notava-se a intenção por trás da solução proposta pelo aluno, mas frequentemente faltava coerência na estruturação do algoritmo. Estudantes sem domínio de conceitos básicos, como variáveis, expressões lógicas e fluxo de execução, mostraram-se especialmente vulneráveis a esses problemas. Essa tendência foi confirmada pelas anotações da professora durante a correção, que frequentemente questionava escolhas inconsistentes de variáveis ou o desenvolvimento incompleto das soluções. Vale ressaltar que muitos desses erros, em um ambiente de programação real, impediriam a execução do código, mas foram considerados parcialmente corretos nas avaliações escritas, pois revelavam um entendimento conceitual mesmo quando mal aplicado na prática.

CONSIDERAÇÕES FINAIS

O objetivo deste estudo foi analisar com que frequência graduandos da disciplina de Introdução à Programação de um curso de Ciência da Computação cometem erros ao programar em Python. Esse objetivo foi atingido por meio da análise de três avaliações, da

identificação dos erros mais comuns e de sua categorização de acordo com o catálogo de antipadrões proposto por Bosse (2020), buscando compreender as principais dificuldades dos estudantes e como elas se manifestam no processo de aprendizagem.

Os resultados apontaram que os três erros mais frequentes foram: i) erros na manipulação de listas e sublistas, como escrita incorreta para exibir valores ou adicionar elementos à lista; ii) equívocos em expressões aritméticas, como realizar cálculos utilizando variáveis erradas ou diferentes tipos de dados de forma inadequada; e iii) erros na utilização de estruturas de repetição, como argumentos incorretos em estruturas de *loops, for* e *while*. Esses resultados reforçam estudos anteriores que indicam que as principais dificuldades dos iniciantes não estão apenas na sintaxe da linguagem, mas também no entendimento e na aplicação de conceitos fundamentais, pois eles enfrentam dificuldades ao avançar para um novo conteúdo que exige conhecimento prévio de um assunto anterior cuja aprendizagem não foi solidificada.

Ao categorizar os erros com base em um catálogo consolidado de antipadrões, esta pesquisa oferece um retrato das dificuldades enfrentadas por graduandos ingressantes ao programar em Python e fornece uma base empírica para que docentes planejem intervenções pedagógicas mais direcionadas, ajustem suas metodologias, elaborem materiais focados nas principais fragilidades e desenvolvam estratégias que antecipem e mitiguem esses erros. Além disso, os resultados dialogam com os esforços da área de Ensino de Computação em promover abordagens mais centradas no estudante.

Como trabalhos futuros, planeja-se ampliar o estudo para outras disciplinas que dependem dos fundamentos de programação, como as posteriores à introdução à programação e estruturas de dados, a fim de verificar se os mesmos antipadrões se manifestam em contextos mais avançados ou se outros emergem.

Palavras-chave: Ensino de programação, Antipadrões de erros, Python, Introdução à Programação, Ensino de Computação.

REFERÊNCIAS

BENNEDSEN, J.; CASPERSEN, M. Failure rates in introductory programming: 12 years later. **ACM Inroads**, v. 10, n. 2, p. 30-36, 2019.

BOSSE, Y. **Padrões de dificuldades relacionadas com o aprendizado de programação**. 2020. 270 f. Tese (Doutorado) – Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2020.

BROWN, W. H.; MALVEAU, R. C.; MCCORMICK, H. W.; MOWBRAY, T. J. **AntiPatterns**: refactoring software, architectures, and projects in crisis. [S. l.]: John Wiley & Sons, 1998.

GARNER, S.; HADEN, P.; ROBINS, A. My program is correct but it doesn't run: a preliminary investigation of novice programmers' problems. In: AUSTRALASIAN CONFERENCE ON COMPUTING EDUCATION, 7., 2005. **Proceedings** [...]. v. 42. Australian Computer Society, 2005. p. 173-180.

MHASHI, M.; ALAKEEL, A. Difficulties facing students in learning computer programming skills at Tabuk University. In: INTERNATIONAL CONFERENCE ON EDUCATION AND EDUCATIONAL TECHNOLOGY, 12., 2013, Iwate, Japan. **Proceedings** [...]. 2013. p. 15-24.

PIWEK, P.; SAVAGE, S. **Challenges with learning to program and problem solve**: an analysis of student online discussions. 2020.

WATSON, C.; LI, F. Failure rates in introductory programming revisited. In: CONFERENCE ON INNOVATION & TECHNOLOGY IN COMPUTER SCIENCE EDUCATION, 2014. **Proceedings** [...]. ACM, 2014. p. 39-44.