Análise dos erros mais comuns de aprendizes de programação que utilizam a linguagem Python

Galileu Santos de Jesus[1], Kleber Tarcísio Oliveira Santos[1], Jaine da Conceição Santos[1], Alberto Costa Neto[1].

[1]Departamento de Computação – Universidade Federal de Sergipe.
Av. Marechal Rondon, Jardim Rosa Elze, São Cristóvão - SE, 49100-000.

galilasmb@gmail.com, klebertarcisio@yahoo.com.br

jainecs@dcomp.ufs.br, alberto@ufs.br

Abstract. *This paper presents the results of an analysis of the user database of the on-line judge The Huxley, illustrating run-time errors – which by using a Python interpreter also reports the most common syntactic errors – committed by the programming learners of the Python programming language. The results of the analysis of the base point out that there are 143 types of errors distributed in 20 classes and that the most common errors are of invalid syntax, when the student misuses some commands, such as if, for, while, input and print and forgets the token ':' of the conditional command if, variable declaration, when using a variable that was not defined previously and type errors when operations are performed with different types of data. This compilation of most common errors is of paramount importance for teachers to emphasize them during their introductory programming classes, since they occur continuously.*

Resumo. *Este trabalho apresenta os resultados de uma análise da base de dados dos usuários do juiz on-line The Huxley, ilustrando os erros em tempo de execução – que por usar um interpretador de Python também reporta os erros sintáticos – mais comuns cometidos pelos aprendizes de programação da linguagem de programação Python. Os resultados da análise da base apontam que existem 143 tipos de erros, distribuídos em 20 classes e que os erros mais comuns são de sintaxe inválida, quando o aluno faz o uso incorreto de alguns comandos, como if, for, while, input e print e esquece o token ':' do comando condicional if, declaração de variáveis, quando é feito o uso de uma variável que não foi definida anteriormente e erros de tipos, quando são realizadas operações com tipos diferentes de dados. Esta compilação de erros mais comuns é de suma importância para que os professores os enfatizem durante suas aulas introdutórias de programação, visto que ocorrem continuadamente.*

1. Introdução

Ensinar programação de computadores é uma tarefa complexa (WEBER; BRUSILOVSKY; STEINLE, 2014), que exige não somente do professor, pois é necessário que o aluno possua uma certa habilidade. Há uma forte indicação de que o conhecimento anterior sobre a resolução de problemas dentro do contexto anterior à computação é um pré-requisito importante no aprendizado de linguagens de programação (LEMOS; BARROS; LOPES, 2003). No caso dos alunos, estes costumam ter muita dificuldade em aplicar suas

habilidades prévias, apresentando-se como fonte de medo e frustração, segundo Júnior e Rapkiewicz (2004).

Várias são as dificuldades enfrentadas pelos alunos no decorrer do processo de ensino-aprendizagem de programação. O trabalho de (LAHTINEN; ALA-MUTKA; JÄRVINEN, 2005) cita que a retirada de erros, modularização e conceitos como recursão e ponteiros são algumas das dificuldades. Todavia, a maioria está relacionada à combinação e à utilização apropriada dos conceitos básicos de programação, de acordo com Caspersen e Kolling (2009). Segundo Miyadera, Huang e Yokoyama (2000) uma das maiores dificuldades de um estudante de programação é entender cada passo da execução do programa. Assim como entender e aprender a gramática de uma linguagem de programação, consertar erros de sintaxe em um programa, desenvolver novos algoritmos, escrever um novo programa, depurar e consertar erros em um programa, em ordem crescente de dificuldade. Assim, os alunos parecem entender os conceitos e estruturas que compõem uma linguagem de programação, mas não sabem como utilizar corretamente, ou seja, entendem como é a parte léxica de uma linguagem, mas não sabem como utilizar de forma correta a parte sintática ou semântica.

Em turmas de introdução à programação, muitas vezes os conceitos básicos são os mesmos, fazendo com que os alunos cometam os mesmos erros de programação de todo iniciante. Diante do grande número de turmas, a replicação de aulas e quantidade de alunos, é inviável que o professor consiga fornecer um acompanhamento individual dos alunos sobre os erros que são cometidos.

Na literatura, alguns pesquisadores catalogaram alguns erros mais comuns de programação. O trabalho de Schorsch (1995) relata 111 mensagens de diagnóstico diferentes (incluindo 62 para erros de sintaxe, 21 para erros de lógica e 28 para erros léxicos) da linguagem Pascal. Tais erros foram recolhidos e classificados através de um estudo informal de todos os 520 alunos matriculados no curso de Introdução à Ciência da Computação - CS110. Já o trabalho de Hristova et al. (2003) elaborou uma lista com 62 erros de programação da linguagem Java, dos quais 20 foram identificados como essenciais. O trabalho de Ahmadzadeh, Elliman e Higgins (2005) utilizou a linguagem Java para analisar 108.652 registros de erros dos estudantes, tendo como resultado 36 erros de sintaxe, 63 semânticos e 1 léxico, constatando que o erro mais comum foi o de não definir uma variável.

Diante deste cenário, e preocupados com a qualidade de ensino, vislumbramos que há uma grande necessidade que os professores saibam quais são os erros mais comuns cometidos pelos aprendizes de programação, enfatizando-os em suas aulas introdutórias. O conhecimento acerca dos erros também serve para encontrar maneiras de preveni-los e de facilitar sua identificação. Por conseguinte, este trabalho traz como contribuição uma análise dos erros mais comuns de programação da linguagem Python, cometidos pelos usuários da ferramenta The Huxley[1] - um ambiente virtual de aprendizagem e juiz *on-line*, agrupado-os como erros de execução e sintaxe.

[1] www.thehuxley.com

2. Metodologia

Neste trabalho foi utilizado o juiz *on-line* The Huxley pelo fato de que foi disponibilizada uma API para conexão direta com a base de dados atual, permitindo acesso a diversas funções disponíveis no sistema, possibilitando assim a extração de diversos dados acerca das submissões de programas de aprendizes para realização de uma análise estatística.

2.1. Questões de Pesquisa

Diante da base de dados da ferramenta The Huxley, temos como questão central de pesquisa a seguinte pergunta: *Quais os erros mais comuns cometidos pelos usuários da ferramenta que utilizam a linguagem Python?* Para responder a essa questão, foram definidas as seguintes questões específicas de pesquisa:

> Questão de Pesquisa 1: Qual o percentual de submissões por linguagem?
> Questão de Pesquisa 2: Qual o percentual de submissões corretas e não corretas por linguagem?
> Questão de Pesquisa 3: Qual o resultado das submissões realizadas na linguagem Python?

2.2. Base de dados

A base de dados do The Huxley é relacional, mas permite o acesso à mesma através de uma API disponível através de Web Services que recebem requisições no formato JSON e retornam respostas também neste formato.

De acordo com Iglesias (2009) JSON é uma notação de objetos feita em JavaScript, é um formato criado para requisição de dados de forma leve. Provê uma notação de fácil leitura e escrita, tanto para seres humanos quanto para máquinas, ou seja, possui uma forma fácil para analisar e gerar sua implementação. É completamente independente do idioma. O The Huxley utiliza esta notação na integração com sua plataforma web e para se acessar a base de dados.

Foram obtidas as estatísticas das submissões, assim como a base de dados com os erros das submissões realizadas na linguagem Python através de requisições feitas ao juiz *on-line* utilizando o padrão REST com JSON, como ilustra o arquivo localizado no caminho *Codes/baseErros.py* disponível em um compartilhamento do Google Drive[2].

A Figura 1 ilustra a notação da mensagem de erro mapeada, onde, inicialmente é identificada a linha que a contém, posteriormente sua classe e por fim, o subtipo do erro com sua descrição. Esta notação é específica da linguagem Python.

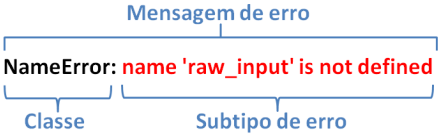

Figura 1. Mensagem de erro mapeada.

[2]gg.gg/galileuFiles

2.3. Extração dos dados

Inicialmente foi criado um *script* de acesso à API da ferramenta, para acessar a base de submissões realizadas desde a sua concepção. Os dados extraídos da base de dados encontram-se no caminho *Codes/baseErros.txt* e os *scripts* em *Codes/Cod_errorMsg.py* e *Codes/RequestData.py*, onde estes dois últimos fazem o acesso às mensagens de erros e às características das submissões, respectivamente.

No momento da execução, em Janeiro de 2017, foram encontradas exatamente 667.836 submissões, sendo distribuídas dentre todas as linguagens disponíveis para submissão e os tipos de retorno da ferramenta.

3. Resultados

Para responde à questão de pesquisa central, foi realizada a análise em toda a base, onde obtivemos 20 classes de erros, subdivididas em 143 mensagens de erros, como ilustra a planilha disponível *on-line*[3]. A Tabela 1 ilustra as classes de erros com sua descrição, a quantidade de subtipos de erros mapeados (Qtd) e seu percentual de ocorrência.

Tabela 1: Classes de erros.

Classe	**Descrição**	**Qtd (%)**
SyntaxError	São os erros referentes aos erros sintáticos, tais como uso incorreto de comandos, falta o uso incorreto de parâmetros ou atributos.	22 (33,39)
NameError	São os erros referentes à nomeação de variáveis, métodos ou comandos que foram escritos de forma equivocada.	1 (15,04)
ValueError	São os erros referentes aos valores, seja de expressões ou tipos de dados.	27 (14,51)
TypeError	São os erros referentes aos tipos dos dados, por atribuição, comparação ou expressão, com dados que não são do mesmo tipo.	64 (11,28)
EOFError	São os erros referentes a final de arquivo - *EOF (End Of File)*, onde acontece geralmente por uma leitura de dados feita de forma errada, onde a entrada não corresponde à leitura de dados realizada pelo código-fonte.	1 (9,35)
IndentationError	São os erros referentes a indentação, visto que Python é uma linguagem que possui delimitação de escopo por indentação.	5 (7,61)
IndexError	São os erros referentes aos índices de estruturas de dados, ao qual foi tentado acessar um determinado índice inexistente.	2 (3,67)
AttributeError	São os erros referentes aos atributos, acontece quando é feito uma tentativa de acesso a um atributo que não existe.	2 (1,88)

[3] gg.gg/requi

Continuação da Tabela 1

Classe	**Descrição**	**Qtd (%)**
ZeroDivisionError	São os erros referentes a operações de divisões quando o divisor possui valor igual a zero.	4 (1,29)
KeyError	São os erros referentes a chaves, geralmente acontecem quando é feito o uso de uma chave que não existe para acessar uma determinada estrutura de dados.	1 (0,73)
UnboundLocal-Error	São os erros referentes ao uso de variáveis locais que não foram declaras, resolvidos, geralmente com sua declaração como global.	1 (0,53)
TabError	São os erros referentes ao uso do *Tab* para indentação, geralmente acontece quando há diferença entre seu uso para indentação.	2 (0,47)
ImportError	São erros referentes à importação de bibliotecas, geralmente acontece quando a biblioteca não existe ou não foi escrita da forma correta.	2 (0,13)
FileNotFound-Error	São os erros referentes a caminhos em arquivos, quando o mesmo não existe ou não foi possível especificá-lo.	1 (0,07)
PermissionError	São os erros referentes a permissões, quando há a tentativa de acessar um caminho em que o usuário não tem permissão para leitura ou escrita.	1 (0,02)
OverflowError	São os erros referentes ao uso de números que possuem um tamanho muito acima do permitido para um determinado tipo de dados.	3 (0,02)
UnicodeDecode-Error	São os erros referentes ao uso de decodificação de caracteres não reconhecidos para o padrão utilizado, geralmente acontece quando se usa caracteres específicos de um determinado idioma ou símbolo.	1 (0,01)
calendar.Illegal-MonthError	São os erros referentes ao uso de calendário, acontece quando o formato do mês, ano ou dia está incorreto.	1 (0,006)
UnicodeEncode-Error	São os erros referentes ao uso de codificação de caracteres não reconhecidos para o padrão utilizado, geralmente acontece quando se usa caracteres específicos de um determinado idioma ou símbolo. A padronização mais comum é *utf8* ou *latin1*.	1 (0,002)
IOError	São os erros referentes à permissão para acessar um determinado diretório, geralmente acontece quando tenta gravar em arquivo em um local não permitido.	1 (0,002)

Ao analisar e catalogar os 143 subtipos de erros, foi feita a pesquisa do significado dos erros na internet e a verificação dos códigos-fontes com os respectivos erros, disponível em compartilhamento do Google Drive[2]. Para cada subtipo de erro, que provém de um tipo de mensagem de erro, foi associada uma descrição que visa explicar o subtipo do erro.

A Tabela 2 ilustra os 21 (vinte e um) subtipos de erros mais frequentes que, após o processo de extração e análise, foram encontrados na base, representando 87,63% do total de ocorrência dos erros. Para cada erro foi atribuído um significado.

Tabela 2: 21 erros mais frequentes.

Classe: subtipo	**Significado**	**Qtd (%)**
SyntaxError: invalid syntax	Ausência de parênteses, ausência de : e declaração incorreta de alguns comandos, tais como: atribuição, if, for, while, input, print e outros.	12371 (23,72)
NameError: name 'raw_input' is not defined	Uso de uma variável que não foi definida ou um comando que foi escrito de forma errada.	7839 (15,03)
ValueError: invalid literal for int() with base 10: '3 5 2'	Uso de literais incompatíveis com o comando *'int'*.	5895 (11,30)
EOFError: EOF when reading a line	EOF - end of file (final de arquivo) - erro de final de arquivo ao realizar a leitura dos dados.	4872 (9,34)
SyntaxError: Missing parentheses in call to 'print'	Uso de parênteses, a partir da versão 3.0 do Python seu uso é obrigatório no comando print.	2430 (4,66)
IndexError: list index out of range	O índice da lista está fora do intervalo.	1905 (3,65)
IndentationError: expected an indented block	Erro de indentação de bloco.	1562 (3,00)
IndentationError: unexpected indent	Indentação inesperada ou desnecessária.	1413 (2,71)
AttributeError: str' object has no attribute 'itmes'	O objeto do tipo "str" não tem nenhum atributo com o nome "itmes".	1281 (2,46)
TypeError: unsupported operand type(s) for -: 'list' and 'list'	A operação - não é suportada para os tipos usados.	1263 (2,42)
IndentationError: unindent does not match any outer indentation level	A indentação não corresponde a nenhum nível de indentação externa.	938 (1,80)
ValueError: could not convert string to float: '100 170 100 95'	Não é possível converter o tipo string para float.	640 (1,23)
ZeroDivisionError: float division by zero	Não é possível realizar divisão por zero.	531 (1,02)
TypeError: int' object is not iterable	O objeto do 'int' não é iterável, é apenas uma literal. Verifique se não é necessário o uso de uma lista ou o comando range().	465 (0,89)

Continuação da Tabela 2

Classe: subtipo	**Significado**	Qtd (%)
TypeError: unorderable types: str() <= int()	Os tipos não são comparáveis, por isso não é possível realizar a comparação. Modifique-os para que sejam do mesmo tipo.	424 (0,81)
KeyError: '12' 12 1000 13 1 'a'	A chave digitada não é válida.	378 (0,72)
SyntaxError: unexpected EOF while parsing	Foram utilizados caracteres que não seguem o padrão da linguagem, assim ocorreu o erro de final de arquivo inesperado.	339 (0,65)
TypeError: not all arguments converted during string formatting	Nem todos os argumentos foram convertidos durante a formatação. Erro de formatação, dos tipos das variáveis ou argumentos.	312 (0,60)
TypeError: int() argument must be a string or a number, not 'list'	A função int() deve ter o argumento do tipo string ou um número, não outro tipo.	206 (0,39)
TypeError: Can't convert 'int' object to str implicitly	Não é possível converter o tipo 'int' implicitamente para o tipo str, ou seja, por mais que seja compreensível sua conversão.	202 (0,39)
TypeError: int() argument must be a string, a bytes-like object or a number, not 'list'.	O argumento da função 'int()' deve ser uma sequência de caracteres ou bytes - como um objeto ou um número, não 'string'.	148 (0,28)

Para responder à questão de pesquisa 1: Qual o percentual de submissões por linguagem? Foi feita a busca das submissões que continham a extensão de cada linguagem disponível. A Figura 2, ilustra o percentual de submissões por linguagem, onde a linguagem C possui o maior percentual de submissões, com 53,44%, seguida pela linguagem Python, com 27,48%, sucedidas pelas linguagens: C++ com 7,88%, Java com 6,42%, Octave com 4,20% e por fim, Pascal com 0,58%.

Para responder à questão de pesquisa 2: Qual o percentual de submissões corretas e erradas por linguagem? Foi feita a análise de todas as submissões erradas, correspondendo a cerca de 71,97% do total. A Figura 3 ilustra as submissões corretas e erradas de acordo com cada linguagem possível para submissão no The Huxley. A linguagem de programação Python se destaca, já que concentra 82,13% das submissões erradas contra 17,87% corretas, isso significa que de todas as submissões, o maior percentual não aceito por linguagem é desta. Em segundo lugar está a linguagem Octave, com 72% consideradas erradas contra 28% corretas. A terceira posição é ocupada pela linguagem Java, com 68,48% erradas e 31,52% corretas. A quarta posição é da linguagem C, com 68,45% erradas e 31,55% corretas. Seguida pela linguagem Pascal com 65,83% erradas e 34,17% corretas e a linguagem C++ com 63,48% erradas e 36,52% corretas.

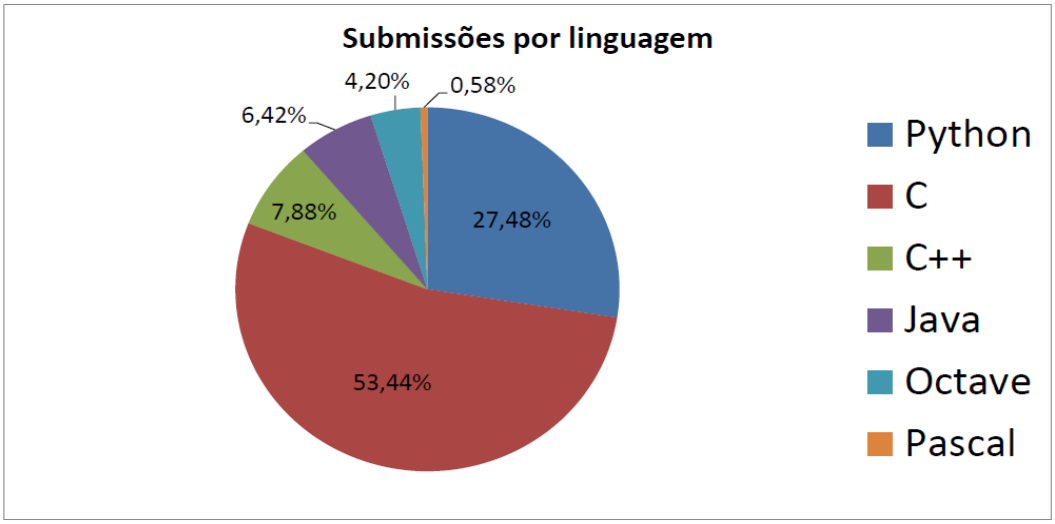

Figura 2. Submissões por linguagem.

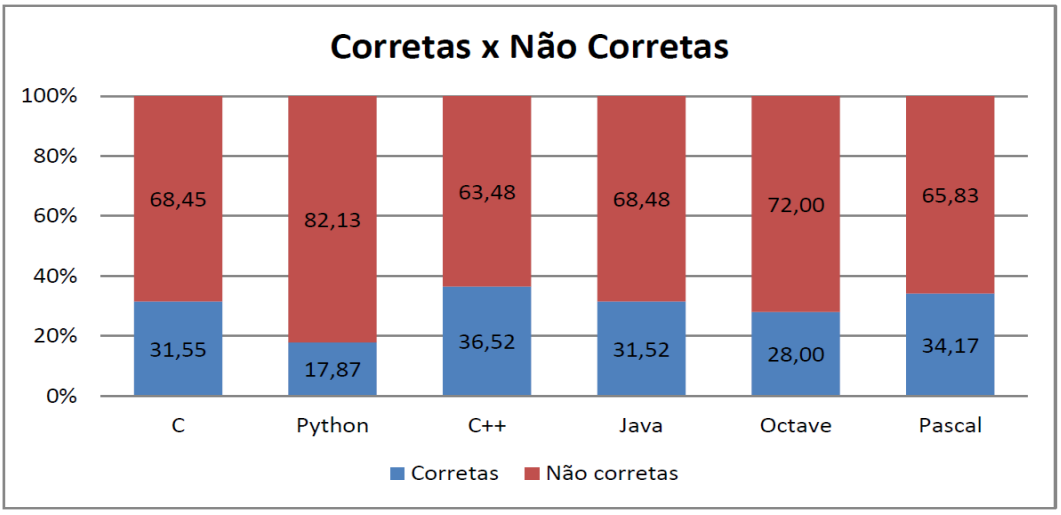

Figura 3. Submissões consideradas correta e não corretas de acordo com cada linguagem.

Para responder à questão de pesquisa 3: Qual o resultado das submissões realizadas na linguagem Python? Foi realizada a busca pelas submissões referentes somente à linguagem Python. Ao analisar os resultados das submissões desta linguagem, obtivemos que a maior porcentagem de submissões são de *RUNTIME_ERROR*, com cerca de 54,33%, caracterizando as mensagens de erros retornadas, seguidas pelos percentuais: *WRONG_ANSWER* com 23,90%; *CORRECT* com 17,98%; *EMPTY_ANSWER* com 2,03%; *PRESENTATION_ERROR* com 0,94%; *TIME_LIMIT_EXCEEDED* com 0,79% e por fim, *FORK_BOMB* com 0,0032%, como ilustra a Figura 4. O erro COMPILATION_ERROR não entrou na análise pois o The Huxley utiliza o interpretador padrão de Python, por isto os erros sintáticos são classificados como RUNTIME_ERROR. A explicação de cada categoria de erro usada nos juízes *on-line* está disponível em[4] e no trabalho de Revilla, Manzoor e Liu (2008).

[4]maratona.ime.usp.br/manualBOCA.html

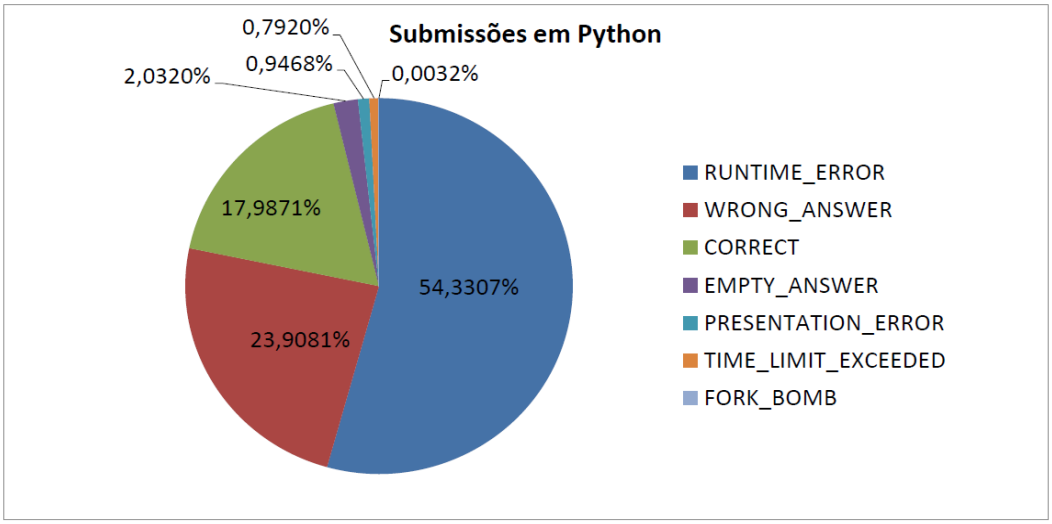

Figura 4. Resultado das submissões realizadas em Python.

A linguagem Python possui maior percentual de submissões não corretas, conforme ilustrado na Figura 3, mesmo não sendo a linguagem com maior quantidade de submissões, como ilustra a Figura 2. Dentre as não corretas, 54,33% de suas submissões são do tipo (*RUNTIME_ERROR*), ou seja, com erro em tempo de execução.

4. Considerações Finais e Trabalhos Futuros

Foram mapeados todos os erros encontrados na base de dados do juiz *on-line* The Huxley, sendo aprofundado o estudo nos erros da Linguagem Python por esta ter apresentado uma incidência de erros bem maior que as outras. Como resultado, chegou-se a 143 mensagens de erros, agrupadas em 20 classes. Através destes dados, obtivemos estatísticas que ilustram os erros mais comuns para aprendizes de programação que utilizam a linguagem Python.

Diante das respostas às questões de pesquisa supracitadas, norteadas pelas estatísticas obtidas, há fortes indícios de que os erros mais comuns são: (a) de sintaxe, quando o aluno faz o uso incorreto de alguns comandos, como *if, for, while, input e print* e esquece o *token* ':' do comando condicional *if*; (b) declaração de variáveis, quando é feito o uso de uma variável que não foi definida anteriormente e (c) erros de tipos, quando são realizadas operações com tipos diferentes de dados, em sua maioria, usando um literal como inteiro. A Tabela 2 ilustra os 21 erros mais comuns encontrados, representando 87,63% do total da base de dados. É de suma importância que os professores enfatizem os erros mais comuns, explicando melhor o uso do ':' em Python, assim como os tipos de dados, conversão entre eles e sua compatibilidade.

Salientamos que os erros catalogados podem não refletir com a mesma proporção entre aprendizes não usuários do The Huxley, mas sua coleta é de difícil realização porque o interpretador não arquiva e nem compartilha os erros para análise posterior.

Como trabalho futuro, buscando a continuidade da pesquisa, faremos o mesmo levantamento nas outras linguagens de programação aceitas pelo The Huxley, como C, C++, Java, Octave e Pascal.

Referências

AHMADZADEH, M.; ELLIMAN, D.; HIGGINS, C. An analysis of patterns of debugging among novice computer science students. *ACM SIGCSE Bulletin*, ACM, v. 37, n. 3, p. 84–88, 2005.

CASPERSEN, M. E.; KOLLING, M. Stream: A first programming process. *ACM Transactions on Computing Education (TOCE)*, ACM, v. 9, n. 1, p. 4, 2009.

HRISTOVA, M. et al. Identifying and correcting java programming errors for introductory computer science students. In: ACM. *ACM SIGCSE Bulletin*. [S.l.], 2003. v. 35, n. 1, p. 153–156.

IGLESIAS, G. *Handling cross-domain web service calls*. [S.l.]: Google Patents, 2009. US Patent App. 12/110,499.

JÚNIOR, J. C. R. P.; RAPKIEWICZ, C. E. O processo de ensino-aprendizagem de fundamentos de programação: Uma visão crítica da pesquisa no brasil. *WEI, I*, v. 7, p. 28, 2004.

LAHTINEN, E.; ALA-MUTKA, K.; JÄRVINEN, H.-M. A study of the difficulties of novice programmers. In: ACM. *Acm Sigcse Bulletin*. [S.l.], 2005. v. 37, p. 14–18.

LEMOS, M. A. de; BARROS, L. N. de; LOPES, R. de D. Uma biblioteca cognitiva para o aprendizado de programação. *WEI*, 2003.

MIYADERA, Y.; HUANG, N.; YOKOYAMA, S. A programming language education system based on program animation. In: *Proceedings of Educational Uses of Information and Communication Technologie, in IFIP 16th World Computer Congress*. [S.l.: s.n.], 2000. p. 258–261.

REVILLA, M. A.; MANZOOR, S.; LIU, R. Competitive learning in informatics: The uva online judge experience. *Olympiads in Informatics*, v. 2, p. 131–148, 2008.

SCHORSCH, T. Cap: an automated self-assessment tool to check pascal programs for syntax, logic and style errors. In: ACM. *ACM SIGCSE Bulletin*. [S.l.], 1995. v. 27, n. 1, p. 168–172.

WEBER, G.; BRUSILOVSKY, M.; STEINLE, F. Elm-pe: An intelligent learning environment for programming, 1996. *Obtain in: www. psychologie. uni-trier. de*, v. 8000, 2014.