Do Current Online Coding Tutorial Systems Address Novice Programmer Difficulties?

Ohud Alasmari
School of Computing Science,
University of Glasgow
Glasgow, United Kingdom
o.alasmari.1@research.gla.ac.uk

Jeremy Singer
School of Computing Science,
University of Glasgow
Glasgow, United Kingdom
jeremy.singer@glasgow.ac.uk

Mireilla Bikanga Ada
School of Computing Science,
University of Glasgow
Glasgow, United Kingdom
mireilla.bikangaada@glasgow.ac.uk

ABSTRACT

Mastering programming skills is a multifaceted challenge, particularly for novice learners. While abundant existing literature examines students' learning obstacles and proposes potential resolutions—predominantly through questionnaire-based methodologies—this study presents a fresh perspective. This paper focuses on identifying programming learning impediments as presented in the literature, paired with an exploration of the supportive features intrinsic to online coding platforms that could potentially mitigate these difficulties. The findings reveal that several of these online coding systems lack crucial features that could effectively address the learning difficulties experienced by novice programmers. This lack stresses an urgent call to create more robust and learner-centric environments that better facilitate the acquisition of programming skills for beginners, thus bridging the identified gap in the existing pedagogical tools.

CCS CONCEPTS

• **Applied computing → Interactive learning environments**.

KEYWORDS

computer programming, difficulties, learning, online coding tool.

ACM Reference Format:
Ohud Alasmari, Jeremy Singer, and Mireilla Bikanga Ada. 2023. Do Current Online Coding Tutorial Systems Address Novice Programmer Difficulties? . In *The 15th International Conference on Education Technology and Computers (ICETC 2023), September 26–28, 2023, Barcelona, Spain.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3629296.3629333

1 INTRODUCTION

Learning to program is a complex task. Learners from diverse educational backgrounds face considerable difficulties learning how to code [5]. Recently, various web-based online systems have been developed to introduce novices to programming concepts, helping to overcome learning difficulties through a range of pedagogical methods [16]. Generally, learners benefit from programming and problem-solving in easy, accessible, and appropriate online systems that support effective learning approaches [24]. Kim and Ko [16] classify online programming learning environments into five categories: interactive platform; web reference platform; educational games system; creative platform; and, massive open online course. Many people use these platforms to learn, whether independently or as part of an organized study group. However, few studies focus on understanding the effectiveness of such learning environments. This research focuses entirely on *Online Coding Tutorial Systems*. These systems adopt many of the features that have been identified in Kim and Ko's first category of interactive platforms, along with some aspects of their creative platforms and MOOCs [16]. In detail, online coding tutorial systems are web-hosted, browser-mediated systems that enable learners to explore coding content through a series of structured coding tutorials [9] with an embedded, interactive interpreter shell known as a Read-Eval-Print loop (REPL), where learners can practice coding by typing fragments of source code into the browser window, running the code, seeing the output, and inspecting error messages as appropriate. There is an intuitive appeal to online coding relative to more traditional software development environments like Visual Studio. In an online setting, there are no requirements for local installation and no platform compatibility issues. All execution takes place either server-side or in the local browser JavaScript runtime, with appropriate sandboxing. Online interpreters are easily embedded in programming language websites, MOOCs, and other learning resources. This builds on the recent trend of programming *playgrounds* as popularized by Apple's Swift language.

The current work aims to identify system features that assist learners in overcoming programming learning difficulties and to investigate whether current online coding tutorial systems provide these features. To summarize the findings of this research: (1) There are a set of challenges that have been faced by novices who are learning to code; (2) In programming education research, several possible solutions have been suggested to overcome the identified problems; (3) The current online coding tutorial systems provide some recognized novice support features, but many such features (e.g., underlining syntax errors, visual maps, code auto-completion) are missing from all the selected systems.

2 RELATED WORK

Several literature reviews related to investigating programming learning challenges, identifying supportive features, and analyzing online coding learning platforms have been previously published.

2.1 Systematic surveys of programming challenges to learners and solutions

Kader et al. [15] published an article related to difficulties in teaching and learning programming. The aim of this paper is to identify the factors that lead to difficulties and challenges in learning computer programming for novice students and suggest strategies for addressing these issues. The main contribution of this paper was assisting computer science educators to improve their teaching approaches for basic programming courses, enhance students' interest, and increase students' performance in programming subjects. In addition, Quian et al. [28] published a review of students' misconceptions and other difficulties in introductory programming. The focus of this paper was to review relevant literature on general definitions of misconceptions and studies about students' misconceptions and other difficulties in introductory programming. This review aimed to explore the factors that contribute to the difficulties, and strategies and tools to address them, including misconceptions, were discussed.

Ahmad et al. [1] also published a review of literature focusing on programming teaching and learning by addressing issues and challenges in the context of introductory programming at the tertiary level. The main objective of this article was to propose a categorization of programming challenges and highlight the key issues in programming teaching and learning in higher education for further research and improvement. Moreover, Paul and Simon [25] published an article that discusses some programming learning challenges. In addition, their article focuses on student understanding and use of a mix of programming environments (in particular, Python IDLE for offline programming and CodeRunner for programming quizzes) and code fragment problems. The purpose of this study was to examine the challenges that first-year distance learning students face when learning a procedural programming language such as Python over the course of a 21-week computing and IT course, which includes six weeks on problem-solving and programming with Python.

In summary, the reviewed four articles only discuss programming, learning, and teaching challenges; no investigations into possible solutions for these identified problems have been done. Therefore, this current systematic review distinguishes itself from the earlier systematic reviews by suggesting a list of possible solutions to assist novice programmers in overcoming programming learning challenges.

2.2 Surveys on online programming learning systems

In the computing education literature, few studies were conducted to analyze or investigate online interactive coding systems. For instance, Zinovieva et al. [42] published an article that discusses a comparative analysis of different online programming learning platforms for teaching programming according to specific criteria. However, they only analyzed some platforms; online coding tutorial platforms or interactive coding systems were not analyzed. Moreover, Sim et al.[35] reviewed research on supporting novice programming, focusing on the implementation of programming environments that might be solutions for programming learning problems. Their study specifically focused on tools that support block programming and intelligent tutoring systems. However, when Sim et al.[35] conducted a general analysis of intelligent tutoring systems, they did not consider most of the systems' features. Also, Kim and Ko [16] published an article related to online programming learning systems. They analyzed and categorized online programming learning systems into several categories. However, they did not analyze the systems based on supporting learners to overcome programming learning problems.

To conclude, in the reviewed articles, the authors did not consider in their analysis any supportive features that might support learners to overcome the programming learning challenges. Therefore, in the current work, the online coding tutorial systems comparative analysis distinguishes itself from the earlier systems analysis by examining current online coding tutorial systems to determine whether they offer supportive features to assist novices.

3 RESEARCH QUESTIONS AND METHODOLOGIES

3.1 Research Questions

Despite abundant literature on programming, teaching, and learning difficulties, we observed that the majority of relevant research predominantly relied on quantitative, questionnaire-based methodology. Although such work has uncovered novice learners' difficulties, an in-depth understanding of the supportive system features that can overcome these challenges is limited. Thus, a key contribution of this work is to identify a set of supportive features for online coding platforms. The current deployed online coding tutorial systems were analyzed, asking whether they provide any of these supportive features for learners. The main research question that has been addressed is, ***Do Current Online Coding Tutorial Systems Address Novice Programmers Difficulties?***, by elucidating sub-questions as follows:

RQ1: What are common programming learning difficulties for novices?

RQ2: Which supportive features are potential solutions for these identified difficulties?

RQ3: Do these identified supportive features exist in currently deployed online coding tutorial systems?

3.2 Research Design and Methods

To answer the three RQs, two studies were conducted. Firstly, a systematic review was conducted on programming education literature to identify the common programming teaching and learning difficulties for novices and propose solutions to address such issues (Section 4). Secondly, a case study was conducted to investigate whether online coding tutorial systems provide the identified solutions (Section 5).

4 NOVICE PROBLEMS AND SOLUTIONS

This section presents the first study that addresses the first two research questions in this paper; **RQ1** (identification of novice learning difficulties, Section 4.1) and **RQ2** (identification of potential solutions, Section 4.2). In this study, two distinct literature reviews were conducted to answer these two research questions.

4.1 Novice Learning Difficulties

Educators often encounter problems when teaching students to programme [36] since coding is dynamic and abstract. Several researchers have identified difficulties in learning programming [21, 39]. Programming novices might fail to recognize their own deficiencies due to a lack of high-level understanding of the problems [18, 22]. Therefore, in this work, a systematic literature review was conducted to explore some of these programming learning challenges.

4.1.1 ***Study method.*** Below are the systematic review strategies used in this study: **Search databases:** The databases searched are mainly from the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE). The reason for focusing only on these two databases is that most research related to computing education can be found in the ACM and IEEE databases, where most of the well-known computing education conference proceedings and journals can be found. For instance, Koli Calling, ICER, UKICER, and SIGCSE. **Search terms:** A narrow search was done in order to select and review the papers that only focused on identifying programming learning problems. In addition, the strategy that has been used is to search the papers only with the titles of programming, teaching, or learning problems. Therefore, the keywords used were boolean combinators, as follows: **"Programming" OR "Coding" OR "Computer Programming" AND "Learning" AND "Difficulties" OR "Issues" OR "Challenges" OR "Problems"**. **Search process:** The search process was done by selecting the "Title" option in both databases in the advanced search. Therefore, the number of returned papers seems to be small. **Publication date:** The initial search was for articles published between 1980 and 2023. This selected period of time has been chosen in order to include most of the published studies in the programming education field. **Inclusion and exclusion criteria:** 52 relevant articles were found. Then, an initial inclusion screening was done based on title and abstract to get a subset of candidate studies that only focus on programming learning challenges, and the article numbers were filtered to 25 after removing duplicates and out-of-focus papers. From these 25 remaining articles, further screening was performed by considering full-text content, excluding articles that did not discuss programming learning difficulties, and removing duplicate and non-English articles. The final number of selected articles was 7.

4.1.2 ***Study findings.*** The seven selected publications discuss programming learning challenges using different methods. For instance, Milne and Rowe [22] conducted a web-based questionnaire on the various concepts and topics of object-oriented programming that students in introductory courses found most difficult to cope with. Also, Hashim et al.[13] distributed questionnaire to 226 undergraduate students to identify the programming problems they are facing. Moreover, Piteira and Costa [24] conducted a case study to obtain teachers' and students' opinions to identify programming challenges. In addition, Tan et al. [39] conducted a case study to investigate the factors that lead to undergraduates' learning difficulties in programming courses and also their perceptions of which teaching methodologies could be implemented to create a richer and more interesting learning process. On the other hand, Gomes and Mendes [12] conducted several interviews with teachers and students to identify programming challenges. Bosse and Gerosa [5] conducted a systematic literature review and empirical study to identify programming learning problems. similar to Qian et al. [29], who investigated students' difficulties in learning to program in Python by conducting an empirical study. Six programming learning difficulties were found in these selected publications, as shown in Table 1. The rows are ordered from the highest number of papers mentioning a problem to the lowest number, with the requirement that at least two papers must corroborate a difficulty before including it in the provided list. Three problems were observed (syntax, structure, and basic concept understanding) that are inherent in reading and understanding code. The remaining three problems (debugging, proceduralization, and algorithm implementation) involve writing and executing code. As shown in Table 1, understanding the syntax of any programming language was considered a main programming learning challenge for novice programmers. [5] [29] [24] [12] [13] [39]. Similar to understanding syntax, understanding the code structure is also considered a common problem for programming learners [22] [5] [29] [24] [13] [39]. On the other hand, the less common programming learning challenge is transferring algorithms to concrete implementation [12] [39].

Table 1: List of programming learning difficulties identified in the literature *answering RQ1*

Programming Learning Difficulties	[22]	[5]	[29]	[24]	[12]	[13]	[39]
Syntax of programming languages		✓	✓	✓	✓	✓	✓
Structure of code	✓	✓	✓	✓		✓	✓
Understanding basic concepts	✓		✓	✓		✓	✓
Debugging		✓		✓		✓	✓
Dividing functionality into procedures				✓		✓	✓
Transferring algorithm to concrete implementation					✓		✓

4.2 Potential Solutions

4.2.1 ***Study method.*** To address the second research question, *RQ2: Which supportive features can potentially mitigate the identified learning difficulties?*, a snowballing technique was employed to uncover features of supportive systems that could potentially serve as solutions for the common programming learning difficulties identified in Section 4.1 [40]. The snowballing procedure began with a review of the potential solutions outlined in the selected articles from the previous section. This initial review facilitated the identification of a number of supportive features designed to assist novice programmers based on the insights from these articles. Subsequently, these identified difficulties were employed as keywords to guide a more focused literature search in the ACM and IEEE databases.

4.2.2 *Study findings.*

This section catalogs the supportive features presented in the literature as possible solutions for the identified difficulties. Features were grouped according to the problem they addressed, and only features that could be incorporated into a software-based programming environment were considered.

Syntax of programming languages According to [17], syntax difficulties are the overhead of learning the syntax and semantics of a language at the same time. In addition, [33] mentioned that understanding syntax problems is simply a challenge that is faced by not only novices but also by learners who have adequate problem-solving skills and manage to phrase a solution to a programming problem in terms of understanding the syntax of a code but find it difficult to turn the code into a syntactically correct computer program. Moreover, according to [33], syntax understanding is not the main difficulty. Learners may know the syntax and semantics of individual statements, but they do not know how to combine those elements in order to produce valid programs. However, three helpful supportive features have been discussed in previous studies. These features are: syntax error messages, underlining syntax errors, and syntax source code highlighting or coloring.

- **Syntax error messages:** Reporting syntax errors in a programming environment helps learners reduce mistakes in spelling, punctuation, and the order of keywords in their programs[14]. For instance, *SyntaxTrain* parses a student's source code and, if it detects a syntax error, displays an error message and a diagram illustrating the required syntax [23]. Moreover, [14] mentioned that syntax error messages do not necessarily point the learners in the right direction needed to fix the code, but providing this technique in programming environments helps learners understand syntax and semantic errors in their codes. Providing comprehensible syntax error messages is often motivated by the need to better serve novice programmers [34].
- **Underlining syntax errors:** Source code errors in modern integrated development environments are highlighted interactively with red underlines below problematic lines of code [4]. This is often accompanied by hints or error-messaging pop ups.
- **Syntax highlighting:** Highlighting helps learners identify keywords and become familiar with language-specific concrete syntax. Researchers have examined the influence of syntax highlighting on novice comprehension of source code [28].

Structure of code Learners struggle to understand how to build blocks of code, syntax constructs, and commands that perform actions[21]. However, a supportive feature has been identified; this feature is a visual map.

- **Visual map:** Natural visual learners are recommended to begin by learning a graphical programming language, which provides a visual map as a bridge to learning textual programming languages [37]. In general, program visualization systems are developed to help beginners understand fundamental programming concepts, structure, and execution. [38].

Understanding basic concepts According to [13] [24] most novice programmers have problems understanding basic concepts. Learners must understand the basic concepts at the beginning of their learning journey. For instance, variables, arrays, and loops However, four supportive features have been identified as helpful for understanding the basic concept of programming languages: lesson content, reference materials, worked solutions, and quizzes.

- **Lesson content:** Programming learning environments can provide contextually relevant, structured lesson content that teaches different concepts [8].
- **Reference materials:** These authoritative resources can promote an understanding of programming concepts since learners can browse and request them at any time. Such reference information might be organized as a digital textbook [8].
- **Worked solutions:** The availability of complete example programs and worked solutions was used in some programming learning environments to demonstrate programming concepts [8].
- **Quizzes:** Basic assessment activities and quizzes allow learners to test their understanding. In addition, providing questions along with interactive exercises can help reinforce concepts and measure learners' achievement [8].

Debugging Novice programmers need to know how to test and analyze their code to identify and correct problems, and this is called debugging [10]. In addition, learners face some difficulties in understanding the problem domain, finding bugs and errors, and resolving bugs [21] [39] [10]. To help learners overcome this difficulty, three supportive features have been identified: detailed error messages, identifying error locations, and customized hints.

- **Detailed error messages:** Raw error messages are often uninformative and sometimes misleading for novices [28]. Researchers have created tools to provide enhanced error messages. For instance, CS1 students who saw detailed error messages made significantly fewer errors than those who only saw raw Java error messages; more detailed error messages helped students debug their programs [3].
- **Identifying error locations:** Novice programmers get frustrated by errors, and they try to debug their code in the hope that they discover the location of the error to make debugging easier [6, 34]. Therefore, novices persistently seek assistance for problems with basic syntactic details and identifying error locations that can help learners debug their code [6].
- **Customized hints:** Providing specific hints based on learner errors is a beneficial addition to any teaching programming platform [2][20]. When a novice types a segment of code and the editor shows an error, the interactive hints feature should provide some guidance to help the developer find and fix the bug [2, 28].

Dividing functionality into procedures

- **Auto-completion:** The code editor should predict what the programmer wants to type. In addition, auto-completion is considered a helpful feature for supporting users in making procedure calls and developing additional procedures that can be used exactly the same way as the built-in library functions [41]. Moreover, this technique saves programmers time by helping them complete keywords rather than typing every character.

Transferring an algorithm to a concrete implementation

- **Syntax-directed editor:** To help novice programmers use a programming language to implement an algorithm for solving a specific problem without concern for syntactic detail, some programming learning tools support templates and menus with syntactically correct choices for every incomplete part of a program. Such templates can be based on textual or graphical representations.

5 CURRENTLY DEPLOYED SYSTEMS

A survey of popular online coding tutorial platforms was conducted to address RQ3:*Do these identified supportive features exist in currently deployed online coding tutorial systems?*

5.1 Study Method

Several current systems were selected and analyzed for the presence or absence of the identified supportive features. The systematic basis for selecting the systems involves looking at the top ten languages by popularity on GitHub (Python, JavaScript, Java, TypeScript, Go, C++, Ruby, PHP, C#, C). Attention to high-level interpretive and scripting languages was restricted, since compiler-based toolchains are not particularly suitable for deployment in online coding environments. From the remaining languages on the list, Python, JavaScript, Java, TypeScript, Go, Ruby, and PHP were selected since they are appropriate for novice programmers. The search for 'interactive tutorial X' on Google was done, with X in turn being each language. The highest ranked links on Google were followed, which led to online coding tools. Based on this method, for the software survey, LearnPython [32], TryJavaScript [26], LearnJava [30], Codecademy LearnTypeScript [7], Tour of Go [11], RubyMonk [27] and LearnPHP [31] were selected.

6 FINDINGS AND DISCUSSION

Table 2 summarizes the analysis of the seven selected online coding tutorial systems across the identified supportive features. The rows are ordered from the most common supportive features that exist in all the selected current online coding tutorial systems to the least common supportive features. The main observation noticed in this comparative analysis study is that most of the identified supportive features exist in more than three of the selected systems, such as lesson content, detailed error messages, a syntax-directed editor, syntax error messages, syntax highlighting, reference materials, and identified error locations. On the other hand, grey rows in Table 2 indicate that three supportive features are not provided by any of the studied systems, such as underlining syntax errors, visual mapping, and auto-completion. In addition, it is noticed that the three features—worked solutions, customized hints, and quizzes—are only provided by three or fewer online coding tutorial systems under study.

To conclude, the findings indicate that all seven selected online coding tutorial systems are missing important supportive features that could serve as potential solutions to the identified programming learning challenges. As shown in Table 2, the findings provide supporting evidence that current online coding tutorial systems do not fully address novice programming difficulties and are therefore not ideal for novice learners since they lack significant supportive features that have proven to be helpful for novice programmers to overcome specific programming learning difficulties. These findings reinforce an important point that might be the reason behind this lack of supportive features in online coding tutorial systems: that they are created by programming language developers (whether original language designers or enthusiastic advocates) with minimal input from computing education experts [19]. One implication of these findings is the urgent need for enhanced support for novice programmers through a more considerate design process for online coding tutorial systems that takes learners' needs into consideration.

7 LIMITATIONS AND FUTURE WORK

The main limitation might this study has is the number of returned papers that seems to be small. The listed search terms in this study should return hundreds of papers. However, the returned number of papers was 52 because the advanced search options were selected to search only through the titles of the papers to narrow the huge number of returned papers in both databases. One fruitful future avenue of research could be a more wide-ranging review of papers and a list of more common and specific programming learning difficulties. Moreover, certain limitations of the software survey in this paper could be addressed in future research. For example, only seven current online coding tutorial systems that cover seven programming languages were examined. A fruitful future avenue of research might include building an online coding tutorial system prototype that incorporates all the identified supportive features in this paper, along with a large-scale user study. Further longitudinal work will be valuable in seeing how programming educators' perspectives alter as web technology evolves.

8 CONCLUSION

This study investigated whether or not current popular online coding tutorial systems address novice programmers' difficulties. A systematic literature review revealed a range of challenges specifically related to the learning of computer programming. A list of supportive features was identified from programming education literature. This set of supportive features was used as a baseline to analyze the selected current online coding tutorial systems. The findings indicate that while many of the identified problems might be partially mitigated by current online systems, there is evidence that some of the identified supportive features are not provided by current systems. It might be speculated that the reason for these deficiencies, which do not appear to be technological in nature, is that the developers of these systems are not informed by pedagogical principles or perspectives. Therefore, the next steps are to raise awareness regarding these supportive features in developing online coding tutorial systems.

REFERENCES

[1] S Ahmad and Juzlinda Ghazali. 2020. Programming Teaching and Learning: Issues and Challenges. *Fstm. Kuis. Edu. My* 16, 1 (2020), 724–398.

Table 2: Comparative analysis of the inclusion of supportive features across seven tutorial systems (the grey row indicates the complete absence of features in all systems

Supportive Features	LearnPython	TryJavaScript	LearnJava	Learn-TypeScript	Tour of Go	RubyMonk	LearnPHP
Lesson content	✓	✓	✓	✓	✓	✓	✓
Detailed error messages	✓	✓	✓	✓	✓	✓	✓
Syntax-directed editor	✓	✓	✓	✓	✓	✓	✓
Syntax error messages	✓		✓	✓	✓	✓	✓
Syntax highlighting	✓		✓	✓		✓	✓
Reference materials	✓	✓	✓	✓			✓
Identifying error locations	✓		✓		✓		✓
Worked solutions				✓		✓	✓
Customized hints						✓	
Quizzes	✓						
Underlining syntax errors							
Visual map							
Auto-completion							

[2] Paolo Antonucci, Christian Estler, Durica Nikolić, Marco Piccioni, and Bertrand Meyer. 2015. An incremental hint system for automated programming assignments. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*. 320–325. https://doi.org/10.1145/2729094.2742607

[3] Brett A Becker. 2016. An effective approach to enhancing compiler error messages. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. 126–131. https://doi.org/10.1145/2839509.2844584

[4] Brett A Becker, Paul Denny, Raymond Pettit, Durell Bouchard, Dennis J Bouvier, Brian Harrington, Amir Kamil, Amey Karkare, Chris McDonald, Peter-Michael Osera, et al. 2019. Compiler error messages considered unhelpful: The landscape of text-based programming error message research. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*. 177–210. https://doi.org/10.1145/3344429.3372508

[5] Yorah Bosse and Marco Aurélio Gerosa. 2017. Why is programming so difficult to learn? Patterns of Difficulties Related to Programming Learning Mid-Stage. *ACM SIGSOFT Software Engineering Notes* 41, 6 (2017), 1–6. https://doi.org/10.1145/3011286.3011301

[6] Joshua Charles Campbell, Abram Hindle, and José Nelson Amaral. 2014. Syntax errors just aren't natural: Improving error reporting with language models. In *Proceedings of the 11th Working Conference on Mining Software Repositories*. 252–261. https://doi.org/10.1145/2597073.2597102

[7] Codecademy. 2022. LearnTypeScript. https://www.codecademy.com/learn/learn-typescript.

[8] Tyne Crow, Andrew Luxton-Reilly, and Burkhard Wuensche. 2018. Intelligent tutoring systems for programming education: a systematic review. In *Proceedings of the 20th Australasian Computing Education Conference*. 53–62. https://doi.org/10.1145/3160489.3160492

[9] Tao Dong and Gale Yang. 2020. Towards a pattern language for interactive coding tutorials. In *Conference Companion of the 4th International Conference on Art, Science, and Engineering of Programming*. 102–105. https://doi.org/10.1145/3397537.3397558

[10] Sue Fitzgerald, Renée McCauley, Brian Hanks, Laurie Murphy, Beth Simon, and Carol Zander. 2009. Debugging from the student perspective. *IEEE Transactions on Education* 53, 3 (2009), 390–396. https://doi.org/10.1109/TE.2009.2025266

[11] Andrew Gerrand et al. 2022. A Tour of Go. https://go.dev/tour.

[12] Anabela Gomes and Antonio Mendes. 2014. A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations. In *Proceedings of the IEEE Frontiers in Education Conference*. 1–8. https://doi.org/10.1109/FIE.2014.7044086

[13] Ahmad Sobri Hashim, Rohiza Ahmad, and Muhammad Shafiq Shahrul Amar. 2017. Difficulties in Learning Structured Programming: A Case Study in UTP. In *7th World Engineering Education Forum*. 210–215. https://doi.org/10.1109/WEEF.2017.8467151

[14] Maria Hristova, Ananya Misra, Megan Rutter, and Rebecca Mercuri. 2003. Identifying and correcting Java programming errors for introductory computer science students. *ACM SIGCSE Bulletin* 35, 1 (2003), 153–156. https://doi.org/10.1145/792548.611956

[15] Rozita Kadar, Naemah Abdul Wahab, Jamal Othman, Maisurah Shamsuddin, and Siti Balqis Mahlan. 2021. A study of difficulties in teaching and learning programming: a systematic literature review. *International Journal of Academic Research in Progressive Education and Development* 10, 3 (2021), 591–605.

[16] Ada S Kim and Andrew J Ko. 2017. A pedagogical analysis of online coding tutorials. In *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education*. 321–326. https://doi.org/10.1145/3017680.3017728

[17] Theodora Koulouri, Stanislao Lauria, and Robert D Macredie. 2014. Teaching introductory programming: A quantitative evaluation of different approaches. *ACM Transactions on Computing Education* 14, 4 (2014), 1–28. https://doi.org/10.1145/2662412

[18] Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. 2005. A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin* 37, 3 (2005), 14–18. https://doi.org/10.1145/1151954.1067453

[19] Lauri Malmi, Ian Utting, and Amy J. Ko. 2019. Tools and Environments. In *The Cambridge Handbook of Computing Education Research*, Sally A. Fincher and Anthony V.Editors Robins (Eds.). Cambridge University Press, 639–662. https://doi.org/10.1017/9781108654555.022

[20] Samiha Marwan, Joseph Jay Williams, and Thomas Price. 2019. An evaluation of the impact of automated programming hints on performance and learning. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. 61–70. https://doi.org/10.1145/3291279.3339420

[21] Rodrigo Pessoa Medeiros, Geber Lisboa Ramalho, and Taciana Pontual Falcão. 2018. A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education* 62, 2 (2018), 77–90. https://doi.org/10.1109/TE.2018.2864133

[22] Iain Milne and Glenn Rowe. 2002. Difficulties in learning and teaching programming—views of students and tutors. *Education and Information technologies* 7, 1 (2002), 55–66. https://doi.org/10.1023/A:1015362608943

[23] Andreas Leon Aagaard Moth, Joergen Villadsen, and Mordechai Ben-Ari. 2011. SyntaxTrain: relieving the pain of learning syntax. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*. 387–387. https://doi.org/10.1145/1999747.1999900

[24] Martinha Piteira and Carlos Costa. 2013. Learning computer programming: study of difficulties in learning programming. In *Proceedings of the 2013 International Conference on Information Systems and Design of Communication*. 75–80. https://doi.org/10.1145/2503859.2503871

[25] Paul Piwek and Simon Savage. 2020. Challenges with learning to program and problem solve: an analysis of student online discussions. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 494–499. https://doi.org/10.1145/3328778.3366838

[26] PluralSight. 2022. TryJavaScript. https://www.javascript.com/try.

[27] Sidu Ponnappa and Jasim A Basheer. 2022. Ruby Monk. http://rubymonk.com.

[28] Yizhou Qian and James Lehman. 2017. Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education* 18, 1 (2017), 1–24. https://doi.org/10.1145/3077618

[29] Yizhou Qian, Peilin Yan, and Mingke Zhou. 2019. Using data to understand difficulties of learning to program: A study with Chinese middle school students. In *Proceedings of the ACM Conference on Global Computing Education*. 185–191. https://doi.org/10.1145/3300115.3309521

[30] Ron Reiter. 2022. LearnJava. https://www.learnjavaonline.org/.

[31] Ron Reiter. 2022. LearnPHP. https://www.learn-php.org/.

[32] Ron Reiter. 2022. LearnPython. https://www.learnpython.org.

[33] Robert S Rist. 1996. Teaching Eiffel as a first language. *Journal of object-oriented programming* 9, 1 (1996), 30–41.

[34] Eddie Antonio Santos, Joshua Charles Campbell, Dhvani Patel, Abram Hindle, and José Nelson Amaral. 2018. Syntax and sensibility: Using language models to detect and correct syntax errors. In *Proceedings of the IEEE 25th International Conference on Software Analysis, Evolution and Reengineering*. 311–322. https://doi.org/10.1109/SANER.2018.8330219

[35] Tze Ying Sim and Sian Lun Lau. 2018. Online tools to support novice programming: A systematic review. In *2018 IEEE Conference on e-Learning, e-Management and e-Services (IC3e)*. IEEE, 91–96.

[36] Derek Sleeman. 1986. The challenges of teaching computer programming. *Commun. ACM* 29, 9 (1986), 840–841. https://doi.org/10.1145/6592.214913

[37] Bryan J Smith. 2009. Conceptual graphs as a visual programming language for teaching programming. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*. 258–259. https://doi.org/10.1109/VLHCC.2009.5295242

[38] Juha Sorva, Ville Karavirta, and Lauri Malmi. 2013. A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education* 13, 4 (2013), 1–64. https://doi.org/10.1145/2490822

[39] Phit-Huan Tan, Choo-Yee Ting, and Siew-Woei Ling. 2009. Learning difficulties in programming courses: undergraduates' perspective and perception. In *Proceedings of the International Conference on Computer Technology and Development*. 42–46. https://doi.org/10.1109/ICCTD.2009.188

[40] Claes Wohlin. 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*. 1–10.

[41] Stelios Xinogalos. 2013. Using flowchart-based programming environments for simplifying programming and software engineering processes. In *Proceedings of the IEEE Global Engineering Education Conference*. 1313–1322. https://doi.org/10.1109/EduCon.2013.6530276

[42] IS Zinovieva, VO Artemchuk, Anna V Iatsyshyn, OO Popov, VO Kovach, Andrii V Iatsyshyn, YO Romanenko, and OV Radchenko. 2021. The use of online coding platforms as additional distance tools in programming education. In *Journal of physics: Conference series*, Vol. 1840. IOP Publishing, 012029. Issue 1.