Journal of Education and Learning Technology (JELT)
ISSN 2720-7730
Volume 6 Issue 12 2025 pp 1555 - 1573
Available online at: https://noyam.org/journals/jelt/
DOI: https://doi.org/10.38159/jelt.202561224

Challenges Experienced by First-Year Students in the Programming Module - A Literature Review

Moretlo Tlale-Mkhize [1]

[1] Central University of Technology, Free State, South Africa.

ABSTRACT

Transitioning from being a learner to a student can be an intimidating experience for others. First-year students enrolled in programming module at higher education institutions often struggle to grasp concepts taught while navigating the difficulties of adapting to the university environment. This theoretical paper discusses the challenges faced by first-year students in programming courses. Such challenges hinder the commencement of subsequent classes, which depend on the foundational concepts taught. Constructivist learning theory is adopted in the study as it enables students to build their understanding of the world through activities. Constructivism, shaped by Jean Piaget, maintains that students construct their knowledge rather than receiving it passively from the environment. This study adopted a desktop literature review methodology to explore the existing knowledge on challenges experienced by first-year students in the programming module. Literature was conducted from several databases, including Google Scholar, Scopus, ProQuest, and ScienceDirect. This paper addresses two key questions: What challenges do first-year students encounter in the programming module, and how can educators effectively support them? The study highlights that constructivism can help first-year students understand programming concepts by constructing meaning of what they have learned. Grasping the theoretical aspects and a lack of sufficient problem-solving skills are some challenges identified. The paper contends that the challenges experienced by first-year students in the programming module can be mitigated through the application of constructivism by assisting students to develop computational thinking skills. Some strategies to address the identified challenges include improving teaching methods and incorporating gamification into programming classes.

Correspondence
Moretlo Tlale-Mkhize
Email:
ctlale@cut.ac.za

Publication History
Received:
31st May, 2025
Accepted:
12th November, 2025.
Published:
30th December, 2025.

To Cite this Article:
Tlale-Mkhize, Moretlo. "Challenges Experienced by First-Year Students in the Programming Module - A Literature Review." *Journal of Education and Learning Technology* 6, no. 12 (2025): 1555 - 1573. https://doi.org/10.38159/jelt.202561224

Keywords: Programming Module, Higher Education Institutions, Constructivism

INTRODUCTION

The shift from high school to university can be a frightening experience for several first-year students, particularly in the computer programming field. Maneuvering the difficulties associated with coding, understanding computational thinking, and adapting to the challenging requirements of a programming module can present several challenges for these students. [1] Programming, which encompasses the

[1] Qi Cao et al., "Learners' Differences in Blended Learner-Centric Approach for a Common Programming Subject," *International Journal of Information and Education Technology* 13, no. 6 (2023): 906–13, https://doi.org/10.18178/ijiet.2023.13.6.1886; Leticia Laura-Ochoa, Norka

development of software, constitutes a pivotal activity in which millions of individuals partake globally, and the instruction of programming has been firmly entrenched within the frameworks of international secondary and higher education.[2] In higher education, the predominant objective in the pedagogy of computer programming is to empower students with the essential competencies to synthesise theoretical knowledge with practical application.[3] The acquisition of programming skills is regarded as a pivotal competency that has the potential to augment and foster students' capabilities in the contemporary educational landscape of the 21st century.[4] A computer program is composed of words and characters that signify a sequence of operations intended for execution by a computational system.[5] This dual emphasis on conceptual knowledge and practical execution highlights the need for teaching methods that successfully connect knowledge with application.

Literature indicates that first-year students enrolled in the programming module at higher learning institutions struggle to grasp the concepts taught in the programming module. This implies that students who fail to acquire critical skills at key points in their learning will likely encounter significant challenges later in their learning journey.[6] It is highlighted that the programming is seen as difficult by most people.[7] Even for students who are enrolled on this module, it can cause anxiety and frustration for them.[8] Introductory programming courses at universities and other higher education institutions aim to empower students with the foundational skills, knowledge, and expertise required to build effective software solutions.[9] Therefore, programming is emerging as the required foundation skill for software developers.

Various studies have been conducted on the teaching of first-year programming modules. Figueiredo and García-Peñalvo noted that student success in introductory programming courses is a widespread concern.[10] The challenges associated with learning programming frequently obstruct the academic progression of novice computing students, particularly those who lack prior knowledge and exposure to programming concepts.[11] Figueiredo and García-Peñalvo developed a prototype to identify potential problems in monitoring students' success and to suggest immediate responses.[12]

Following significant shifts in teaching methodologies and the provision of educational resources, Groher et al. investigated which diversity factors are crucial when instructing non-computer science students

---

Bedregal-Alpaca, and Elizabeth Vidal, "Improving Computational Thinking in Nursing Students through Learning Computer Programming," *International Journal of Advanced Computer Science and Applications* 13, no. 5 (2022), https://doi.org/10.14569/IJACSA.2022.0130571.

2 Greg Michaelson, "Teaching Programming with Computational and Informational Thinking," *Journal of Pedagogic Development* 5 (March 1, 2015): 51–65.

3 Wilson Osafo Apeanti and Daniel Essel, "Learning Computer Programming Using Project-Based Collaborative Learning," *International Journal for Innovation Education and Research* 9, no. 8 (August 1, 2021): 191–207, https://doi.org/10.31686/ijier.vol9.iss8.3278.

4 Anastasios Theodoropoulos and George Lepouras, "Augmented Reality and Programming Education: A Systematic Review," *International Journal of Child-Computer Interaction* 30 (December 2021): 100335, https://doi.org/10.1016/j.ijcci.2021.100335.

5 Andreas Larsson and Karin Stolpe, "Hands on Programming: Teachers' Use of Metaphors in Gesture and Speech Make Abstract Concepts Tangible," *International Journal of Technology and Design Education* 33, no. 3 (2023): 901–19.

6 Claudio Alvarez, Maira Marques Samary, and Alyssa Friend Wise, "Modularization for Mastery Learning in CS1: A 4-Year Action Research Study," *Journal of Computing in Higher Education* 36, no. 2 (August 16, 2024): 546–89, https://doi.org/10.1007/s12528-023-09366-1.

7 Brett A Becker et al., "Programming Is Hard-or at Least It Used to Be: Educational Opportunities and Challenges of Ai Code Generation," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, 2023, 500–506.

8 Sithandiwe Pornelia Twetwa-Dube, "Exploring Group Work Strategies to Teach Computer Programming: A Case Study of First-Year and Extended Programme Students at One South African University," *International Journal of Learning, Teaching and Educational Research* 23, no. 7 (July 30, 2024): 271–88, https://doi.org/10.26803/ijlter.23.7.14.

9 Sarita Singh, "Identifying Learning Challenges Faced by Novice/Beginner Computer Programming Students: An Action Research Approach," in *QuASoQ/SEED@APSEC*, 2022, https://api.semanticscholar.org/CorpusID:255968169.

10 José Figueiredo and Francisco García-Peñalvo, "Teaching and Learning Strategies for Introductory Programming in University Courses," in *Ninth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'21)* (New York, NY, USA: ACM, 2021), 746–51, https://doi.org/10.1145/3486011.3486540.

11 Billy Javier, "Understanding Their Voices from Within: Difficulties and Code Comprehension of Life-Long Novice Programmers" 1 (August 1, 2021): 53–76.

12 Figueiredo and García-Peñalvo, "Teaching and Learning Strategies for Introductory Programming in University Courses."

in programming, as well as how educators and teaching assistants perceive diversity. Their analysis confirmed that a collaborative teaching concept supports female students and those with language barriers.[13] This research aims to identify the primary challenges faced by first-year students in programming courses, with the goal of introducing new perspectives by exploring strategies and interventions that can alleviate these difficulties and foster a more supportive learning environment. Grounded in constructivist theory, the research emphasizes active, student-centered learning environments where knowledge is constructed through exploration, collaboration, and reflection. By integrating inclusive teaching strategies, reflective practice, and problem-based learning cycles, this study seeks to enhance programming instruction and foster deeper conceptual understanding, ultimately supporting student success at the foundational level. Faced with first-year programming students, such as those with limited prior exposure to coding, language barriers, and high dropout rates.

THEORETICAL FRAMEWORK

The constructivist theory of learning implies that learners build their understanding of the world through activities, reflecting on what was done previously. Constructivism highlights that learning should go beyond merely transmitting information from the lecturer to the student, but encourages students to actively participate in their learning process.[14] Therefore, constructivism is more of a theoretical endeavour relating to students constructing their knowledge. Through experience and interaction with the environment, students can learn and understand what they are doing. The emphasis is that individuals construct knowledge and that learning is an active process. As a learning theory, constructivism argues that learners develop their understanding by physically interacting with their surroundings. This model presupposes that learners participate in the process of learning, paying attention to the processes of knowledge acquisition so that learning can be maximally effective.[15] According to Smith, constructivism is a suitable theory for computer science education.[16] Therefore, constructivist theory is identified as a theoretical approach to assist students with computational thinking. Computational thinking skills encompass understanding basic computing concepts, including problem-solving, algorithms, programming, and data handling.[17]

The role of the instructor in constructivist education is that of a moderator who supports learners and walks with them through the different stages of the course, helping them to search for and make sense of available information.[18] Through participation, students become more active in the knowledge construction process, acquiring the power in the learning process.[19] Piaget's theory significantly influenced educational theory and practice, particularly approaches where the importance of constructing knowledge is highlighted.[20] Therefore, in relation to education, constructivism calls for the design of educational processes that promote interaction, inquiry, and active problem-solving among learners. The constructivist instructional

[13] Iris Groher et al., "Exploring Diversity in Introductory Programming Classes," in *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Software Engineering Education and Training* (New York, NY, USA: ACM, 2022), 102–12, https://doi.org/10.1145/3510456.3514155.

[14] Lilis Marina Angraini, Nia Kania, and Ferit Gürbüz, "Students' Proficiency in Computational Thinking Through Constructivist Learning Theory," *International Journal of Mathematics and Mathematics Education*, February 21, 2024, 45–59, https://doi.org/10.56855/ijmme.v2i1.963.

[15] M. Givi Efgivia et al., "Analysis of Behaviorism Learning Theory, STEM Learning Model and Gamification," 2021, https://doi.org/10.2991/assehr.k.211020.029.

[16] Julie Smith, "Constructivism in Computer Science Education," in *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2*, 2022, 1171.

[17] Angraini, Kania, and Gürbüz, "Students' Proficiency in Computational Thinking Through Constructivist Learning Theory."

[18] Hai-Ninh Do, Bich Ngoc Do, and Minh Hue Nguyen, "3How Do Constructivism Learning Environments Generate Better Motivation and Learning Strategies? The Design Science Approach," *Heliyon* 9, no. 12 (December 2023): e22862, https://doi.org/10.1016/j.heliyon.2023.e22862.

[19] S. Ahmadvand and M. Khoshchehreh, "Theory And Practice: Implications Of Social Constructivism In Education," *International Journal of Humanities and Social Development Research* 7, no. 2 (January 4, 2023), https://doi.org/10.30546/2523-4331.2023.7.2.19.

[20] Grzegorz P. Karwasz and Katarzyna Wyborska, "How Constructivist Environment Changes Perception of Learning: Physics Is Fun," *Education Sciences* 13, no. 2 (February 12, 2023): 195, https://doi.org/10.3390/educsci13020195.

approach emphasises meaningful learning and knowledge construction through internal cognitive processes. New knowledge emerges from existing knowledge through the transformation, organisation, and reorganisation of prior understanding.[21] Constructivist learning theory stresses that individuals generate knowledge and meaning based on their experiences.[22] As a result, learners must take responsibility for their educational journey.

Constructivism provides a theoretical framework for first-year programming students by fostering self-directed learning, critical thinking, and active knowledge construction.[23] As a theoretical framework for first-year programming students, emphasis is on student-centred, multisensory learning experiences. It encourages active engagement with programming concepts, allowing students to construct meaning by linking new information to their prior knowledge, enhancing understanding and retention.[24] Constructivism serves as a theoretical framework for first-year programming students by emphasising the synthesis of prior knowledge and experience, fostering creativity through collaborative project work, social interaction, scaffolding, and active learning, ultimately enhancing their programming competency.[25] Constructivism, as a theoretical framework for first-year programming students, emphasises active engagement in learning, where students construct their knowledge. Instructors facilitate this process by creating effective learning environments and utilising technology as a tool for problem-solving and skill enhancement.[26] Students construct their knowledge by developing meaning from their learning experiences.[27]

METHODOLOGY

This study is guided by the framework set by the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA). The PRISMA Statement and its extensions constitute a minimum set of recommendations for the transparent and complete reporting of systematic reviews.[28] Figure 1 shows the PRISMA flow diagram that was adopted for this study. Using the PRISMA framework made it possible to comprehensively identify, screen, assess for eligibility, and incorporate scholarly literature detailing the obstacles faced by first-year students in the programming module. Searches conducted in Google Scholar, Scopus, ProQuest, and ScienceDirect were exported to EndNote for citation management.

[21] Winda Ismi Hidayanti, Diana Rochintaniawati, and Rika Rafikah Agustin, "The Effect of Brainstorming on Students' Creative Thinking Skill in Learning Nutrition," *Journal of Science Learning* 1, no. 2 (March 31, 2018): 44, https://doi.org/10.17509/jsl.v1i2.8738.

[22] Efgivia et al., "Analysis of Behaviorism Learning Theory, STEM Learning Model and Gamification."

[23] Liudmyla Byrkun and Larysa Liashenko, "Theory of Constructivism and a Communicative Coursebook as a Prerequisite for Organizing the First Year It Students' Independent Work at the Esp Classes," *Grail of Science*, no. 38 (April 29, 2024): 298–302, https://doi.org/10.36074/grail-of-science.12.04.2024.050.

[24] David Carless, *Educational Assessment and Data Collection Methods* (London: Routledge, 2022).

[25] Natalie Kiesler, "Reviewing Constructivist Theories to Help Foster Creativity in Programming Education," in *2022 IEEE Frontiers in Education Conference (FIE)* (IEEE, 2022), 1–5.

[26] Lazarus Ndiku Makewa, "Constructivism Theory in Technology-Based Learning," in *Technology-Supported Teaching and Research Methods for Educators* (IGI Global, 2019), 268–87, https://doi.org/10.4018/978-1-5225-5915-3.ch015.

[27] Angraini, Kania, and Gürbüz, "Students' Proficiency in Computational Thinking Through Constructivist Learning Theory."

[28] Sarkis-Onofre, R., Catalá-López, F., Aromataris, E., & Lockwood, C. (2021). How to properly use the PRISMA Statement. *Systematic reviews*, *10*(1), 117.

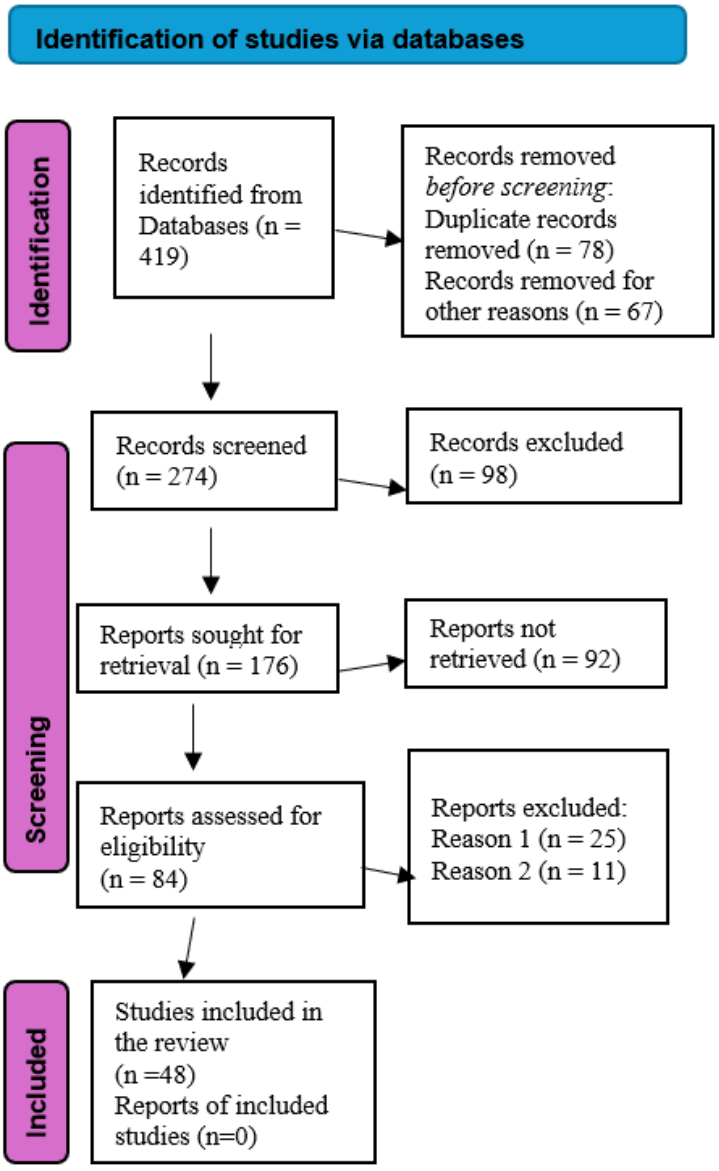

Figure 1: PRISMA flow diagram adapted from Rethlefsen and Page[29]

Identification

A structured literature search was completed using Google Scholar, Scopus, ProQuest, and ScienceDirect. To capture the literature outlining the challenges faced by first-year students in the programming module. The literature search focused on articles that were published between 2015 and 2025. The initial search yielded 419 records by using the following keyword combinations with Boolean operators:

• ("challenges in programming" OR "programming module") AND
• ("introductory programming" OR "first-year programming") AND
• ("teaching programming" OR "learning programming"). 419 records were retrieved in total.

Screening

Duplicates were eliminated through the automated 'Find Duplicates' feature, and remaining records were subjected to manual screening in EndNote based on the study's predetermined inclusion and exclusion

[29] Rethlefsen, M. L., & Page, M. J. (2022). PRISMA 2020 and PRISMA-S: common questions on tracking records and the flow diagram. *Journal of the Medical Library Association: JMLA*, *110*(2), 253

criteria. The process left 274 articles for title and abstract screening. Studies that were irrelevant, not peer-reviewed, and unrelated to teaching and learning programming were excluded.
• Articles excluded at this phase: 98
• Remaining for full-text review: 176

Eligibility
The full texts of 176 articles were reviewed based on the criteria set for inclusion and exclusion as outlined below.:

RESULTS AND DISCUSSION
Literature on programming challenges has grown. This section synthesises key studies to identify:

- The Thematic areas – (i) the role of programming in the first-year curriculum, (ii)
- common challenges faced by first-year programming students, (iii) strategies to overcome these challenges, and (iv) the present and future of programming.
- Inclusion Criteria: This paper focused on studies that focused on challenges experienced by first-year students in the programming module; Articles employing programming techniques, Peer-reviewed journal and conference papers, and English-language publications.
- Exclusion Criteria: The exclusion criteria focused on non-programming for first-year students and articles without available full-text. After applying these criteria, 84 articles were deemed eligible.

In addition, 36 articles had to be excluded for not meeting the inclusion criteria (reason 1 = No recommendations or contribution), leaving a total of 48 articles adopted and used in this paper.

Data extraction and analysis
The author(s) and date, the research objectives and methodology, the first-year programming module, the evaluation metrics (accuracy, false positive rate, etc.), the dataset and experimental setup, and the key findings and limitations were key information for each included study and were recorded using a structured data extraction form. This systematic approach facilitated the identification of key themes, literature gaps, and trends that informed the findings and theoretical interpretation sections synthesis.

The Role of Programming in the First-Year Curriculum
Introduction to programming courses hold particular significance within computing disciplines, as they serve as a prerequisite for advancing to higher-level courses.[30] This is the case at a university of technology in South Africa, where the Programming Principles I module serves as a prerequisite to advance to the Software Development I module in the second year of study. The curriculum includes designing flowcharts, writing pseudocode, and coding using the C# programming language. Pseudocode is a tool that can be used to understand a problem and consider algorithms.[31] The use of pseudocode to formulate a solution represents a level of abstraction that is neither too low nor too high. On the other hand, Sobral noted that the curriculum for introduction to programming promotes the initial contact of science, technology, engineering, and mathematics (STEM) for undergraduate students with computational thinking and programming languages.[32]

[30] Uzma Omer, Muhammad Shoaib Farooq, and Adnan Abid, "Introductory Programming Course: Review and Future Implications," *PeerJ Computer Science* 7 (July 22, 2021): e647, https://doi.org/10.7717/peerj-cs.647.
[31] Orly Barzilai et al., "Cognitive Aspects in Problem Solving: The Case of a Data Structures Course for IS Students," *Journal of Information Systems Education* 35, no. 2 (2024): 175–88, https://doi.org/10.62273/JJUB4136.
[32] Sónia Rolland Sobral, "Teaching and Learning to Program: Umbrella Review of Introductory Programming in Higher Education," *Mathematics* 9, no. 15 (July 23, 2021): 1737, https://doi.org/10.3390/math9151737.

Programming integration differs per nationality; some integrate it with math and even crafts, showcasing its significance in boosting cross-disciplinary skills.[33]

Choosing the appropriate programming language for foundational courses is a critical choice, as the majority of educators focus on more accessible languages for novice users and the job market.[34] There are many options available for selecting the first programming language. Some institutions of learning suggest starting off with block-based visual programming languages like Scratch or Blockly that use drag-and-drop block interfaces, which abstract syntax to make programming's underlying concepts more accessible, while others opt for more versatile entry-level languages like Python or Java, which prepare learners for more advanced tasks in later stages.[35] Prokop et. al., in their study, developed the criteria for selecting the first programming language as presented in Figure 2.[36] Their results demonstrate that ease of learning basic concepts and demand in the labour market are the major drivers of selecting the programming language.

[33] Tiina Korhonen et al., "Finnish Teachers as Adopters of Educational Innovation: Perceptions of Programming as a New Part of the Curriculum," *Computer Science Education* 33, no. 1 (January 2, 2023): 94–116, https://doi.org/10.1080/08993408.2022.2095595.

[34] Sónia Rolland Sobral, "The First Programming Language and Freshman Year in Computer Science: Characterization and Tips for Better Decision Making," in *Trends and Innovations in Information Systems and Technologies*, 2020, 162–74, https://doi.org/10.1007/978-3-030-45697-9_16.

[35] David Weintrop and Uri Wilensky, "Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms," *ACM Transactions on Computing Education* 18, no. 1 (March 31, 2018): 1–25, https://doi.org/10.1145/3089799.

[36] Yuliia Prokop et al., "An Analysis of Criteria for Choosing a First Programming Language in Universities.," in *ICTERI*, 2019, 420–25.

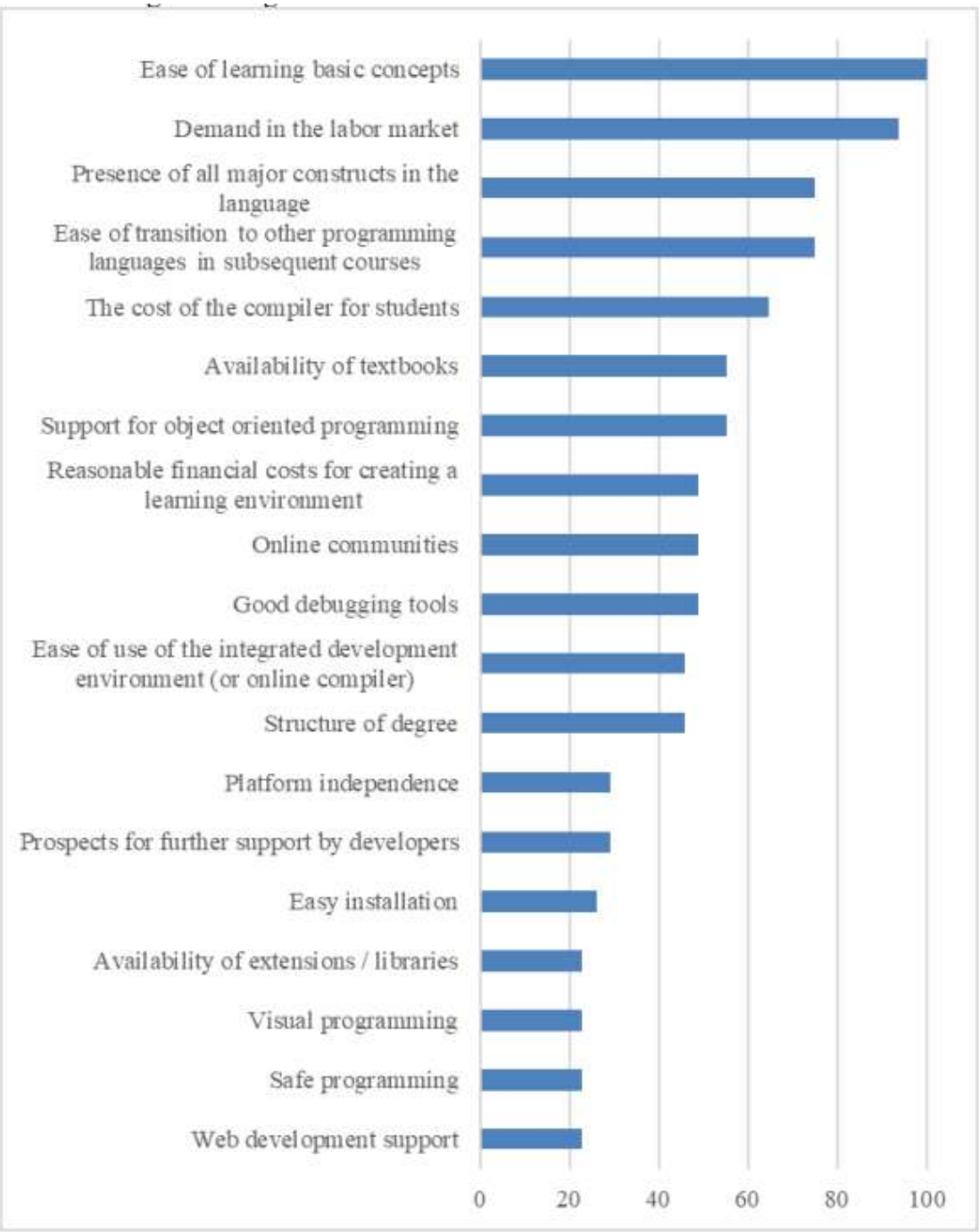

Figure 1: Rating of the criteria for choosing the first programming language

The higher-level programming languages often act as an interface for algorithmic problem solving, allowing students to focus on the organizational logic of their programs rather than the intricate detailing of advanced syntax, hence improving the construction of logic-based programming.[37] In any case, the choice of languages must address the goals of the program's curriculum, taking into consideration the availability of resources, ease of acquisition, and relevance to the intended professional pathways. The selection of

[37] Elpida Keravnou-Papailiou, "Figuring and Drawing: A Visual Approach to Principled Programming," *ArXiv Preprint ArXiv:2202.09229*, 2022.

programming languages and pedagogical approaches has an important effect on students' understanding of programming, thereby influencing their readiness for advanced academic professions.[38]

A well-structured programming course does not introduce only critical concepts but also prepares students for practical applications in their individual fields. Programming serves as a foundation for developing problem-solving skills and critical thinking, which are essential for future assignments and professional work.[39] The students are first introduced to basic programming concepts and advance to more complex concepts as they progress.[40] Such understanding forms a basis for lower-order and mid-order thinking skills in computer science and engineering, which in turn aid their tackling of progressively more advanced subjects in the future. Students undergo training in algorithm formulation and problem decomposition, which are vital to practical programming.[41]

Achieving a practical and theoretical equilibrium is a common objective of several first-year programming classes. Specifically, project-based learning affords students opportunities to implement their knowledge in addressing real-world challenges, enhancing their problem-solving abilities, critical reasoning, and collaborative abilities.[42] Moreover, using interactive coding websites and other digital materials can offer helpful feedback and assistance, which improves a student's learning experience. The students can grasp basic concepts of design patterns and reasoning skills very early on, so they do not get lost in the web of programming languages.[43] Eneji et. al. regard programming as “the” fundamental technical skill for any information technologist, and they assert that for information technologists, the need for proficient programming skills will only grow in the coming years.[44]

Common Challenges Faced by First-Year Programming Students

Existing scholarly literature has highlighted a variety of difficulties that first-year students frequently experience within the programming module. This section discusses the identified challenges.

Problem-solving skills

Within an academic setting, learners may struggle with grasping the theoretical aspects of programming, problem-solving, and abstract concepts in coding processes.[45] A lack of sufficient problem-solving skills, along with a superficial grasp of the concepts, is the primary reason why students face difficulties in acquiring computer programming skills.[46] Medeiros and Ramalho also identified problem-solving as a major learning challenge for students in their first year of programming.[47]

---

[38] Francisca Onaolapo Oladipo and Memunat A. Ibrahim, “The CodeEazee Tool Support for Computational Thinking in Python,” *European Journal of Engineering Research and Science* 3, no. 3 (March 17, 2018): 12, https://doi.org/10.24018/ejers.2018.3.3.637.

[39] Ioanna Moraiti, Anestis Fotoglou, and Athanasios Drigas, “Coding with Block Programming Languages in Educational Robotics and Mobiles, Improve Problem Solving, Creativity & Critical Thinking Skills,” *International Journal of Interactive Mobile Technologies (IJIM)* 16, no. 20 (October 31, 2022): 59–78, https://doi.org/10.3991/ijim.v16i20.34247.

[40] Ramazan Yilmaz and Fatma Gizem Karaoglan Yilmaz, “Augmented Intelligence in Programming Learning: Examining Student Views on the Use of ChatGPT for Programming Learning,” *Computers in Human Behavior: Artificial Humans* 1, no. 2 (August 2023): 100005, https://doi.org/10.1016/j.chbah.2023.100005.

[41] Paul Piwek et al., “Learning to Program,” in *Proceedings of the 3rd Conference on Computing Education Practice* (New York, NY, USA: ACM, 2019), 1–4, https://doi.org/10.1145/3294016.3294024.

[42] Rania Azad M San Ahmed et al., “The Impact of Teaching Materials on Learning Computer Programming Languages in Kurdistan Region Universities and Institutes,” *Kurdistan Journal of Applied Research* 3, no. 1 (2018): 27–33.

[43] Lodi Michael et al., “Constructionist Attempts at Supporting the Learning of Computer Programming: A Survey,” *Olympiads in Informatics* 13 (July 13, 2019): 99–121, https://doi.org/10.15388/ioi.2019.07.

[44] S. E. Eneji et al., “The Significance Role of Programming in Information Technology (IT),” *International Journal of Information Communication Science and Technology*, 2019.

[45] Lieve Laporte and Bieke Zaman, “Informing Content-Driven Design of Computer Programming Games,” in *Proceedings of the 9th Nordic Conference on Human-Computer Interaction* (New York, NY, USA: ACM, 2016), 1–10, https://doi.org/10.1145/2971485.2971499.

[46] J. Culpeper and D. Kádár, *Speech Acts and Politeness in Historical Contexts* (London: Bloomsbury, 2021).

[47] Rodrigo Pessoa Medeiros, Geber Lisboa Ramalho, and Taciana Pontual Falcao, “A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education,” *IEEE Transactions on Education* 62, no. 2 (May 2019): 77–90, https://doi.org/10.1109/TE.2018.2864133.

Logic-related learning challenges

In addition, the rapid pace of instruction, coupled with the necessity to learn a new programming language, along with its grammar, poses huge challenges for some learners, leading to extreme frustration accompanied by a deep sense of alienation. Novice computer programmers often have little knowledge of the logical order of programming languages and their structural framework.[48] According to Ranjeeth and Padayachee, who conducted the research in South Africa, computer programming is challenging to learn as it demands an intensive cognitive effort to develop a high degree of skill and expertise.[49]

Debugging and detecting syntax

First-year students encounter most of the routine challenges, such as debugging and detecting syntax errors. Identifying mistakes in the code process can prove very tedious for newcomers to programming.[50] Medeiros et al. identified that difficulties in learning the syntax of programming languages are a challenge for the students.[51] The students have difficulties with interpreting error messages and determining what measure would be considered most useful in amending their program.[52]

Cultural differences

Cultural differences and language barriers add another layer of challenges when teaching basic programming concepts to students.[53] Some students enter university with limited computer literacy and face challenges in English proficiency, which can hinder their academic progress.[54] Teaching programming to these first-year and extended-programme students presents significant challenges,[55] leading to high dropout rates[56] and students changing their IT specialisations.[57] The course is perceived as challenging for them to grasp, leading to anxiety and frustration.[58]

Group work and collaboration

Group work and collaboration are both exciting and demanding for first-year students. Group work is exciting because it provides space for peer learning and collaborative problem-solving, but at the same time, it may lead to issues such as unequal effort and communication issues.[59] Students get frustrated when they sense that others are not sharing efforts equally, leading to a situation known as "social loafing."[60]

---

[48] Culpeper and Kádár, *Speech Acts and Politeness in Historical Contexts*.

[49] Lerushka Ranjeeth and Indira Padayachee, "Factors That Influence Computer Programming Proficiency in Higher Education: A Case Study of Information Technology Students," *South African Computer Journal* 36, no. 1 (2024): 40–75.

[50] Rakhi Batra and Zahra Atiq, "Understanding Students' Frustration and Confusion during a Programming Task: A Multimodal Approach," in *2023 IEEE Frontiers in Education Conference (FIE)* (IEEE, 2023), 1–10, https://doi.org/10.1109/FIE58773.2023.10343499.

[51] Medeiros, Ramalho, and Falcao, "A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education."

[52] Cris Norman Olipas, "A Phenomenological Study on the Feelings, Challenges and Difficulties Experienced by Information Technology Students in Learning Computer Programming," *Path of Science*, July 31, 2022, 2001–6, https://doi.org/10.22178/pos.83-3.

[53] Groher et al., "Exploring Diversity in Introductory Programming Classes."

[54] Twetwa-Dube, "Exploring Group Work Strategies to Teach Computer Programming: A Case Study of First-Year and Extended Programme Students at One South African University."

[55] Jose Figueiredo and Francisco Jose Garcia-Penalvo, "Increasing Student Motivation in Computer Programming with Gamification," in *2020 IEEE Global Engineering Education Conference (EDUCON)* (IEEE, 2020), 997–1000, https://doi.org/10.1109/EDUCON45650.2020.9125283; Singh, "Identifying Learning Challenges Faced by Novice/Beginner Computer Programming Students: An Action Research Approach."

[56] Figueiredo and García-Peñalvo, "Teaching and Learning Strategies for Introductory Programming in University Courses."

[57] Twetwa-Dube, "Exploring Group Work Strategies to Teach Computer Programming: A Case Study of First-Year and Extended Programme Students at One South African University."

[58] Twetwa-Dube, "Exploring Group Work Strategies to Teach Computer Programming: A Case Study of First-Year and Extended Programme Students at One South African University."

[59] Apeanti and Essel, "Learning Computer Programming Using Project-Based Collaborative Learning."

[60] Apeanti and Essel, "Learning Computer Programming Using Project-Based Collaborative Learning."

Instructional methods

Instruction methods and the nature of support available can also determine students' outcomes. High numbers of students per class and the absence of individual attention can hinder students' ability to access adequate assistance. Moreover, the shift from high school to university-level education can be strenuous since students can become overwhelmed by the pace and standards.[61] The use of real-world examples that students can relate to can help them understand the programming content. The absence of examples that students can relate to, shared by instructors with students, further exacerbates the challenges faced in the acquisition of programming skills.[62]

Cognitive and Conceptual Challenges

Learning programming comes with various hurdles that stem mostly from the abstract nature of programming itself and the convoluted form of problem-solving involved. Learners typically have a hard time with abstract reasoning, which is pivotal in grasping programming concepts such as algorithms and data structures. The foremost cognitive problem for first-year students revolves around grasping the basic concepts of programming. In particular, some students have difficulty with recursion and concepts involving arrays, error handling, and function/procedure methodologies.[63] Together, lack of experience and abstract concepts contribute to students facing these challenges. Take, for example, recursion; it poses a critical problem for beginners as it expects an understanding of self-referential functions.[64]

As articulated by Qian and Lehman, a programming framework comprises a higher-level unit made of elements from other programming frameworks, which refine different programming constructs and principles.[65] Time and space encapsulation allow units in functional programs to maintain relations that are not explicit in the program structure, making reasoning about their relationships easier. The semicolon, loops, and assignment statements are examples of the provided knowledge concepts. The programmers' loops assignment semicolons also showcase the contextual functionality with a self-contained example.

Programming is heavily dependent on a person's reasoning and problem-solving capabilities. A programmer's problem-solving abilities are put to the test in situations where they need to emerge from a deadlock, which is bound by a previously known or familiar solved route.[66] When first learning to program, students struggle with the step-by-step breakdown of a problem. This is often complicated by the need to convert the algorithm into a different set of instructions. For many first-year students, this is a new way of thinking.[67] The primary difficulty is algorithmic thinking, a cognitive mode that many first-year students encounter for the very first time.[68] Michaelson noted that in its simplest form, a programming language is an instrument for algorithm composition, lacking a comprehensive and organised structure and problem solutions.[69]

---

[61] Angela Zavaleta Bernuy et al., "Student Transitions Through an Entire Computing Program," in *The 26th Western Canadian Conference on Computing Education* (New York, NY, USA: ACM, 2024), 1–7, https://doi.org/10.1145/3660650.3660661.

[62] Rozita Kadar et al., "A Study of Difficulties in Teaching and Learning Programming: A Systematic Literature Review," *International Journal of Academic Research in Progressive Education and Development* 10, no. 3 (August 21, 2021), https://doi.org/10.6007/IJARPED/v10-i3/11100.

[63] Alain Mbiada, Bassey Isong, and Francis Lugayizi, "A Comparative Study of Computer Programming Challenges of Computing and Non-Computing First-Year Students," *The Indonesian Journal of Computer Science* 12, no. 4 (August 30, 2023), https://doi.org/10.33022/ijcs.v12i4.3330.

[64] Javier, "Understanding Their Voices from Within: Difficulties and Code Comprehension of Life-Long Novice Programmers."

[65] Yizhou Qian and James Lehman, "Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review," *ACM Transactions on Computing Education (TOCE)* 18, no. 1 (2017): 1–24.

[66] Barzilai et al., "Cognitive Aspects in Problem Solving: The Case of a Data Structures Course for IS Students."

[67] Juan Pablo Saenz and Luigi De Russis, "On How Novices Approach Programming Exercises before and during Coding," in *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, 2022, 1–6.

[68] Twetwa-Dube, "Exploring Group Work Strategies to Teach Computer Programming: A Case Study of First-Year and Extended Programme Students at One South African University."

[69] Michaelson, "Teaching Programming with Computational and Informational Thinking."

Emotional and Psychological Challenges

Students often encounter emotional and psychological challenges when they are learning programming. These challenges often include frustration and anxiety, motivation and engagement, and confidence and self-efficacy. The learning process of programming can be a source of substantial frustration and anxiety among first-year students. The complexity of programming tasks, in conjunction with the pressure to achieve high performance, can lead to feelings of being overwhelmed and stressed.[70] Frustration frequently manifests when students confront unexpected outputs, are unable to resolve compilation errors, or forget essential syntax rules.[71] Motivation and engagement are critical factors in learning to program. However, first-year students often struggle with maintaining motivation, especially when faced with repeated failures or difficulties. Negative emotions such as anger, anxiety, and hopelessness can negatively impact academic performance, while positive emotions like enjoyment and pride can enhance it.[72]

Confidence and self-efficacy are crucial determinants in students' capacity to achieve success in programming. Many first-year students lack confidence in their ability to write accurate code, which can obstruct their progression and overall academic performance.[73] Fostering confidence is essential to assist students in overcoming the initial obstacles associated with learning to program.

Strategies to overcome these Challenges

Strategies can be incorporated into the teaching and learning of the programming module to address the challenges faced during the first year of study. Strategies include improving teaching methods, using gamification in programming classes, encouraging collaboration and peer learning, addressing emotional and psychological barriers, and providing adequate support and resources.

Improving Teaching Methods

First-year students can be better assisted by the teachers through different techniques in class. Incorporating more fun and engaging activities would help these students, especially when teaching. Engaging students in active problem-based learning enhances their performance.[74] These activities shift the focus from the old passive model of the classroom to a more active one, where learners work on the problems together and apply their skills and knowledge. Students constructing an understanding of complex ideas alongside prior knowledge ensure higher rates of achievement.[75] The use of Artificial Intelligence (AI) can also assist the students with personalised learning. According to Tapalova and Zhiyenbayeva, instructors can use AI to personalise the learning of students so that they can develop professional competencies and become competent in new knowledge.[76]

---

[70] Zahra Atiq and Michael C. Loui, "A Qualitative Study of Emotions Experienced by First-Year Engineering Students during Programming Tasks," *ACM Transactions on Computing Education* 22, no. 3 (September 30, 2022): 1–26, https://doi.org/10.1145/3507696.

[71] Batra and Atiq, "Understanding Students' Frustration and Confusion during a Programming Task: A Multimodal Approach."

[72] Nishaal Bhaw, Jeanne Kriek, and Petra le Roux, "Emotional and User-Experience Factors Influencing Student Performance: A Case Study of First-Year Online Computer Programming Assignments," *UnisaRxiv*, 2024.

[73] Twetwa-Dube, "Exploring Group Work Strategies to Teach Computer Programming: A Case Study of First-Year and Extended Programme Students at One South African University."

[74] Hafizah Mohamad Hsbollah and Haslinda Hassan, "Creating Meaningful Learning Experiences With Active, Fun, And Technology Elements In The Problem-Based Learning Approach And Its Implications," *Malaysian Journal of Learning and Instruction* 19 (2022), https://doi.org/10.32890/mjli2022.19.1.6.

[75] Jacques Laubscher, "Extended Curriculum Programmes: A Case Study of Architecture Students at the Tshwane University of Technology, South Africa," 2021.

[76] Olga Tapalova and Nadezhda Zhiyenbayeva, "Artificial Intelligence in Education: AIEd for Personalised Learning Pathways," *Electronic Journal of E-Learning* 20, no. 5 (December 9, 2022): 639–53, https://doi.org/10.34190/ejel.20.5.2597.

Use of gamification in programming classes

A variety of strategies, including gamification, have been studied by researchers to overcome these hurdles in learning.[77] The implementation of gamification while students learn programming has been shown to decrease intimidation and even foster collaborative efforts.[78] Research conducted by Zhao et al. indicated that the use of games in the teaching and learning process benefits the students.[79] The games were designed to assist students in understanding programming concepts and to boost their confidence.

Grabner-Hagen and Kingsley also indicated that instructors are using gamification for students.[80] The research by Bai et al. highlights the importance of program-specific exercises like labs, project work, and solo assignments to master the basic principles and be skilled enough to move beyond the theoretical aspects.[81] For better engagement, a few educators adapt computer games to use as models in programming lessons. Instructors often provide students with software libraries or source code that have been simplified so that students can achieve tangible results swiftly and effortlessly.[82]

Use of an automated marking system

The use of technology in education brings about a new set of problems. On the one hand, the availability of materials online could make students lazy. Rather than truly grasping the concepts, they may resort to copying code.[83] Tasks such as programming have been automated with assessment. Automated marking systems have been developed whereby instructors' workloads are lessened as they provide instant feedback to students.[84] Still, such systems may not appropriately address the requirements of open-ended projects that involve any form of creativity and critical thinking.[85] There is a possibility that both the learners and the educators are required to show flexibility, creativity, reflective practices, and actively seek to improve their skills in relation to an emerging demand for a shifting societal context.[86]

Encouraging Collaboration and Peer Learning

Peer learning and collaboration are vital for responding to the programming difficulties faced by most first-year students. Students can support each other using group work and peer teaching as strategies.[87] The benefits of collaborative learning can best be achieved if measures are taken to provide equality of participation and contribution among group members.

---

[77] Figueiredo and Garcia-Penalvo, "Increasing Student Motivation in Computer Programming with Gamification."

[78] Chris Brown and Chris Parnin, "Nudging Students toward Better Software Engineering Behaviors," in *2021 IEEE/ACM Third International Workshop on Bots in Software Engineering (BotSE)* (IEEE, 2021), 11–15.

[79] Dan Zhao et al., "Game-Based Learning: Enhancing Student Experience, Knowledge Gain, and Usability in Higher Education Programming Courses," *IEEE Transactions on Education* 65, no. 4 (November 2022): 502–13, https://doi.org/10.1109/TE.2021.3136914.

[80] Melissa M. Grabner-Hagen and Tara Kingsley, "From Badges to Boss Challenges: Gamification through Need-Supporting Scaffolded Design to Instruct and Motivate Elementary Learners," *Computers and Education Open* 4 (December 2023): 100131, https://doi.org/10.1016/j.caeo.2023.100131.

[81] Xue Bai, Ade Ola, and Yingjin Cui, "Implementation of an Automated Programming Lab," *International Journal for Digital Society* 8, no. 2 (June 30, 2017): 1278–87, https://doi.org/10.20533/ijds.2040.2570.2017.0157.

[82] Neil C. C. Brown and Greg Wilson, "Ten Quick Tips for Teaching Programming," *PLOS Computational Biology* 14, no. 4 (April 5, 2018): e1006023, https://doi.org/10.1371/journal.pcbi.1006023.

[83] Juan Carlos López-Pimentel et al., "Sustainable Project-Based Learning Methodology Adaptable to Technological Advances for Web Programming," *Sustainability* 13, no. 15 (July 29, 2021): 8482, https://doi.org/10.3390/su13158482.

[84] Zahid Ullah et al., "The Effect of Automatic Assessment on Novice Programming: Strengths and Limitations of Existing Systems," *Computer Applications in Engineering Education* 26, no. 6 (November 29, 2018): 2328–41, https://doi.org/10.1002/cae.21974.

[85] Romeo A Botes and Imelda Smit, "A Programming Assessment Software Artefact Enhanced with the Help of Learners," n.d.

[86] João Batista, Anabela Mesquita, and Gonçalo Carnaz, "Generative AI and Higher Education: Trends, Challenges, and Future Directions from a Systematic Literature Review," *Information* 15, no. 11 (October 28, 2024): 676, https://doi.org/10.3390/info15110676.

[87] Apeanti and Essel, "Learning Computer Programming Using Project-Based Collaborative Learning"; Twetwa-Dube, "Exploring Group Work Strategies to Teach Computer Programming: A Case Study of First-Year and Extended Programme Students at One South African University."

The Present and Future of Programming

Within the first year of study, a university of technology has a curriculum that mandates C# programming. Kadar et al. suggest that the programming curriculum should begin with educational programming languages like ELISA, BASIC, and Python before transitioning to advanced languages such as C, C++, or JAVA.[88] This necessitates that the curriculum designers of first-year programming languages should consider developing a curriculum that starts with easier programming languages in the first semester of study. This is something that first-year syllabus planners on programming need to rethink regarding guiding documents aimed at the curriculum frameworks, with syllabi that begin with basic programming languages in the first semester.

According to Kadar et al., once students understand the semantics of a programming language in their initial class, they will encounter minimal difficulties in assimilating new programming languages in subsequent advanced programming courses.[89] The noticeable gap is that students are enrolled for a minimum of three years in a program where they are taught in a structured way to achieve the learning outcomes. This structure allows little flexibility because students must pass the current module to advance to subsequent modules, where each module serves as a prerequisite. Students tend to lag when repeating a failed module, which can extend the period of study by six to twelve months, depending on the duration of the module. This delay increases the number of years a student takes to complete the programme.

The use of AI for personalised learning has not been explored to assist students who are lagging behind in programming modules in completing the programme in record time. Policymakers in higher institutions must consider making a thorough assessment of the use of AI as a tool to assist students and how it can be used ethically.

In the study conducted by Sankaranarayanan et al., the authors observed that students spend two-thirds of their time engaged in collaborative problem-solving before being interrupted to examine a given solution for reflective collaboration.[90] Even though it is the solution for students to reflect on the correct answer, the students were not given time to reflect on their work. During that time, they can be asked if they had more time, what they would do differently in their work before they are presented with the correct answers. Although Sankaranarayanan et al. argue that completing a task allows the students to engage with and practice content aligned with the learning objective, followed by collaborative reflection on their learning before the next task,[91] it seems like the process short-circuits the process of reflection for the students. Modules such as programming should be given enough time for the students to construct their knowledge.

CONCLUSION

Navigating the challenges linked to coding, understanding computational thinking, and adapting to the demanding requirements of a programming module can pose several difficulties for students. This theoretical paper identified challenges faced by first-year students in the programming module and strategies to address them. Challenges such as grasping the theoretical concepts, a lack of sufficient problem-solving skills, and limited access to appropriate materials are some problems highlighted in the paper. These difficulties also impact students emotionally and psychologically. Mental health is not something that can be managed solely in class; therefore, students should be referred to wellness programmes for additional support. The paper argues that applying constructivist learning theory can help mitigate these challenges faced by first-year students in the programming module. Constructivist learning theory supports students in developing critical

[88] Kadar et al., "A Study of Difficulties in Teaching and Learning Programming: A Systematic Literature Review."
[89] Kadar et al., "A Study of Difficulties in Teaching and Learning Programming: A Systematic Literature Review."
[90] Sreecharan Sankaranarayanan et al., "Collaborative Programming for Work-Relevant Learning: Comparing Programming Practice With Example-Based Reflection for Student Learning and Transfer Task Performance," *IEEE Transactions on Learning Technologies* 15, no. 5 (October 1, 2022): 594–604, https://doi.org/10.1109/TLT.2022.3169121.
[91] Sankaranarayanan et al., "Collaborative Programming for Work-Relevant Learning: Comparing Programming Practice With Example-Based Reflection for Student Learning and Transfer Task Performance."

thinking and acquiring active knowledge, making it particularly suitable for programming studies. Some strategies to tackle these issues include enhancing teaching methods and integrating gamification into programming classes. Encouraging students to collaborate with one another can facilitate peer learning. Although the concept of AI has also emerged as a potential strategy to aid those who are falling behind, it has not yet been thoroughly explored. Different teaching methods have been mentioned, and reflective collaboration was employed in one of the mentioned studies. Offering workshops and supplementary materials for students to debug and solve problems helps them become accustomed to tackling programming material. Providing students with additional materials containing problems to solve will assist them in constructing their own knowledge. Instructors can play a crucial role in ensuring the success of mitigating the challenges faced by first-year students in programming by implementing the strategies mentioned above.

BIBLIOGRAPHY

Ahmadvand, S., and M. Khoshchehreh. "Theory And Practice: Implications Of Social Constructivism In Education." *International Journal of Humanities and Social Development Research* 7, no. 2 (January 4, 2023). https://doi.org/10.30546/2523-4331.2023.7.2.19.

Alvarez, Claudio, Maira Marques Samary, and Alyssa Friend Wise. "Modularization for Mastery Learning in CS1: A 4-Year Action Research Study." *Journal of Computing in Higher Education* 36, no. 2 (August 16,2024):546–89.https://doi.org/10.1007/s12528-023-09366-1.

Angraini, Lilis Marina, Nia Kania, and Ferit Gürbüz. "Students' Proficiency in Computational Thinking Through Constructivist Learning Theory." *International Journal of Mathematics and Mathematics Education*, February 21, 2024, 45–59. https://doi.org/10.56855/ijmme.v2i1.963.

Apeanti, Wilson Osafo, and Daniel Essel. "Learning Computer Programming Using Project-Based Collaborative Learning." *International Journal for Innovation Education and Research* 9, no. 8 (August 1, 2021): 191–207. https://doi.org/10.31686/ijier.vol9.iss8.3278.

Atiq, Zahra, and Michael C. Loui. "A Qualitative Study of Emotions Experienced by First-Year Engineering Students during Programming Tasks." *ACM Transactions on Computing Education* 22, no. 3 (September 30, 2022): 1–26. https://doi.org/10.1145/3507696.

Bai, Xue, Ade Ola, and Yingjin Cui. "Implementation of an Automated Programming Lab." *International Journal for Digital Society* 8, no. 2 (June 30, 2017): 1278–87. https://doi.org/10.20533/ijds.2040.2570.2017.0157.

Barzilai, Orly, Sofia Sherman, Moshe Leiba, and Hadar Spiegel. "Cognitive Aspects in Problem Solving: The Case of a Data Structures Course for IS Students." *Journal of Information Systems Education* 35, no. 2 (2024): 175–88. https://doi.org/10.62273/JJUB4136.

Batista, João, Anabela Mesquita, and Gonçalo Carnaz. "Generative AI and Higher Education: Trends, Challenges, and Future Directions from a Systematic Literature Review." *Information* 15, no. 11 (October 28, 2024): 676. https://doi.org/10.3390/info15110676.

Batra, Rakhi, and Zahra Atiq. "Understanding Students' Frustration and Confusion during a Programming Task: A Multimodal Approach." In *2023 IEEE Frontiers in Education Conference (FIE)*, 1–10. IEEE, 2023. https://doi.org/10.1109/FIE58773.2023.10343499.

Becker, Brett A, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. "Programming Is Hard-or at Least It Used to Be: Educational Opportunities and Challenges of Ai Code Generation." In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, 500–506, 2023.

Bhaw, Nishaal, Jeanne Kriek, and Petra le Roux. "Emotional and User-Experience Factors Influencing Student Performance: A Case Study of First-Year Online Computer Programming Assignments." *UnisaRxiv*, 2024.

Botes, Romeo A, and Imelda Smit. "A Programming Assessment Software Artefact Enhanced with the Help of Learners," n.d.

Brown, Chris, and Chris Parnin. "Nudging Students toward Better Software Engineering Behaviors." In *2021 IEEE/ACM Third International Workshop on Bots in Software Engineering (BotSE)*, 11–15. IEEE, 2021.

Brown, Neil C. C., and Greg Wilson. "Ten Quick Tips for Teaching Programming." *PLOS Computational Biology* 14, no. 4 (April 5, 2018): e1006023. https://doi.org/10.1371/journal.pcbi.1006023.

Byrkun, Liudmyla, and Larysa Liashenko. "Theory of Constructivism and a Communicative Coursebook as a Prerequisite for Organizing the First Year It Students' Independent Work at the Esp Classes." *Grail of Science*, no. 38 (April 29, 2024): 298–302. https://doi.org/10.36074/grail-of-science.12.04.2024.050.

Cao, Qi, Chee Kiat Seow, Li Hong Idris Lim, Sye Loong Keoh, Vicki Dale, Sarah Honeychurch, Nathalie Tasler, and Duncan Bremner. "Learners' Differences in Blended Learner-Centric Approach for a Common Programming Subject." *International Journal of Information and Education Technology* 13, no. 6 (2023): 906–13. https://doi.org/10.18178/ijiet.2023.13.6.1886.

Carless, David. *Educational Assessment and Data Collection Methods*. London: Routledge, 2022.

Culpeper, J., and D. Kádár. *Speech Acts and Politeness in Historical Contexts*. London: Bloomsbury, 2021.

Do, Hai-Ninh, Bich Ngoc Do, and Minh Hue Nguyen. "3How Do Constructivism Learning Environments Generate Better Motivation and Learning Strategies? The Design Science Approach." *Heliyon* 9, no. 12 (December 2023): e22862. https://doi.org/10.1016/j.heliyon.2023.e22862.

Efgivia, M. Givi, Anggi Arista, Reni Kurniawati, and Kasori. "Analysis of Behaviorism Learning Theory, STEM Learning Model and Gamification," 2021. https://doi.org/10.2991/assehr.k.211020.029.

Eneji, S. E., W. E. Ibe, M. U. Angib, and M. A. Atianashie. "The Significance Role of Programming in Information Technology (IT)." *International Journal of Information Communication Science and Technology*, 2019.

Figueiredo, José, and Francisco García-Peñalvo. "Teaching and Learning Strategies for Introductory Programming in University Courses." In *Ninth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'21)*, 746–51. New York, NY, USA: ACM, 2021. https://doi.org/10.1145/3486011.3486540.

Figueiredo, Jose, and Francisco Jose Garcia-Penalvo. "Increasing Student Motivation in Computer Programming with Gamification." In *2020 IEEE Global Engineering Education Conference (EDUCON)*, 997–1000. IEEE, 2020. https://doi.org/10.1109/EDUCON45650.2020.9125283.

Grabner-Hagen, Melissa M., and Tara Kingsley. "From Badges to Boss Challenges: Gamification through Need-Supporting Scaffolded Design to Instruct and Motivate Elementary Learners." *Computers and Education Open* 4 (December 2023): 100131. https://doi.org/10.1016/j.caeo.2023.100131.

Groher, Iris, Michael Vierhauser, Barbara Sabitzer, Lisa Kuka, Alexander Hofer, and David Muster. "Exploring Diversity in Introductory Programming Classes." In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Software Engineering Education and Training*, 102–12. New York, NY, USA: ACM, 2022. https://doi.org/10.1145/3510456.3514155.

Hidayanti, Winda Ismi, Diana Rochintaniawati, and Rika Rafikah Agustin. "The Effect of Brainstorming on Students' Creative Thinking Skill in Learning Nutrition." *Journal of Science Learning* 1, no. 2 (March 31, 2018): 44. https://doi.org/10.17509/jsl.v1i2.8738.

Javier, Billy. "Understanding Their Voices from Within: Difficulties and Code Comprehension of Life-Long Novice Programmers" 1 (August 1, 2021): 53–76.

Kadar, Rozita, Naemah Abdul Wahab, Jamal Othman, Maisurah Shamsuddin, and Siti Balqis Mahlan. "A Study of Difficulties in Teaching and Learning Programming: A Systematic Literature Review."

International Journal of Academic Research in Progressive Education and Development 10, no. 3 (August 21, 2021). https://doi.org/10.6007/IJARPED/v10-i3/11100.

Karwasz, Grzegorz P., and Katarzyna Wyborska. "How Constructivist Environment Changes Perception of Learning: Physics Is Fun." *Education Sciences* 13, no. 2 (February 12, 2023): 195. https://doi.org/10.3390/educsci13020195.

Keravnou-Papailiou, Elpida. "Figuring and Drawing: A Visual Approach to Principled Programming." *ArXiv Preprint ArXiv:2202.09229*, 2022.

Kiesler, Natalie. "Reviewing Constructivist Theories to Help Foster Creativity in Programming Education." In *2022 IEEE Frontiers in Education Conference (FIE)*, 1–5. IEEE, 2022.

Korhonen, Tiina, Laura Salo, Noora Laakso, Aino Seitamaa, Kati Sormunen, Minna Kukkonen, and Heidi Forsström. "Finnish Teachers as Adopters of Educational Innovation: Perceptions of Programming as a New Part of the Curriculum." *Computer Science Education* 33, no. 1 (January 2, 2023): 94–116. https://doi.org/10.1080/08993408.2022.2095595.

Laporte, Lieve, and Bieke Zaman. "Informing Content-Driven Design of Computer Programming Games." In *Proceedings of the 9th Nordic Conference on Human-Computer Interaction*, 1–10. New York, NY, USA: ACM, 2016. https://doi.org/10.1145/2971485.2971499.

Larsson, Andreas, and Karin Stolpe. "Hands on Programming: Teachers' Use of Metaphors in Gesture and Speech Make Abstract Concepts Tangible." *International Journal of Technology and Design Education* 33, no. 3 (2023): 901–19.

Laubscher, Jacques. "Extended Curriculum Programmes: A Case Study of Architecture Students at the Tshwane University of Technology, South Africa," 2021.

Laura-Ochoa, Leticia, Norka Bedregal-Alpaca, and Elizabeth Vidal. "Improving Computational Thinking in Nursing Students through Learning Computer Programming." *International Journal of Advanced Computer Science and Applications* 13, no. 5 (2022). https://doi.org/10.14569/IJACSA.2022.0130571.

Lodi Michael, Dario Malchiodi, Mattia Monga, Anna Morpurgo, and Bernadette Spieler. "Constructionist Attempts at Supporting the Learning of Computer Programming: A Survey." *Olympiads in Informatics* 13 (July 13, 2019): 99–121. https://doi.org/10.15388/ioi.2019.07.

López-Pimentel, Juan Carlos, Alejandro Medina-Santiago, Miguel Alcaraz-Rivera, and Carolina Del-Valle-Soto. "Sustainable Project-Based Learning Methodology Adaptable to Technological Advances for Web Programming." *Sustainability* 13, no. 15 (July 29, 2021): 8482. https://doi.org/10.3390/su13158482.

Makewa, Lazarus Ndiku. "Constructivism Theory in Technology-Based Learning." In *Technology-Supported Teaching and Research Methods for Educators*, 268–87. IGI Global, 2019. https://doi.org/10.4018/978-1-5225-5915-3.ch015.

Mbiada, Alain, Bassey Isong, and Francis Lugayizi. "A Comparative Study of Computer Programming Challenges of Computing and Non-Computing First-Year Students." *The Indonesian Journal of Computer Science* 12, no. 4 (August 30, 2023). https://doi.org/10.33022/ijcs.v12i4.3330.

Medeiros, Rodrigo Pessoa, Geber Lisboa Ramalho, and Taciana Pontual Falcao. "A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education." *IEEE Transactions on Education* 62, no. 2 (May 2019): 77–90. https://doi.org/10.1109/TE.2018.2864133.

Michaelson, Greg. "Teaching Programming with Computational and Informational Thinking." *Journal of Pedagogic Development* 5 (March 1, 2015): 51–65.

Mohamad Hsbollah, Hafizah, and Haslinda Hassan. "Creating Meaningful Learning Experiences With Active, Fun, And Technology Elements In The Problem-Based Learning Approach And Its Implications." *Malaysian Journal of Learning and Instruction* 19 (2022). https://doi.org/10.32890/mjli2022.19.1.6.

Moraiti, Ioanna, Anestis Fotoglou, and Athanasios Drigas. “Coding with Block Programming Languages in Educational Robotics and Mobiles, Improve Problem Solving, Creativity & Critical Thinking Skills.” *International Journal of Interactive Mobile Technologies (IJIM)* 16, no. 20 (October 31, 2022): 59–78. https://doi.org/10.3991/ijim.v16i20.34247.

Oladipo, Francisca Onaolapo, and Memunat A. Ibrahim. “The CodeEazee Tool Support for Computational Thinking in Python.” *European Journal of Engineering Research and Science* 3, no. 3 (March 17, 2018): 12. https://doi.org/10.24018/ejers.2018.3.3.637.

Olipas, Cris Norman. “A Phenomenological Study on the Feelings, Challenges and Difficulties Experienced by Information Technology Students in Learning Computer Programming.” *Path of Science*, July 31, 2022, 2001–6. https://doi.org/10.22178/pos.83-3.

Omer, Uzma, Muhammad Shoaib Farooq, and Adnan Abid. “Introductory Programming Course: Review and Future Implications.” *PeerJ Computer Science* 7 (July 22, 2021): e647. https://doi.org/10.7717/peerj-cs.647.

Piwek, Paul, Michel Wermelinger, Robin Laney, and Richard Walker. “Learning to Program.” In *Proceedings of the 3rd Conference on Computing Education Practice*, 1–4. New York, NY, USA: ACM, 2019. https://doi.org/10.1145/3294016.3294024.

Prokop, Yuliia, Elena Trofimenko, Nikolay Severin, and Liudmila Bukata. “An Analysis of Criteria for Choosing a First Programming Language in Universities.” In *ICTERI*, 420–25, 2019.

Qian, Yizhou, and James Lehman. “Students’ Misconceptions and Other Difficulties in Introductory Programming: A Literature Review.” *ACM Transactions on Computing Education (TOCE)* 18, no. 1 (2017): 1–24.

Ranjeeth, Lerushka, and Indira Padayachee. “Factors That Influence Computer Programming Proficiency in Higher Education: A Case Study of Information Technology Students.” *South African Computer Journal* 36, no. 1 (2024): 40–75.

Rethlefsen, M. L., & Page, M. J. (2022). PRISMA 2020 and PRISMA-S: common questions on tracking records and the flow diagram. *Journal of the Medical Library Association: JMLA*, *110*(2), 253.

Saenz, Juan Pablo, and Luigi De Russis. “On How Novices Approach Programming Exercises before and during Coding.” In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, 1–6, 2022.

San Ahmed, Rania Azad M, M Sardasht, Raouf Mahmood, Rebwar M Nabi, and Dana L Hussein. “The Impact of Teaching Materials on Learning Computer Programming Languages in Kurdistan Region Universities and Institutes.” *Kurdistan Journal of Applied Research* 3, no. 1 (2018): 27–33.

Sankaranarayanan, Sreecharan, Siddharth Reddy Kandimalla, Christopher A. Bogart, R. Charles Murray, Michael Hilton, Majd F. Sakr, and Carolyn P. Rose. “Collaborative Programming for Work-Relevant Learning: Comparing Programming Practice With Example-Based Reflection for Student Learning and Transfer Task Performance.” *IEEE Transactions on Learning Technologies* 15, no. 5 (October 1, 2022): 594–604. https://doi.org/10.1109/TLT.2022.3169121.

Sarkis-Onofre, Rafael, Ferrán Catalá-López, Edoardo Aromataris, and Craig Lockwood. “How to Properly Use the PRISMA Statement.” *Systematic Reviews* 10, no. 1 (2021): 117.

Singh, Sarita. “Identifying Learning Challenges Faced by Novice/Beginner Computer Programming Students: An Action Research Approach.” In *QuASoQ/SEED@APSEC*, 2022. https://api.semanticscholar.org/CorpusID:255968169.

Smith, Julie. “Constructivism in Computer Science Education.” In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2*, 1171, 2022.

Sobral, Sónia Rolland. “Teaching and Learning to Program: Umbrella Review of Introductory Programming in Higher Education.” *Mathematics* 9, no. 15 (July 23, 2021): 1737. https://doi.org/10.3390/math9151737.

Sobral, Sónia Rolland. "The First Programming Language and Freshman Year in Computer Science: Characterization and Tips for Better Decision Making." In *Trends and Innovations in Information Systems and Technologies*, 162–74, 2020. https://doi.org/10.1007/978-3-030-45697-9_16.

Tapalova, Olga, and Nadezhda Zhiyenbayeva. "Artificial Intelligence in Education: AIEd for Personalised Learning Pathways." *Electronic Journal of E-Learning* 20, no. 5 (December 9, 2022): 639–53. https://doi.org/10.34190/ejel.20.5.2597.

Theodoropoulos, Anastasios, and George Lepouras. "Augmented Reality and Programming Education: A Systematic Review." *International Journal of Child-Computer Interaction* 30 (December 2021): 100335. https://doi.org/10.1016/j.ijcci.2021.100335.

Twetwa-Dube, Sithandiwe Pornelia. "Exploring Group Work Strategies to Teach Computer Programming: A Case Study of First-Year and Extended Programme Students at One South African University." *International Journal of Learning, Teaching and Educational Research* 23, no. 7 (July 30, 2024): 271–88. https://doi.org/10.26803/ijlter.23.7.14.

Ullah, Zahid, Adidah Lajis, Mona Jamjoom, Abdulrahman Altalhi, Abdullah Al-Ghamdi, and Farrukh Saleem. "The Effect of Automatic Assessment on Novice Programming: Strengths and Limitations of Existing Systems." *Computer Applications in Engineering Education* 26, no. 6 (November 29, 2018): 2328–41. https://doi.org/10.1002/cae.21974.

Weintrop, David, and Uri Wilensky. "Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms." *ACM Transactions on Computing Education* 18, no. 1 (March 31, 2018): 1–25. https://doi.org/10.1145/3089799.

Yilmaz, Ramazan, and Fatma Gizem Karaoglan Yilmaz. "Augmented Intelligence in Programming Learning: Examining Student Views on the Use of ChatGPT for Programming Learning." *Computers in Human Behavior: Artificial Humans* 1, no. 2 (August 2023): 100005. https://doi.org/10.1016/j.chbah.2023.100005.

Zavaleta Bernuy, Angela, Andrew Chung, Alana Hodge, Ayesha Tayyiba, Michael Liut, and Andrew Petersen. "Student Transitions Through an Entire Computing Program." In *The 26th Western Canadian Conference on Computing Education*, 1–7. New York, NY, USA: ACM, 2024. https://doi.org/10.1145/3660650.3660661.

Zhao, Dan, Cristina Hava Muntean, Adriana E. Chis, Gregor Rozinaj, and Gabriel-Miro Muntean. "Game-Based Learning: Enhancing Student Experience, Knowledge Gain, and Usability in Higher Education Programming Courses." *IEEE Transactions on Education* 65, no. 4 (November 2022): 502–13. https://doi.org/10.1109/TE.2021.3136914.

ABOUT AUTHOR

Moretlo Tlale-Mkhize serves as a Departmental Manager and lecturer at the Central University of Technology in Free State, where she specializes in Graphical User Interface (GUI) Design. She holds PhD in Computer Science and Information Systems from North West University. With a strong academic and professional focus, her research spans several key areas, including teaching and learning technologies, Artificial Intelligence (AI), Human-Computer Interaction (HCI), and Reflective Practice. Throughout her career, Moretlo has actively contributed to the academic community by presenting her research and insights at both national and international conferences, allowing her to share valuable knowledge and findings with a wider audience. Her work is dedicated to advancing the intersection of technology, user experience, and education, as she continues to explore innovative approaches in these fields. Additionally, her commitment to fostering reflective and interactive teaching practices plays a key role in shaping modern educational methodologies in the realm of technology and design.