Needs Analysis for Developing an Educational Hybrid Board Game for Python Programming

Mohamad Firdaus Che Abdul Rani[1,2,*], Nor Hafizah Adnan[2], Ahmad Zamri Mansor[2], Melor Md Yunus[2], Imane El Mourabit[3], and Tan Chin Ike[1]

[1]School of Computing, Asia Pacific University of Technology and Innovation, Malaysia
[2]Faculty of Education, Universiti Kebangsaan Malaysia, Malaysia
[3]Faculty of Sciences Ain Chock, University Hassan II of Casablanca, Morocco
Email: firdaus@apu.edu.my (M.F.C.A.R.); norhafizah@ukm.edu.my (N.H.A.); azamri@ukm.edu.my (A.Z.M.); melor@ukm.edu.my (M.M.Y.); imane.elmourabit-etu@etu.univh2c.ma (I.E.M.); tan.chinike@apu.edu.my (T.C.I.)
[*]Corresponding author
Manuscript received April 28, 2025; revised June 5, 2025; accepted July 15, 2025; published December 9, 2025

***Abstract*—This study investigates the challenges faced by introductory-level Information and Communications Technology (ICT) higher education students in learning Python programming and examines the potential of using a hybrid board game as a formative assessment tool. A needs analysis was conducted through a survey of 143 diploma-level students, focusing on their understanding of Python topics, difficulties encountered, and preferred learning methods. Key challenges identified include difficulties in understanding advanced topics, debugging, and program design. The study adopts a Design and Development Research (DDR) approach, with this paper focusing on the initial needs analysis phase. The results highlight a demand for interactive, formative tools to enhance programming comprehension and motivation. Findings will inform the design of a hybrid board game that blends physical and digital elements to improve programming comprehension, support formative assessment, and foster learner engagement. Future research will focus on developing the game, defining its components, aligning game mechanics with educational goals, and evaluating its usability and effectiveness.**

***Keywords*—needs analysis, higher education, hybrid board game, python programming, formative assessment, programming learning difficulties**

I. Introduction

Introductory programming courses are foundational components in higher education curricula for Information Technology (IT), Computer Science (CS), and related fields. These courses, typically offered in the early stages of undergraduate programs, are designed to equip students with algorithmic thinking and problem-solving skills through exposure to essential programming concepts such as abstraction, data structures, control flow, and software development principles [1]. Languages like Python and Java are frequently chosen for instruction due to their readability and relevance in industry applications [2, 3].

Despite the critical importance of these courses, many introductory-level learners encounter persistent challenges in mastering programming. Students often struggle to understand abstract concepts, debug code effectively, and design functioning programs [4, 5]. These difficulties can impede learning, reduce motivation, and negatively impact academic performance [6]. In the Malaysian higher education context, diploma-level Information and Communications Technology (ICT) students similarly face these challenges, which are compounded by the lack of interactive, engaging, and accessible instructional tools that cater to their learning needs [7].

Although digital learning tools have advanced significantly in recent years, there is still a notable gap in the availability of hybrid educational tools, which refer to learning environments that combine both physical and digital components, especially in programming education. Hybrid tools offer the potential to engage students more holistically by combining tactile, visual, and interactive elements [8, 9]. The limited use of such tools in programming education highlights a missed opportunity to enhance formative assessment and support concept mastery [10].

To address this gap, this study investigates the specific learning needs of introductory-level ICT students in Python programming. The aim is to identify how a hybrid board game could be developed to align with these needs and serve as a formative assessment tool. Accordingly, this paper addresses the following research questions: 1) What are the learning needs of introductory-level ICT higher education students in mastering computer programming concepts? 2) How can a hybrid board game be designed to address these learning needs and function as a formative assessment tool?

To contextualize this study, a literature review is presented in the next section, highlighting key challenges in programming education, limitations of existing tools, and the theoretical frameworks guiding this research.

II. Literature Review

A. Challenges in Learning Programming

Students encountering programming for the first time often face considerable challenges that affect their comprehension and performance. These difficulties stem largely from the abstract and logical nature of programming, which demands a high level of cognitive engagement. Introductory-level learners commonly struggle with understanding syntax, applying logical structures, and debugging errors [4]. Debugging is particularly difficult, as many students lack the skills to systematically identify and correct errors, often leading to frustration and disengagement [11]. In addition, misconceptions about program behavior and incomplete conceptual understanding further impede learning [5], especially among ICT diploma students in Malaysian institutions who often lack tailored support tools [7].

B. Limitations of Current Programming Education Tools

To address the challenges faced by introductory-level programmers, educators have developed a range of digital

tools, including block-based environments (e.g., Scratch, Blockly), text-based platforms (e.g., Repl.it, Codecademy), and game-based tools (e.g., CodeCombat, LightBot). These tools aim to simplify programming logic, reduce syntactic complexity, and enhance motivation through gamification and interactive feedback [12, 13].

While effective in supporting certain aspects of learning, such tools often share critical limitations. Many operate in fully digital environments that prioritize individual learning, lacking physical interaction and social collaboration, which are the factors known to support memory, motivation, and reduced cognitive load [14]. Block-based tools are helpful for beginners but may hinder the transition to text-based languages, as they often abstract away syntax and deeper problem-solving processes [7]. Text-based platforms provide real-world coding experience but tend to overlook tactile and peer-based engagement [15]. Similarly, digital game-based tools gamify logic practice but generally omit hands-on interaction and social feedback loops [13].

C. Potential and Application of Hybrid Board Games

Game-based learning has gained significant traction for its ability to enhance learner engagement, motivation, and knowledge retention across diverse disciplines [16]. Hybrid board games, which combine tangible components with digital interfaces [17], offer a unique educational advantage by integrating physical interaction with immediate feedback mechanisms. These games have been successfully implemented in domains such as chemistry [18, 19], environmental science [9], and teacher education [20], where they have been shown to foster collaboration, engagement, and deeper conceptual understanding.

However, no hybrid board games have yet been developed specifically for programming education in higher education. Existing tools in this field remain primarily digital, such as mobile apps or online coding platforms [10, 21]. While these digital tools can gamify learning and provide automated feedback, they often fail to incorporate the tactile and social learning benefits offered by hybrid games.

This gap represents a missed opportunity to support introductory-level programming learners through physical-digital integration. Therefore, this study proposes the future development of a hybrid board game tailored to Python programming, drawing on proven successes in other subject areas to introduce an engaging, formative, and multimodal approach to programming education.

D. Theoretical Frameworks Supporting Game-Based Learning

The design of effective educational games should be grounded in sound learning theories. Cognitive Load Theory (CLT) is especially relevant for programming education, where learners must process abstract information and manage multiple cognitive demands simultaneously. CLT emphasizes optimizing intrinsic cognitive load while minimizing extraneous load through instructional strategies such as worked examples, scaffolding, and interactive feedback [22, 23]. These strategies can be incorporated into game mechanics to improve learning outcomes and reduce cognitive overload.

In parallel, Constructivist Learning Theory underscores the importance of active, experiential, and socially situated learning. Rooted in the works of Piaget and Vygotsky, constructivism asserts that learners construct knowledge through interaction with their environment and collaboration with peers. In programming education, this is reflected in practices like project-based learning, peer debugging, and hands-on activities [16]. Hybrid board games, by nature, embody constructivist principles by enabling students to learn through doing, sharing, and exploring within a structured yet engaging environment. These theories collectively support the development of educational games that reduce cognitive barriers and foster collaborative, active learning, making them particularly suitable for addressing the challenges faced by introductory-level programming students.

E. Contextual Justification for Game Design

Given the challenges faced by introductory-level programming students, the promising outcomes of hybrid games in other educational contexts, and the alignment with Cognitive Load Theory and constructivist learning principles, there is a strong case for incorporating hybrid board games into programming instruction. However, the success of such a tool depends on a clear understanding of learners' specific needs and preferences. Conducting a needs analysis is therefore a critical first step toward designing a pedagogically grounded and contextually relevant educational hybrid board game. This study addresses this gap by examining the experiences and expectations of diploma-level ICT students at a private higher education institution in Malaysia, offering context-specific insights that can inform the initial stages of educational game development.

III. Methodology

A. Study Design

Design and Development Research (DDR) is dedicated to creating new knowledge and validating existing practices [24]. It operates on theory, drives action, and centres on participants, merging pragmatic design with traditional research methods [25]. DDR is divided into two types: Type 1, which focuses on research on products and tools, and Type 2, which focuses on research on design and development models. DDR's empirical nature often mirrors scientific problem-solving processes.

This study employed DDR Type 1. The DDR Type 1 approach in this study provides a substantial method for addressing the gap in understanding and application of programming concepts by introductory-level ICT higher education students. Additionally, this research design was chosen because it allows the researcher to evaluate the usability and acceptance of a hybrid board game as a formative assessment tool for reinforcing programming concepts.

B. Population and Study Sample

In the context of this study, the research population comprises introductory-level students pursuing a Diploma in Information and Communication Technology (ICT) at a Private Higher Education Institution in Klang Valley, Malaysia. These students, who have already completed the introductory programming course, Programming with Python (PWP), have been purposefully selected for this study based on their characteristics and their willingness to participate.

The average number of students enrolled in the introductory computer programming course of the Diploma across four intakes in 2023 is 172. The minimum sample size is calculated using Cochran's [26] formula and Krejcie & Morgan's [27] method. According to these calculations, the sample size for this study should be at least 120 to ensure the findings are representative of the population. A total of 143 respondents voluntarily answered the survey, which exceeds the minimum required sample size and ensures robust and representative findings.

C. Study Instrument

For the purposes of this research, a cross-sectional survey was utilized. The survey is designed to identify the needs of introductory-level ICT higher education students when learning computer programming concepts. It is organized into the following sections: 1) Demographic background, 2) Level of understanding of different Python programming topics, 3) Difficulties while learning programming, 4) Situations that would help to learn programming more effectively, 5) Materials that would help to learn programming, 6) Factors that lead to perform poorly in programming subject, and 7) Level of agreement with using interactive environment in learning programming. The survey items are adapted from Tan *et al.* [28]. The questionnaires were distributed via Microsoft Forms. A 4-point Likert scale was used for section 2, with the following scale: 1 = Never Use, 2 = Weak, 3 = Moderate, 4 = Strong. For sections 3 to 5, a 5-point Likert scale was used, with the following scale: 1 = strongly disagree, 2 = disagree, 3 = neither agree nor disagree, 4 = agree, and 5 = strongly agree. The reason for using different Likert scales is to accurately capture the varying degrees of understanding and agreement among the respondents [29].

The survey consisted of seven sections designed to assess students' programming learning needs. The complete set of questionnaire items is provided in Appendix A.

D. Validity and Reliability of the Instrument

The importance of an instrument's validity and reliability is emphasized to ensure accurate measurements [30]. The survey questionnaires underwent thorough consideration for their validity and reliability. In this study, content validation was carried out by experts in the field of computer science education. All experts agreed on the face validity, content validity, and construct validity of the questionnaire.

A pilot study involving 35 students was conducted to assess the reliability of the instrument and refine the questionnaire items. A reliability analysis using Cronbach's Alpha was then performed to evaluate the internal consistency of the questionnaire items. Cronbach's Alpha values greater than 0.9 are considered very high and acceptable, while values ranging from 0.7 to 0.89 are also acceptable and indicate high internal consistency. Values between 0.30 and 0.69 are considered average but still acceptable, although they reflect lower reliability [31].

Although the survey employed both 4-point and 5-point Likert scales across different sections to suit the specific nature of the constructs being measured, the internal consistency of each section was assessed independently. Cronbach's Alpha was calculated separately for each scale type, ensuring that the use of different Likert formats did not compromise the reliability of the responses. This approach maintained methodological rigor and supported the robustness of the instrument's design.

Two sections were not subjected to Cronbach's Alpha analysis. First, the demographic background section collects factual information rather than measuring a latent construct, making internal consistency analysis unnecessary [32]. Second, Section 7 contains only one item, and Cronbach's Alpha requires multiple items to assess internal consistency; therefore, calculation is not applicable in this case [33].

As presented in Table 1, the Cronbach's Alpha values for all applicable sections exceeded 0.7, indicating that the questionnaire demonstrates high reliability.

Table 1. Cronbach's Alpha values for survey sections

No	Section	Cronbach's Alpha	N of Items
1	Demographic background	-	2
2	Level of understanding of different Python programming topics	0.893	15
3	Difficulties while learning programming	0.850	7
4	Situations that would help to learn programming more effectively	0.710	4
5	Materials that would help to learn programming	0.857	7
6	Factors that lead to perform poorly in programming subject	0.843	8
7	Level of agreement with using interactive environment in learning programming	-	1

E. Data Analysis

The collected data were descriptively analyzed to derive percentage figures, mean values, and standard deviations. The instruments aimed to identify the specific needs of introductory-level ICT higher education students in learning computer programming concepts. It also aided in identifying the requirements needed for the design and development of a hybrid board game for formative assessment in Python programming. The levels of understanding of different Python programming topics, difficulties while learning programming, situations that would help to learn programming more effectively, materials that would help to learn programming, factors that lead to poor performance in programming subjects and level of agreement with using interactive environment in learning programming were analyzed in terms of percentages, mean values, and standard deviations. Table 2 presents the interpretation of mean values, adapted from Sullivan & Artino [34].

Table 2. Interpretation guide for mean values based on Likert scales

Section with…	Mean Value	Interpretation
4-point Likert scale	0.00–0.75	Low understanding
	0.76–1.50	Weak understanding
	1.51–2.25	Moderate understanding
	2.26–3.00	Strong understanding
5-point Likert scale	1.00–2.33	Low agreement
	2.34–3.67	Moderate agreement
	3.68–5.00	Strong agreement

To explore deeper relationships between variables, inferential analyses were conducted. Pearson correlation analysis was used to examine associations between students' programming difficulties and their understanding of Python topics. Pearson's r is appropriate for identifying linear relationships between continuous variables and is widely used in educational research when examining associations

among survey-derived scales [35, 36].

Additionally, multiple linear regression analysis was employed to identify significant predictors of students' agreement with using interactive learning environments. This method allows for the assessment of how multiple independent variables (e.g., learning difficulties, preferences, material usefulness) collectively influence a dependent variable, offering predictive insights that go beyond correlation alone [37]. All statistical analyses were conducted using IBM SPSS Statistics Version 29.

IV. Results and Discussion

Based on the results of need analysis, the students' needs in learning computer programming concepts which includes the 1) Demographic background, 2) Level of understanding of different Python programming topics, 3) Difficulties while learning programming, 4) Situations that would help to learn programming more effectively, 5) Materials that would help to learn programming, 6) Factors that lead to perform poorly in programming subject, and 7) Level of agreement with using interactive environment in learning programming are analyzed and discussed.

A. Demographic Background

To better understand the background of the participants in this study, demographic data were collected. This information provides insight into the age distribution and the types of devices owned by the students, which may influence their learning experiences and preferences. In Malaysia, students can directly join a diploma program after completing school [38], which means that the participants in this study are introductory-level students in higher education. The demographic attributes of the respondents are summarized in the Table 3 below.

Table 3. Demographic background of respondents

Item	Demographic attribute	Frequency (n)	%
Age	18–28 years	143	100
Device owned	Mobile phone	143	100
	Desktop	2	1.40
	Tablet	23	16.08
	Laptop	143	100

The demographic data indicate that all participants are in between 18 to 28 years old. Additionally, all participants own a mobile phone and a laptop, which suggests that these devices are the primary tools for their learning activities. The ownership of tablets and desktops is less common. Understanding these demographic characteristics is crucial for tailoring the design of the hybrid board game to meet the specific needs and preferences of these introductory-level ICT students in higher education.

B. Level of Understanding of Different Python Programming Topics

Before designing the hybrid board game for formative assessment, it is essential to understand students' levels of comprehension across various Python programming topics. This analysis involved evaluating students' understanding of 15 programming topics. Students rated their level of understanding using a 4-point Likert scale, where 1 = Never Use, 2 = Weak, 3 = Moderate, and 4 = Strong. The descriptive findings are presented in Fig. 1 and Table 4.

The results indicate that students demonstrated varying levels of understanding across different programming topics. Generally, students showed stronger comprehension in basic programming concepts such as Variables and Data Types (Mean = 2.20, SD = 0.64), Input/Output Handling (Mean = 2.18, SD = 0.62), and Logic Structures: Decision (Mean = 2.14, SD = 0.64) and Iteration (Mean = 2.20, SD = 0.60). These topics exhibited relatively higher mean scores and lower standard deviations, suggesting a more consistent understanding among the students.

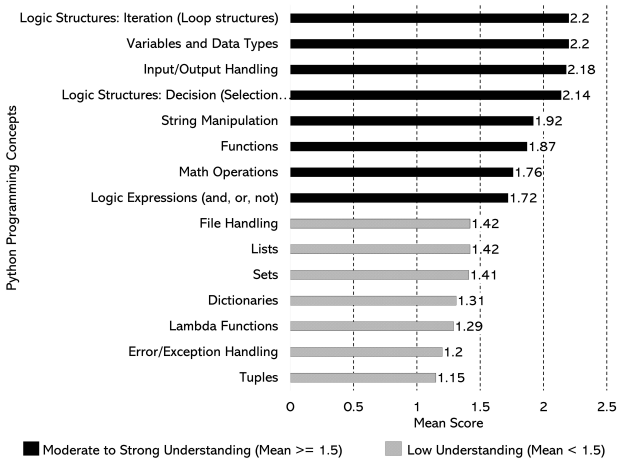

Fig. 1. Mean understanding scores for python programming topics.

Table 4. Mean and Standard Deviation (SD) of understanding of python programming topics

Python Programming Topics	Mean	SD
1. Logic Structures: Iteration (Loop structures)	2.2	0.6
2. Variables and Data Types	2.2	0.64
3. Input/Output Handling	2.18	0.62
4. Logic Structures: Decision (Selection structures, If, Else, Elif)	2.14	0.64
5. String Manipulation	1.92	0.59
6. Functions	1.87	0.57
7. Math Operations	1.76	0.7
8. Logic Expressions (and, or, not)	1.72	0.61
9. Lists	1.42	0.68
10. File Handling	1.42	0.81
11. Sets	1.41	0.74
12. Dictionaries	1.31	0.73
13. Lambda Functions	1.29	0.81
14. Error/Exception Handling	1.2	0.85
15. Tuples	1.15	0.74

Conversely, students exhibited greater difficulties with more advanced topics, including Logic Expressions (Mean = 1.72, SD = 0.61), Lambda Functions (Mean = 1.29, SD = 0.81), and Error/Exception Handling (Mean = 1.20, SD = 0.85). The lower mean scores and higher standard deviations for these topics suggest variability in student comprehension and highlight areas where additional instructional support may be needed.

In designing the hybrid board game for Python programming formative assessment, it is important to apply principles from Constructivist Learning Theory [39, 40], which emphasize building on learners' prior knowledge and providing scaffolding [16]. Accordingly, the board game should introduce foundational concepts, such as Variables and Data Types and Input/Output Handling, early in gameplay, gradually progressing toward more complex topics like Logic Expressions and Error/Exception Handling. This scaffolded approach ensures that learners consolidate

their understanding of basic programming concepts before engaging with more advanced material. Incorporating interactive elements, immediate feedback, and collaborative learning opportunities within the game design can further enhance engagement and facilitate deeper mastery of Python programming concepts.

C. Difficulties While Learning Programming

The next part of the needs analysis aimed to identify the specific challenges that introductory-level higher education students face while learning programming. Understanding these difficulties is crucial for informing the design of the hybrid board game, ensuring it effectively addresses students' needs and supports their learning process. Students rated their level of agreement with various difficulty statements using a 5-point Likert scale, where 1 = Strongly Disagree and 5 = Strongly Agree. The descriptive findings are summarized in Table 5.

Table 5. Mean and Standard Deviation (SD) of students' reported difficulties in learning programming

Difficulties while learning programming	Mean	SD
1. Designing a program to solve certain task	3.27	0.88
2. Dividing functionality into procedures	3.23	0.87
3. Learning the programming language syntax	3.20	0.92
4. Finding bugs from my own program	3.48	0.93
5. Understanding basic concept of programming structures	2.96	1.02
6. Using program development environment	3.10	0.93
7. Gaining access to computers/ networks	2.92	1.04

The results reveal that students encounter significant challenges across several areas of programming. The highest mean score (Mean = 3.48, SD = 0.93) was observed for "Finding bugs from my own program", indicating that debugging is a major difficulty among students. Other notable challenges include "Designing a program to solve certain tasks "(Mean = 3.27, SD = 0.88) and "Dividing functionality into procedures" (Mean = 3.23, SD = 0.87). According to the interpretation of mean values, these difficulties fall within the range of moderate agreement (Mean 2.34–3.67), suggesting they are commonly experienced issues among students at the introductory level of programming.

These findings align with previous research, which highlights debugging, problem-solving, and understanding programming structures as persistent challenges for beginners [11, 41]. Without targeted interventions, these obstacles can hinder students' learning experiences and progression in programming education.

Addressing these challenges is essential in the design of the hybrid board game. Specific modules and activities should be incorporated to directly tackle these areas, such as including debugging exercises, providing structured guides for program design, and offering scaffolded tasks that focus on breaking down functionalities into procedures. By targeting these key difficulties, the hybrid board game can serve as a formative assessment tool that supports students in overcoming their obstacles and improving their overall programming skills.

D. Situations That Would Help Students Learn Programming More Effectively

This part of the needs analysis questionnaire explored the learning environments and methods that students believe would enhance their ability to learn programming. Understanding students' preferred learning contexts is essential for designing educational tools that align with their needs and learning styles. Students rated each situation on a 5-point Likert scale, where 1 = Strongly Disagree and 5 = Strongly Agree. The descriptive findings are summarized in Table 6.

Table 6. Mean and Standard Deviation (SD) of preferred situations for learning programming

Situations that would help to learn programming more effectively	Mean	SD
1. In lecture, practical or lab sessions	3.84	0.88
2. Consultation or discussion with lecturers, tutors, seniors or friends	3.92	0.87
3. In small group exercise sessions	3.62	0.96
4. While working alone on programming coursework	3.57	0.95

The results indicate that students perceive certain environments as more conducive to learning programming effectively. The highest mean score (Mean = 3.92, SD = 0.87) was recorded for "Consultation or discussion with lecturers, tutors, seniors, or friends", suggesting a strong preference for collaborative and interactive learning experiences. Other highly rated situations include "In lecture, practical, or lab sessions" (Mean = 3.84, SD = 0.88) and "In small group exercise sessions" (Mean = 3.62, SD = 0.96). "While working alone on programming coursework" also received a relatively high score (Mean = 3.57, SD = 0.95), although slightly lower than collaborative scenarios.

These preferences align with prior research on programming education, which emphasizes the importance of social learning and collaborative engagement. Collaborative learning environments, where students can interact with peers and instructors, have been shown to enhance both learning outcomes and student motivation [16]. For instance, Daradoumis *et al.* [42] reported that distributed online programming environments that promote collaboration significantly improve students' self-efficacy and intrinsic motivation. Additionally, the inclusion of practical exercises and lab-like simulations has been found to reinforce programming skills and enhance student engagement [43].

These findings underscore the need to incorporate collaborative and interactive elements into the design of the hybrid board game. The game should facilitate opportunities for discussion, peer consultation, and teamwork, potentially through multiplayer modes or team-based challenges. Furthermore, embedding practical exercises and simulated lab activities within the game can replicate the types of learning environments that students find most effective, thereby enhancing engagement and supporting the development of programming competencies.

E. Materials That Would Help to Learn Programming

Following the identification of preferred learning situations, the needs analysis also explored the types of learning materials that students believe would aid their programming education. Understanding these preferences is crucial for designing educational tools that align with students' learning needs and styles. Students rated each material on a 5-point Likert scale, where 1 = Strongly Disagree and 5 = Strongly Agree. The descriptive findings are presented in Table 7.

The results indicate that students highly value certain

materials for supporting their programming learning. The highest mean score (Mean = 3.93, SD = 0.94) was observed for "Example programs", suggesting that students strongly prefer practical examples that demonstrate real applications of programming concepts. Other materials receiving high mean scores include "Exercise questions and answers" (Mean = 3.90, SD = 0.91) and "Interactive visualizations" (Mean = 3.90, SD = 0.89), both falling within the strong agreement range (3.68–5.00). These results underscore the importance of integrating diverse, interactive, and practical learning materials to accommodate different learning styles and enhance students' comprehension.

Table 7. Mean and Standard Deviation (SD) of preferred learning materials

Materials that would help to learn programming	Mean	SD
1. Example programs	3.93	0.94
2. Exercise questions and answers	3.90	0.91
3. Interactive visualizations	3.90	0.89
4. Programming course book	3.41	0.95
5. Still pictures of programming structures	3.46	0.93
6. Lecture notes	3.63	0.89
7. Forums	3.56	0.88

These findings align with previous research, which emphasizes the effectiveness of using practical examples and interactive elements in programming education. Rovshenov and Sarsar [41] found that exposure to practical programming examples significantly enhances student learning outcomes and academic success. Similarly, Anindyaputri *et al.* [44] emphasized the importance of adaptive learning systems that offer interactive visualizations and exercises tailored to diverse learner needs.

Incorporating these preferred materials into the design of the hybrid board game can enhance its effectiveness as a formative assessment tool. The game can integrate features such as embedded example programs, exercise-based challenges, and interactive visualizations that simulate real-world programming applications. This approach aligns with findings from Adipat *et al.* [16], who highlighted the value of practical examples in game-based learning environments. Furthermore, offering varied exercise questions supports repeated practice and skill reinforcement, as advocated by Videnovik *et al.* [21]. The use of interactive visualizations within the game can also help students better grasp complex programming concepts, consistent with the findings of Chukusol *et al.* [10] in their work on hybrid learning platforms using virtual board games.

F. Factors that Lead to Perform Poorly in Programming Subject

The needs analysis questionnaire also investigated the factors that students perceive as contributing to poor performance in programming subjects. Identifying these barriers is critical for informing strategies to enhance learning effectiveness and create more supportive educational environments. Students rated each factor on a 5-point Likert scale, where 1 = Strongly Disagree and 5 = Strongly Agree. The descriptive findings are presented in Table 8.

The findings reveal that several factors are perceived as significant contributors to poor learning outcomes in programming. The highest mean score (Mean = 3.78, SD = 0.88) was recorded for "Less examples in practical use are shown", indicating that students strongly associate a lack of practical examples with weaker performance. Other highly rated factors include "Syllabi focuses too much on theory" (Mean = 3.75, SD = 0.86) and "Syllabi coverage per semester is too wide" (Mean = 3.54, SD = 0.93). These factors suggest a perceived imbalance between theoretical instruction and practical application in current programming curricula.

Table 8. Mean and Standard Deviation (SD) of factors contributing to poor performance in programming

Factors that lead to perform poorly in programming subjects	Mean	SD
1. Less examples in practical use are shown	3.78	0.88
2. Syllabi focuses too much on theory	3.75	0.86
3. Syllabi coverage per semester is too wide	3.54	0.93
4. Lack of interest to learn	3.45	1.02
5. Learning environment that is not conducive	3.44	0.89
6. Presentation of instructors and their attention on students	3.29	0.98
7. Teaching methodology is less effective	3.36	0.95
8. Computers provided in labs are not functioning well	3.16	1.11

Additional factors such as "Lack of interest to learn" (Mean = 3.45, SD = 1.02) and "Learning environment that is not conducive" (Mean = 3.44, SD = 0.89) received moderate agreement, highlighting the impact of intrinsic motivation and external learning conditions on students' academic performance.

These findings emphasize the need for a more balanced, practice-oriented approach to programming education. Prior research supports this perspective: Rovshenov and Sarsar [41] demonstrated that incorporating practical examples into programming education significantly enhances students' learning outcomes and academic success. Similarly, Anindyaputri *et al.* [44] highlighted that an overemphasis on theoretical content can reduce student engagement and recommended adaptive learning environments that cater to diverse learning preferences through interactive and applied learning opportunities.

Addressing these issues through curriculum redesign, such as integrating more practical exercises, limiting overly broad syllabi, and fostering more engaging and supportive learning environments, can significantly improve student performance and satisfaction in programming education.

G. Level of Agreement on Using Interactive Environments for Learning Programming

Following the identification of factors contributing to poor performance in programming subjects, the needs analysis also examined students' perceptions of using interactive environments to support their programming learning. Understanding students' views on interactive learning is crucial for designing educational tools that effectively engage and assist them.

The analysis revealed that students demonstrated a strong overall agreement regarding the use of interactive environments for learning programming, with a mean score of 3.94 and a standard deviation of 1.00. This finding suggests that students highly value dynamic and engaging learning tools to enhance their understanding, application, and retention of programming concepts.

These results are consistent with previous research emphasizing the importance of interactive elements in educational design. Adipat *et al.* [16], for instance, highlighted that interactive features in game-based learning

significantly enhance student engagement, motivation, and learning outcomes. Interactive environments provide immediate feedback, promote active learning, and accommodate different learning styles, all of which contribute to more effective educational experiences.

H. Inferential Statistical Analysis

To prepare for inferential statistical analysis, composite scores were computed for selected variables. The difficulty score was calculated as the mean of the seven items in Section III (Difficulties While Learning Programming). The collaborative learning preference score was derived by averaging two items from Section IV that reflect peer interaction: 1) consultation with others, and 2) participation in small group exercises. The perceived usefulness of instructional materials score was computed as the mean of three items from Section V: 1) example programs, 2) exercise questions and answers, and 3) interactive visualizations. These composite scores were then treated as continuous predictor variables in the multiple linear regression analysis.

To deepen the analysis beyond descriptive statistics, inferential techniques were employed to explore relationships between students' programming difficulties, their understanding of Python programming concepts, and their perceptions of interactive learning environments.

1) Correlation analysis

A Pearson correlation analysis was conducted to examine the relationship between students' reported programming difficulties and their understanding of Python topics. The results indicated a significant negative correlation between difficulty in debugging and understanding of advanced Python concepts, such as error handling and lambda functions ($r = -0.42$, $p < 0.01$). Similarly, difficulties in designing programs were negatively correlated with understanding of functions and data structures ($r = -0.37$, $p < 0.01$). These findings suggest that students who reported greater difficulty in problem-solving and debugging also demonstrated lower levels of understanding in key programming areas.

Furthermore, a positive correlation was observed between students' preference for interactive learning environments and their perceived usefulness of visual and example-based materials ($r = 0.45$, $p < 0.01$). This suggests that students who are more inclined toward interactive formats also place high value on concrete, practical learning resources.

2) Regression analysis

To further explore predictive relationships, a multiple linear regression analysis was performed with overall agreement on using interactive environments as the dependent variable. The independent variables included students' difficulty scores, preference for collaborative learning situations, and perceived usefulness of various programming materials.

The regression model was statistically significant, $F(3, 139) = 12.45$, $p < 0.001$, and accounted for approximately 21% of the variance in students' agreement with using interactive environments ($R^2 = 0.21$). The results indicated that preference for collaborative learning environments was the strongest positive predictor ($\beta = 0.38$, $p < 0.001$), followed by the perceived usefulness of interactive visualizations ($\beta = 0.29$, $p = 0.003$). In contrast, difficulty scores were not a statistically significant predictor ($p = 0.076$), suggesting that students' openness to interactive solutions is more strongly influenced by their preferred learning methods and instructional materials rather than the challenges they encounter during programming tasks.

These results indicate that while programming challenges do exist, students' openness to interactive learning environments is more strongly influenced by their engagement preferences and the value they assign to specific types of instructional content. Table 9 summarizes the key findings from the needs analysis and their direct implications for the design of the hybrid board game.

Table 9. Summary of key findings and their implications for hybrid board game design

Key Finding	Student Need or Insight	Design Implication / Game Feature
Low understanding of advanced topics (e.g., lambda functions, error handling)	Scaffolding is needed to build from basic to complex concepts	Game levels progress from basic syntax to complex logic via tiered challenges
Debugging is the most difficult task reported	Students lack confidence in finding and fixing bugs	Include scenario-based debugging mini-games with feedback hints
Strong preference for collaborative learning	Students prefer peer support and discussion	Multiplayer or cooperative gameplay with team-based missions
High value placed on example programs and interactive visualizations	Students benefit from seeing working code and visual structures	Game integrates example code cards and visual aids linked by QR codes
Dissatisfaction with theory-heavy syllabi	Students want more hands-on, applied practice	Game includes concept-driven, scenario-based Multiple-choice Questions (MCQs) and code snippets simulating practical programming tasks in a time-efficient format
Favorable views of interactive environments	Students are open to tech-enhanced, gamified learning	Digital integration for scoring, feedback, and hint systems using devices students already own

As shown in Table 9, the key findings from the needs analysis directly informed the game design features proposed for the hybrid board game. Each challenge or preference identified among students was translated into a corresponding gameplay element aimed at improving learning outcomes. This alignment ensures that the game remains pedagogically grounded while addressing real student needs, laying a strong foundation for the next phase of game development.

V. Proposed Conceptual Framework

This study developed a conceptual framework to guide the design of a hybrid board game aimed at enhancing programming education for introductory-level ICT students in higher education. The framework is grounded in the findings of the needs analysis, which identified common challenges students face when learning programming, preferred learning environments, favored instructional materials, and the positive perception of interactive learning environments.

Based on the needs analysis, four major challenges were prioritized for instructional intervention: 1) Difficulties with debugging, logic structures, and syntax, 2) Strong preference for collaborative and practical learning situations, 3) Need for accessible, example-based materials and interactive visualizations, and 4) Favorable attitudes toward interactive and engaging learning environments.

The conceptual framework proposes aligning these identified needs with key design features informed by Plass *et al.* [45], which promote multiple forms of learner engagement, which is cognitive, physical, affective, and sociocultural engagement. This alignment is intended to foster improved programming competencies, increased learner motivation, enhanced collaboration, and better retention of knowledge.

The overall framework in Fig. 2 visually represents the relationships between students' learning needs, the targeted engagement strategies, and the expected educational outcomes. This framework serves as a foundation for the next stage of the research, guiding the design and development of the hybrid board game and aligning game elements with pedagogical objectives.

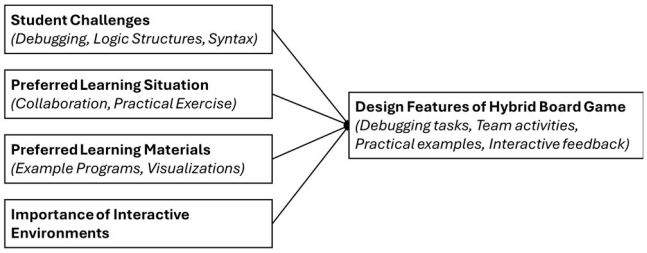

Fig. 2. Conceptual framework for designing a hybrid board game for programming education.

As shown in Fig. 2, the conceptual framework illustrates how identified student needs are translated into specific design features for the hybrid board game. Student challenges, such as difficulties with debugging, logic structures, and syntax, inform the inclusion of targeted game tasks. Preferred learning situations, including collaborative and hands-on activities, guide the incorporation of team-based gameplay and practical problem-solving exercises. Likewise, students' preference for example programs and visualizations informs the inclusion of tangible cards or digital elements that illustrate coding concepts. Finally, the strong support for interactive learning environments supports the integration of real-time feedback features. Collectively, these design strategies aim to foster increased motivation, deeper engagement, and improved understanding of Python programming concepts.

The following subsection explains how Cognitive Load Theory and Constructivist Learning Theory underpin these design choices.

A. Theoretical Integration into Game Design

Building on the four priority challenges identified earlier, the hybrid board game design is shaped by Cognitive Load Theory (CLT) and Constructivist Learning Theory.

1) Cognitive load theory

To minimize extraneous load, the interface includes color-coded rule cards and succinct instructions [46]. Intrinsic load is managed through the sequencing of tasks, beginning with basic concepts like variable manipulation and gradually advancing to more complex topics such as debugging and file handling. To enhance germane load, QR-linked hints and "worked-example" tiles provide immediate feedback, encouraging reflection and the formation of problem-solving schemas.

2) Constructivist learning theory

The game embraces core constructivist principles: 1) active learning through physical token manipulation and hands-on engagement, 2) social interaction via team-based challenges and peer explanation tasks, aligned with Vygotsky's emphasis on the social dimension of learning [40], and 3) authentic context through scenario cards that reflect real-world programming problems. These features support the co-construction of knowledge and help learners bridge abstract concepts with meaningful experiences.

By weaving these theoretical principles into tangible game mechanics, the proposed hybrid game is both pedagogically grounded and aligned with student needs, maximizing its potential as an engaging and effective formative assessment tool for introductory programming education.

VI. Conclusion

This study examined the learning needs of introductory-level ICT students in programming, focusing on their understanding of Python concepts, difficulties encountered, and preferred learning approaches. The findings highlight persistent challenges with advanced topics, debugging, and program design, while also revealing students' strong preference for collaborative learning, example-based resources, interactive visualizations, and the use of interactive environments. These insights underscore the importance of designing dynamic and engaging tools that accommodate diverse learning styles and support more effective programming education.

Building on these results, the study proposes the development of a hybrid board game as a formative assessment tool. The game will integrate physical components such as syntax cards and tokens with digital features including QR-coded puzzles and real-time feedback, offering interactive and collaborative learning experiences. While the current study was conducted in a single Malaysian higher education context, its findings provide a strong foundation for future research. Planned work will involve the detailed design, development, and evaluation of the game, with potential extensions to other programming languages, educational levels, and contexts. By aligning theory-driven design principles with empirically identified student needs, this study contributes to the advancement of gamified, multimodal approaches that hold promise for enhancing programming comprehension, motivation, and long-term retention.

Appendix

Survey Questionnaire Items

The following items were used in the survey instrument to identify the needs of diploma-level ICT students in learning Python programming. The questionnaire consisted of seven sections.

Section 1: Demographic Background

1. What is your age?
2. What type(s) of device(s) do you own? (Select all that apply)
   a. Mobile phone
   b. Laptop
   c. Tablet
   d. Desktop computer

Section 2: Level of Understanding of Python Programming Topics

Scale: 1 = Never Use, 2 = Weak, 3 = Moderate, 4 = Strong

Please indicate your level of understanding for each of the following topics:

1. Variables and Data Types
2. Input/Output Handling
3. Logic Structures: Decision (if, else, elif)
4. Logic Structures: Iteration (loops)
5. Logic Expressions (and, or, not)
6. String Manipulation
7. Math Operations
8. Functions
9. Lists
10. Tuples
11. Sets
12. Dictionaries
13. Lambda Functions
14. File Handling
15. Error/Exception Handling

Section 3: Difficulties While Learning Programming

Scale: 1 = Strongly Disagree, 5 = Strongly Agree

1. I have difficulty designing a program to solve a certain task.
2. I find it hard to divide functionality into procedures.
3. Learning the syntax of the programming language is challenging.
4. I struggle to find bugs in my own programs.
5. I have difficulty understanding basic programming structures.
6. I am not comfortable using programming development environments.
7. I face problems accessing computers or networks needed for programming tasks.

Section 4: Situations That Would Help Learn Programming More Effectively

Scale: 1 = Strongly Disagree, 5 = Strongly Agree

1. Learning during lectures, practicals, or lab sessions
2. Consulting or discussing with lecturers, tutors, seniors, or friends
3. Participating in small group exercises
4. Working alone on programming coursework

Section 5: Materials That Would Help Learn Programming

Scale: 1 = Strongly Disagree, 5 = Strongly Agree

1. Example programs
2. Exercise questions and answers
3. Interactive visualizations
4. Programming course books
5. Still pictures of programming structures
6. Lecture notes
7. Online discussion forums

Section 6: Factors That Lead to Poor Performance in Programming Subjects

Scale: 1 = Strongly Disagree, 5 = Strongly Agree

1. Few examples of practical use are shown
2. The syllabus focuses too much on theory
3. The syllabus covers too many topics in one semester
4. I lack interest in learning programming
5. The learning environment is not conducive
6. Instructor presentation and engagement are lacking
7. Teaching methods are not effective
8. Lab computers are not functioning properly

Section 7: Agreement with Using Interactive Environments for Learning Programming

Scale: 1 = Strongly Disagree, 5 = Strongly Agree

1. I agree that using an interactive environment would help me learn programming better.

Informed Consent Statement

Informed consent was obtained from all participants involved in the study.

Availability of Data

The original contributions presented in the study are included in the article; further inquiries can be directed to the corresponding author.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

M.F.C.A.R: Conceptualization, Data Curation, Formal analysis, Investigation, Methodology, Writing—Original Draft, Writing—Review & Editing. N.H.A, A.Z.M, M.M.Y: Methodology, Supervision, Validation. I.E.M, T.C.I: Formal Analysis. All authors have read and agreed to the published version of the manuscript.

References

[1] J. Figueiredo and F. Garcia-Penalvo, "Teaching and learning tools for introductory programming in university courses," in *Proc. SIIE 2021—2021 International Symposium on Computers in Education*, 2021. doi: 10.1109/SIIE53363.2021.9583623

[2] E. Kaila, M. Luukkainen, A. Laaksonen, and K. Lemström, "On changing the curriculum programming language from Java to Python (discussion paper)," in *Proc. 23rd Koli Calling Int. Conf. Computing Education Research*, New York, NY, USA: ACM, Nov. 2023, pp. 1–7. doi: 10.1145/3631802.3631814

[3] H. C. Ling, K. L. Hsiao, and W. C. Hsu, "Can students' computer programming learning motivation and effectiveness be enhanced by learning Python language? A multi-group analysis," *Front. Psychol*., vol. 11, 2021. doi: 10.3389/fpsyg.2020.600814

[4] A. Henley, J. Ball, B. Klein, A. Rutter, and D. Lee, "An inquisitive code editor for addressing novice programmers' misconceptions of program behavior," in *Proc. 2021 IEEE/ACM 43rd Int. Conf. Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, IEEE, May 2021, pp. 165–170. doi: 10.1109/ICSE-SEET52601.2021.00026

[5] Q. A. Batiha, N. A. A. Majid, N. Sahari, and N. M. Ali, "Analysis of the learning object-oriented programming factors," *Int. J. Electr. Comput. Eng.*, vol. 13, no. 5, 2023. doi: 10.11591/ijece.v13i5.pp5599-5606

[6] E. Mehmood, A. Abid, M. S. Farooq, and N. A. Nawaz, "Curriculum, teaching and learning, and assessments for introductory programming course," *IEEE Access*, vol. 8, pp. 125678–125689, 2020. doi: 10.1109/ACCESS.2020.3008321

[7] Sukirman, D. A. Pramudita, A. Afiyanto, and Utaminingsih, "Block-based visual programming as a tool for learning the concepts of programming for novices," *Int. J. Inf. Educ. Technol*., vol. 12, no. 5, pp. 365–371, 2022. doi: 10.18178/ijiet.2022.12.5.1628

[8] J. N. Da Silva Júnior *et al*., "A hybrid board game to engage students in reviewing organic acids and bases concepts," *J. Chem. Educ.*, vol. 97, no. 10, pp. 3657–3664, 2020. doi: 10.1021/acs.jchemed.0c00614

[9] A. M. Onencan, "Assessment of hybrid board game-based learning outcomes using the beatty theoretical framework," *Lecture Notes in Computer Science*, 2018. doi: 10.1007/978-3-319-91902-7_16

[10] C. Chukusol, P. Nilsook, and P. Wannapiroon, "Challenge-based hybrid learning model using virtual board games platforms," *Int. Educ. Stud.*, vol. 17, no. 3, pp. 39–49, May 2024. doi: 10.5539/ies.v17n3p39

[11] F. Napalit, B. Tanyag, C. L. So, C. Sy, and J. R. San Pedro, "Examining student experiences: Challenges and perception in computer programming," *Int. J. Res. Stud. Educ.*, vol. 12, no. 8, Nov. 2023. doi: 10.5861/ijrse.2023.71

[12] R. Hijón-Neira, C. Connolly, D. Palacios-Alonso, and O. Borrás-Gené, "A guided scratch visual execution environment to introduce programming concepts to cs1 students," *Information (Switzerland)*, vol. 12, no. 9, pp. 1–15, 2021. doi: 10.3390/info12090378

[13] W. C. Choi and I. C. Choi, "The influence of CodeCombat on computational thinking in Python programming learning at primary school," in *Proc. 5th Int. Conf. Educ. Development and Studies (ICEDS)*, New York, NY, USA: ACM, Apr. 2024, pp. 26–32. doi: 10.1145/3669947.3669951

[14] S. C. Chang and C. Wongwatkit, "Effects of a peer assessment-based scrum project learning system on computer programming's learning motivation, collaboration, communication, critical thinking, and cognitive load," *Educ. Inf. Technol.*, 2024, vol. 29, no. 6, pp. 8123–8141. doi: 10.1007/s10639-023-12084-x.

[15] C. M. Kandemir, F. Kalelioğlu, and Y. Gülbahar, "Pedagogy of teaching introductory text-based programming in terms of computational thinking concepts and practices," *Comput. Appl. Eng. Educ*., vol. 29, no. 1, pp. 114–126, 2021. doi: 10.1002/cae.22374

[16] S. Adipat, K. Laksana, K. Busayanon, A. Ausawasowan, and B. Adipat, "Engaging students in the learning process with game-based learning: The fundamental concepts," *Int. J. Technol. Educ.*, vol. 4, no. 3, pp. 71–83, 2021. doi: 10.46328/ijte.169

[17] V. Kankainen and J. Paavilainen, "Hybrid board game design guidelines," in *Proc. 12th Digital Games Research Association Int. Conf. (DiGRA)*, 2019. doi: 10.26503/dl.v2019i1.1098

[18] J. N. Silva *et al*., "HSG400—Design, implementation, and evaluation of a hybrid board game for aiding chemistry and chemical engineering students in the review of stereochemistry during and after the COVID-19 pandemic," *Educ. Chem. Eng*., vol. 36, pp. 36–44, 2021. doi: 10.1016/j.ece.2021.04.004

[19] J. N. S. Júnior *et al*., "Reactions: An innovative and fun hybrid game to engage the students reviewing organic reactions in the classroom," *J. Chem. Educ.*, vol. 97, no. 3, pp. 749–753, Mar. 2020. doi: 10.1021/acs.jchemed.9b01020

[20] F. Pozzi, A. Ceregini, S. Ivanov, M. Passarelli, D. Persico, and E. Volta, "Digital vs. hybrid: Comparing two versions of a board game for teacher training," *Educ. Sci*., vol. 14, no. 3, pp. 1–15, 2024. doi: 10.3390/educsci14030318

[21] M. Videnovik, T. Vold, L. Kiønig, A. M. Bogdanova, and V. Trajkovik, "Game-based learning in computer science education: a scoping literature review," *Comput. Sci. Educ*., 2023. doi: 10.1186/s40594-023-00447-2

[22] J. H. Berssanette and A. C. D. Francisco, "Cognitive load theory in the context of teaching and learning computer programming: A systematic literature review," *IEEE Trans. Educ*., 2022. doi: 10.1109/TE.2021.3127215

[23] Y. C. Liu, W. T. Wang, and W. H. Huang, "The effects of game quality and cognitive loads on students' learning performance in mobile game-based learning contexts: The case of system analysis education," *Educ. Inf. Technol*., vol. 28, no. 12, pp. 15873–15895, 2023. doi: 10.1007/s10639-023-11856-9

[24] R. C. Richey and J. D. Klein, *Design and Development Research*, New York, NY, USA: Routledge, 2014. doi: 10.4324/9780203826034

[25] N. A. Alias and S. Hashim, "Design and development research in instructional technology," in *Proc. Instructional Technology Research, Design and Development: Lessons from the Field*, Hershey, PA, USA: IGI Global, 2012, pp. 1–16. doi: 10.4018/978-1-61350-198-6.ch001

[26] W. G. Cochran, *Sampling Techniques*, 3rd ed., New York, NY, USA: John Wiley & Sons, 1977. doi: 10.2307/2573260

[27] R. V. Krejcie and D. W. Morgan, "Determining sample size for research activities," *Educ. Psychol. Meas*., vol. 30, no. 3, pp. 607–610, 1970. doi: 10.1177/001316447003000308

[28] P.-H. Tan, C.-Y. Ting, and S.-W. Ling, "Learning difficulties in programming courses: undergraduates' perspective and perception," in *Proc. Int. Conf. Computer Technology and Development (ICCTD)*, 2009, pp. 42–46. doi: 10.1109/ICCTD.2009.188

[29] A. DeCastellarnau, "A classification of response scale characteristics that affect data quality: A literature review," *Qual. Quant*., vol. 52, no. 4, pp. 1523–1549, 2018. doi: 10.1007/s11135-017-0533-4

[30] R. Heale and A. Twycross, "Validity and reliability in quantitative studies," *Evid. Based Nurs*., vol. 18, no. 3, pp. 66–67, Jul. 2015. doi: 10.1136/eb-2015-102129

[31] J. D. Brown, "The Cronbach alpha reliability estimate," *JALT Testing & Evaluation SIG Newsletter*, vol. 6, no. 1, pp. 17–20, 2002.

[32] F. J. Fowler, *Survey Research Methods*, 5th ed., Thousand Oaks, CA, USA: SAGE Publications, 2014.

[33] M. Tavakol and R. Dennick, "Making sense of Cronbach's alpha," *Int. J. Med. Educ.*, vol. 2, pp. 53–55, 2011. doi: 10.5116/ijme.4dfb.8dfd

[34] G. M. Sullivan and A. R. Artino, "Analyzing and interpreting data from Likert-type scales," *J. Grad. Med. Educ*., vol. 5, no. 4, pp. 541–542, Dec. 2013. doi: 10.4300/jgme-5-4-18

[35] J. W. Creswell and J. D. Creswell, *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*, 5th ed., Thousand Oaks, CA, USA: SAGE, 2018.

[36] A. Field, *Discovering Statistics Using IBM SPSS Statistics*, 5th ed., Thousand Oaks, CA, USA: SAGE, 2013.

[37] B. G. Tabachnick, L. S. Fidell, and J. B. Ullman, *Using Multivariate Statistics*, 7th ed., Boston, MA, USA: Pearson, 2018.

[38] Malaysian Qualifications Agency. (2023). Programme standard: Computing. *Cyberjaya*. [Online]. Available: https://www2.mqa.gov.my/qad/v2/2023/PS%20Computing%203rd%20Edition%20-%2031.5.23.pdf

[39] J. Piaget, *The Construction of Reality in the Child*, New York, NY, USA: Basic Books, 1954. doi: 10.1037/11168-000

[40] L. S. Vygotsky, *Mind and Society: The Development of Higher Psychological Processes*, Cambridge, MA, USA: Harvard Univ. Press, 1978. doi: 10.2307/j.ctvjf9vz4

[41] A. Rovshenov and F. Sarsar, "Research trends in programming education: A systematic review of the articles published between 2012–2020," *J. Educ. Technol. Online Learn*., vol. 6, no. 1, pp. 1–15, 2023. doi: 10.31681/jetol.1201010

[42] T. Daradoumis, J. M. M. Puig, M. Arguedas, and L. C. Liñan, "Enhancing students' beliefs regarding programming self-efficacy and intrinsic value of an online distributed programming environment," *J. Comput. Higher Educ*., vol. 34, no. 3, pp. 555–574, 2022. doi: 10.1007/s12528-022-09310-9

[43] S. Choi and H. Kim, "The impact of a large language model-based programming learning environment on students' motivation and programming ability," *Educ. Inf. Technol*., Nov. 2024. doi: 10.1007/s10639-024-13107-x

[44] N. A. Anindyaputri, R. A. Yuana, and P. Hatta, "Enhancing students' ability in learning process of programming language using adaptive learning systems: A literature review," *Eng. Sci. Technol. Int. J*., 2020. doi: 10.1515/eng-2020-0092

[45] J. L. Plass, B. D. Homer, and C. K. Kinzer, "Foundations of game-based learning," *Educ. Psychol*., vol. 50, no. 4, pp. 258–283, Oct. 2015. doi: 10.1080/00461520.2015.1122533

[46] J. Sweller, J. J. G. van Merriënboer, and F. Paas, "Cognitive architecture and instructional design: 20 years later," *Educ. Psychol. Rev.*, vol. 31, pp. 261–292, 2019. doi: 10.1007/s10648-019-09465-5