I.J. Modern Education and Computer Science, **2025, 1, 1-16**
Published Online on February 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2025.01.01

Explicit Instruction-based Methodology for Teaching Introductory Computer Programming

Alain Kabo Mbiada*
Department of Computer Science, North-West University, Mafikeng, South Africa
Email: mbiadaalain@gmail.com
ORCID iD: https://orcid.org/0000-0003-4614-1370
*Corresponding Author

Bassey Isong
Department of Computer Science, North-West University, Mafikeng, South Africa
Email: bassey.isong@nwu.ac.za
ORCID iD: https://orcid.org/0000-0002-3915-4627

Received: 29 May, 2024; Revised: 26 June, 2024; Accepted: 15 August, 2024; Published: 08 February, 2025

Abstract: Non-computing students often encounter greater challenges in programming courses compared to their computing counterparts, primarily stemming from a lack of motivation in the subject. Motivation plays a pivotal role in the success of introductory programming (IP) modules, with intrinsically and extrinsically motivated students exhibiting greater enjoyment and engagement in learning activities. While numerous studies have attempted to enhance motivation in IP modules, most have focused on computing students which is influenced lar gely by the constructivist theory. This paper addresses this gap by proposing a cognitive-based teaching framework aimed at bolstering motivation among non-computing students. The proposed approach employs the Explicit Instruction paradigm, where the instructor first designs learning strategies and provides students with detailed explanations, demonstrations, examples, and non-examples. This enables the students to apply the strategies in groups, practice with feedback, and finally individually. The effectiveness of this approach was assessed using first-year students at two universities, one in South Africa and the other in Cameroon. We collected student motivation data using a quantitative questionnaire post-experiment. The results indicate that the proposed teaching method had a positive impact on participant motivation in terms of attendance, perceived relevance, confidence, and satisfaction. However, the specific degree of improvement varied among the participants.

Index Terms: Computer Programming, Teaching Methodology, Explicit Instruction, Student Motivation, Student Engagement

1. Introduction

Computer programming is increasingly becoming an essential skill for non-computer science students, as knowledge of information and communication technologies has become one of the most sought-after skills in real-life job openings. Programming is a complex and creative thought activity that requires both knowledge and strategies for achieving success. The diversity of student backgrounds enrolled in programming modules today in most developed nations makes teaching and learning difficult. Earlier studies demonstrated that non-computing students experience the most difficulties, to a greater degree than computing students [1]. Organizing and allocating time and effort to learn programming is one of the challenges. Other difficulties include not understanding basic programming concepts, not being able to use those concepts in a specific order to write a working program, not being able to debug programs that generate errors after compilation, and not being able to analyse and predict a program's behaviour, etc. Several factors may help to explain these difficulties, including a lack of self-confidence, inattention during lessons, learning less during lecture sessions, and so on [1]. Most of these factors result from a lack of motivation in the subject, which is why this same study advised the development of a cognition-based teaching method to improve the motivation, engagement, and outcomes of non-computing students in introductory programming (IP) modules. This is in line with the study by Hong et al. [2] who observed that students, who have a high level of intrinsic motivation consistently perform better in the class. Additionally, motivation plays a pivotal role in the success of IP modules, with intrinsically and extrinsically motivated students exhibiting greater enjoyment and engagement in learning activities [3,4]. Hence, students' motivation in IP modules is an important component that might improve their learning outcomes [3]. On the other hand,

learning programming demands sufficient cognitive abilities, which are often enhanced when students are intrinsically motivated. Therefore, to be effective and successful, the instructional approach must cover non-computing demands and provide a motivating framework [5]. Previous studies [6,12] have also shown that insufficient or a lack of motivation and commitment are, among other factors, at the root of major failures and increased drop-out percentages in computer programming courses. To improve this situation, the bulk of interventions based on constructivism were directed more at computer science (CS) students than non-CS students. On the other hand, the choice of cognitive-based technique may be justified by the fact that most non-computing students have no prior programming experience, an insufficient level of mathematics, inadequate programming strategies, and a lack of in-depth, well-organized knowledge needed to implement effective problem-solving processes. Moreover, most of them, regardless of their young age, need a lot of guidance during instruction, whereas experts can work independently. Then, non-computing students should receive direct pedagogical guidance on core concepts and strategies, and not be left to find out about them on their own [13]. In addition, designing a new instructional approach for teaching non-computing students IP modules is a current and ongoing concern in the CSE field [14]. Therefore, this study suggests an explicit instruction (EI) approach for teaching IP to novices as non-computing students. The proposed teaching approach is based on cognitivist theory, and its usefulness has been proven in reading native and foreign languages, mathematics, sciences, and history [15]**.** In this teaching framework, the teacher first models learning strategies and provides students with detailed explanations, demonstrations, examples, and non-examples. Afterwards, students apply the strategies in groups, on sustained practices with feedback, and finally individually. Explicit Instruction (EI) promotes a direct approach to practice, eliminating ambiguity and complexity from learning situations. It is an effective way to encourage understanding, participation, and inquisitiveness in students instead of being a teacher-centered monologue [16]. EI has the potential to increase non-computing students' motivation and is innovative in teaching IP modules for several reasons [17]. 1). EI provides clarity and understanding due to its usefulness in explaining challenging subjects by attracting students' attention to essential areas of the course material. The instructional material provides unambiguous guidance, enabling students to fully comprehend the content at the end of the class. 2). EI promotes active student engagement, unlike passive learning in direct instruction, as students can ask questions, request more explanations, and actively interact with peers and the instructor. 3). EI increases curiosity among students as they are encouraged to question the reasons and mechanisms behind a topic, engaging in deeper exploration rather than just tackling it superficially. This element is critical in IP modules for students to be able to later analyze or design a proper program that effectively solves problems. 4). EI can be tailored by a teacher to accommodate unique learning styles and paces, ensuring that no student lags. 5). EI involves structured oversight while also fostering critical thinking, problem-solving, and creativity [17]. The proposed teaching framework was tested using two case studies. Firstly, in a CMPG121 module with non-CS first-year students from the Faculty of Natural and Agricultural Sciences (FNAS) at the North-West, Mafikeng, South Africa (Group 1). Secondly, in an introduction to Python course, the groundwork for module INFO2122, Object-Oriented Programming with Python, was laid with first-year trainee teachers from the HTTC, University of Yaoundé 1, Cameroon (Group 2). The objective was to find out how the proposed teaching framework impacts student motivation. Based on this purpose, the following research questions (RQs) were formulated: 1). *How can we design a novel teaching framework that can engage and motivate learners in an IP module? 2). How effective is the designed teaching framework from the learner's motivation perspective based on the four components of the ARCS model* [18] *which are attention, relevance, confidence and, satisfaction?* These RQs drive the design, implementation, and evaluation of the proposed EI framework's ability to increase the motivation of non-computer students. We rigorously analyzed and presented the findings. Therefore, we can characterize the significance of this paper as follows:

- Designed a new teaching EI-based framework for teaching IP modules to non-computing students.
- Conducted a cross-cultural study to assess the impact of the proposed EI framework on the student's motivation.
- Provided a discussion on the effectiveness and implications (academic and application) of the proposed teaching methods and the comparison with other existing teaching methods.

The remaining parts of the paper are structured as follows: Section II presents the background information and some of the related works, Section III presents the proposed cognitive teaching approach; Section IV presents the experimentation; Section V presents the survey on students' motivation; Section VI analyses the impact and compares the new teaching method with other existing approaches; and Section VII concludes the paper.

2. Literature Review

2.1 Background Information

A. Motivation and Engagement

Motivation and engagement are important success factors in IP courses [4], and how to motivate and engage students is a major concern in all disciplines. A success factor represents certain skills, attitudes, or beliefs that, once a student has them, lead to the conclusion that he or she will have a good outcome in learning activities [19]. It is

important for a teaching strategy to incorporate or include at least one success factor. Students are motivated to learn or gain new skills when their potential is well understood or when gaining these skills will lead to a good grade and the privileges that a good grade confers [20]. To support motivation, teaching methods need to integrate techniques that deal with the attention, relevance, confidence, and satisfaction (ARCS) model [21]. The ARCS model is therefore an excellent way of measuring students' motivation. The ARCS model was developed by John Keller [18], as a result of a concern to provide more accurate approaches to the key influences on motivation to learn, as well as systematic means of detecting and addressing problems associated with motivation to learn.

Furthermore, engagement is viewed as the ability to invest time and effort throughout the training program [22] and involves the initiation of action and active participation. It is defined as " *a positive, fulfilling, work-related state of mind that is characterized by vigour, dedication, and absorption*…" [23]. Vigor is described as high energy levels and mental toughness during work, a desire to work hard, and perseverance even in case of difficulties. Dedication is about being deeply immersed in one's task and feeling a sense of significance, enjoyment, pride, and challenge. Absorption is being completely focused and joyfully absorbed in one's task, so time flies by and it is hard to detach oneself from one's task. To support students' engagement in learning, the teaching approach must include opportunities for control over the activities offered to develop a positive perception of competence; including tasks that correspond to the students' level of ability; provide self-assessment grids to allow students to assess their performance and thus foster the development of their autonomy. Schaufeli et al. [23] designed an assessment questionnaire known as the Utrecht Work Engagement Scale (UWES) for the measurement of engagement.

On the other hand, in teaching and learning IP modules, in addition to motivation and engagement, some studies pointed out other success factors such as grades from secondary school, prior programming experience, gender, mathematics or English background, learning style, self-efficacy, mental model, comfort level, achievement goals, teaching tools and methods, etc. [24,26]. Important to notice is that each teaching strategy should support or develop at least one of these factors in learners.

B. Misconception and Concept Inventories in IP

Misconceptions and concept inventories (CI) are critical to providing the right interventions to improve the learning process. Misconceptions are academic concepts that are challenging and well-rooted [27]. Learners most often build huge misconceptions when it comes to programming. In learning programming, students' misconceptions involve syntax errors, improper semantic comprehension, and other challenges in the attempt to write correct programs [28]. Moreover, the age of the student, the programming language, the context of programming, the challenges of switching from a native language to a programming language, a textbook that includes wrong information and mistakes, previous learning experiences, and insufficient insight into the topic by the teacher are some factors that can develop or prevent students' misconceptions [29,30]. Previous studies developed some misconceptions [30,33] that could help teachers develop their understanding (or lists or data banks) of their students' misconceptions. For example, Altadmri and Brown [34] in their study identified the 18 most common mistakes made by a wide range of beginners worldwide in the Java programming language and classified them into three main types: misunderstanding or forgetting the syntax, type errors, and semantic errors. In addition, Johnson et al. [31] identified seven first-year students' misconceptions of Python programming languages. Developing misconceptions is an important skill for teachers. To this end, this study strongly recommends that teaching strategies for IP courses that include students' misconceptions of the content, is an important measure to reduce students' frustration, failure, and dropout rates. It might also be helpful in designing an effective learning and programming environment.

On the other hand, CI is a specialized multiple-choice test designed to assess students' understanding of the basic notions of a subject [35]. In this case, teachers and researchers are in accord that if students have a good understanding of the basic notions, they should certainly understand all of the rest of the concepts in the subject [36]. The CIs are not instructor assessments but are supplementary to the final exam. They reveal students' misconceptions and help the instructor identify struggling students' fundamental concepts and build his instruction accordingly. The multi-choice item is a test composed of distractors and correct choice options, as shown in Fig. 1 [37]. Distractors reflect students' misconceptions and come from analysis of results from exercises during lectures tutorial classes, or exams [37,38]. Thus, we recommend the following process, designed by Taylor et al. [35], for developing and validating reliable CIs: a) Identify core concepts; b) Identify student thinking; c) Create multi-choice survey questions; d) Create forced-answer tests; e) Validate test questions through interviews; f) Administer and conduct statistical analysis.

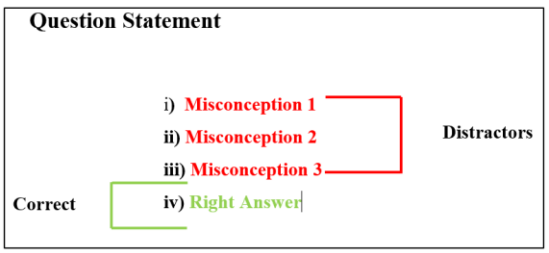

Fig. 1. Predicted anatomy of a concept inventory

C. Learning Theory

The literature on pedagogy offers many theories and models that advocate or encourage teaching and learning practices, such as behaviourism, cognitivism, and constructivism. Behaviourism is a theory of learning based on the definition of the knowledge that needs to be acquired in terms of observable behaviours at the end of the learning process [39]. Behaviourists believe that mental structures are like a black box to which we have no access and that it is, therefore, more realistic and effective to focus on inputs and outputs than on the processes themselves. Behaviorism was reproached for only sustaining lower-level thinking processes. But these processes are essential in most fields, for example, coding demands repeated, and in-depth practice in the use of various programming concepts. Cognitivism is a theory of learning that focuses more on mental processes and is therefore much more concerned with how we learn and how our minds absorb, store, process, and access information [40].

The key principles of cognitive learning theory are as follows [39]: Learning is a process of organizing information into conceptualized models; Instructions must be organized, sequenced, and, presented in a way that is understandable and meaningful to the learner; Retention and recall are important for building patterns in the brain; The organization of learning materials supports memory; Teachers need to provide tools that help the learner's brain process information. It is a better vision than behaviourism, as it focuses on the mental process of knowledge acquisition by the learner. Constructivism is a theory of learning that holds that students learn best when they actively construct their knowledge [39]. It is based on the assumption that students are active learners, capable of developing their understanding of a subject and applying it to new situations. Thus, the teacher's role is to facilitate learning by helping students learn how to learn.

D. Explicit Instruction Method

Explicit instruction (EI) is a teacher-centred method, unlike less structured models, which are student-centred and based on the pedagogy of discovery. This method emanates from cognitivism theory, which focuses on the development of intelligence and cognitive processes such as perception, memory, and thinking, as well as the resulting behaviours [40]. Rosenshine defined EI as "*…a systematic method of teaching with an emphasis on proceeding in small steps, checking for student understanding, and achieving active and successful participation by all students*" [16]. It is called explicit due to its actions for designing and implementing instruction. With this method, teachers intentionally use a variety of strategies to support students' learning. They guide students throughout the learning process with detailed explanations, demonstrations, and supported practices with feedback [41]. Its success is due to students' deployed efforts and the teaching strategies used by teachers. On the other hand, a strategy can be defined as a complex cognitive or meta-cognitive operation that allows one to reach a given goal by employing a series of actions carried out consistently or inconsistently [42]. The strategy will serve to carry out cognitive processes such as comprehension and memorization. It is an effective instructional approach to promote the success of the greatest number and is recommended for struggling, slow-paced, weak, and good students as well [15,43]. EI focuses on teaching critical content, and we assume that the implementation of EI in IP courses can help overcome most of the difficulties encountered by novices as non-computing students to enhance their motivation, mental model construct, commitment, learning strategies, outcomes, etc. That is, offer clear and direct teaching techniques, providing explicit and straightforward guidance to novice learners. However, implementing EI in a teaching and learning situation can be difficult. Therefore, for quality explicit teaching, we recommend the use of the ten guidelines suggested by Ben Newmark, detailed here [44]. These principles do not constitute a definitive set of rules, but rather a contribution of advice to teachers on how to achieve high-quality explicit teaching. Its key features include clear learning objectives, teacher modelling, guide practice, gradual release of responsibility, structured feedback, repetition and review, monitoring progress, and versatile application [16,40].

2.2 Related Works

This section discusses the teaching strategies developed over the years for IP courses to improve students' motivation and engagement. For example, Horváth and Javorský [7] experienced the use of a graphical robot method in a Java programming course and found that it is an effective way to promote students' motivation in/out-class activities. They argued that motivation stems from the fact that this teaching approach enables students to immediately see the result of their efforts. The visualization of their results is a source of motivation for students. These students were future informatics high school teachers, mostly with no prior programming experience. Analysis of the results of the qualitative survey (interviews) revealed that lack of time, the mismatch between the teacher's explanations and the student's comprehension, motivation, and the challenge of forming a group were the most common reasons for the difficulties students encountered in IP courses.

In the same vein, Facey-Shaw et al. [11] using a mixed-methods survey design, evaluated the effect of Gamification with the use of Digital Badges on students' intrinsic motivation in IP courses. The intrinsic Motivation Inventory instrument was used to assess students' intrinsic motivation. The quantitative results showed that digital badges do not increase students' intrinsic motivation while the qualitative results revealed the positive impact of digital badges on students' intrinsic motivation. Moreover, Jiau et al. [8] reported on a learning environment for simulated programming that aims to support the implementation of game-based simulations and metrics and to improve students' self-motivation. Through this environment, students learn basic programming skills by programming a game strategy

that can win against the computer. After the game, the metrics indicate the effectiveness of the strategy and provide the student with clues (feedback) to refine the strategy that could further enhance the result of the game. Other research studies [45,46] have also pointed out gamification as a good strategy to enhance students' motivation and engagement.

In addition, Souza and Bittencourt [9] analyzed, through a qualitative survey design method, the impact of using problem-based learning (PBL) to improve students' motivation and engagement in an IP course. Students' motivation was measured using the four categories of the ARCS motivation model, while students' engagement was measured using the Utrecht Work Engagement Scale. The PBL approach is an active teaching method derived from constructivist theory that improves students' skills in problem-solving and critical thinking. The results showed that the PBL approach had a positive impact on students' motivation, especially in the categories of attention and relativity. On the other hand, it did not have an impact on students' engagement. Challenges encountered relate to the application of the method and the selection of problems. Thus, the authors suggested a good design of problems and activities for their efficiency in increasing students' engagement and motivation.

Santana et al. [10], on the other hand, evaluated the impact of the mixed-context teaching approach for IP modules on CS non-major motivation. This teaching approach combines the use of Scratch in the context of game creation, and Python with both turtle graphics and image manipulation. The results showed an increase in students' motivation in terms of all four categories of the ARCS motivation model. Besides, Hartanto et al. [47] work demonstrated that the holistic approach improves students' engagement in an IP course. The holistic approach includes a set of activities such as problem-based learning, learning by doing, building prior knowledge, learning by example, creating good student-teacher communication, collaborative work, and providing feedback. In the same vein, Pabón and Villegas [6] presented the partial result of the application of an integrated teaching approach in IP courses. This approach includes the following elements: Python as a first programming language, project-oriented and problem-based learning, multimedia resources, and assessment rubrics. This approach has been implemented by students at a university, in Colombia, and data was collected from 2011 to 2017. The output showed that this approach has a positive impact on students' engagement and performance as well.

On the other hand, Carbonaro [12] demonstrated the positive impact of peer review on student engagement, time management skills, and competencies in a traditional IP course. Peer assessment is a good chance for students to assess code, write and read comments, and see how fellow students are dealing with the same problem. Teachers can adopt this method only when they are sure that students can be trusted and its outcomes can be reliable.

Notwithstanding the pertinence of the above-mentioned strategies to engage and motivate students in IP modules, studies still report students' failure and dropout in IP modules [48,49]. Furthermore, most of these strategies have been heavily based on constructivist and socio-constructivist theories such as active learning, problem-based learning, peer review, etc. Unfortunately, with the constructivism theory, students will need more time to solve problems, and learn to solve them; they will commit more mistakes and feel much more frustrated. In the end, they will learn less. Thus, constructivism theory implies minimal or unguided supervision during instruction, which seems to be much less effective for novices [13]. In addition, the essentials of these strategies are aimed more at computing students than non-computing students. For non-computing students, an earlier study from [1] recommended the development of a cognitive teaching method. Therefore, the paper proposes a cognitive approach to teaching IP based on EI to support teaching and learning activities for non-computing students, which may improve their motivation, engagement, and performance.

3. Proposed EI-based Teaching Methodology

This section presents and discusses the proposed teaching methodology, which is a cognitive approach based on the EI of the cognitivist theory [40]. This section answers the first research question defined in this paper.

3.1 EI Method Process

The proposed EI method for IP modules is known as IPEI, which has iterative phases. Fig. 2 shows the phases including reviewing, model-based programming teaching, guided programming, non-guided programming, and report and transfer in our context. These are discussed as follows:

1) *Reviewing*: At this stage, the teacher will address Why… are students learning this. When… will it be relevant to them? And Where… will students use these skills? This stage also includes a review of homework and relevant previous learning, as well as a review of prerequisite skills and knowledge.
2) *Model-based programming teaching*: Modelling is a method of training self-management strategies for understanding, where the teacher acts as a guide, explaining his or her behaviour when faced with a comprehension problem. The teacher explains the source of his or her difficulty and the strategy used to overcome the problem. The teacher also models effective attention and performance strategies for students. Thus, this phase is more based on strategies deployed and steps followed to solve a programming task, which is one of the main focuses when preparing the class lesson. The teacher offers through demonstration an appropriate set of programming examples and non-examples to draw the line as to when to apply or not a skill, strategy, concept, or rule [16]. The use of non-examples, in this case, might help students reduce inappropriate

uses of skills, and frustration faced when starting programming. During this phase, the teacher should introduce, whenever possible, misconceptions, how he or she or programming experts proceed to understand programming, and how they overcome the difficulties they have faced as well. Examples will progressively move from daily life-based to mathematics-based problems to ease students' integration, increase students' motivation, and promote knowledge acquisition. During this phase, students must listen, not interrupt, and save questions until an appropriate time. On the other hand, the teacher is advised to involve students as much as possible through questioning.

3) *Guided programming*: In this phase, the teacher supports the acquisition of the strategy through explanation, teaching, and modelling. In addition, following the principle of scaffolding, the teacher encourages independent and autonomous use of the strategies introduced and developed during the modelling phase. Students better learn strategies when they experience the role of the teacher and have to justify the importance of using the strategies. These practical classes are assisted by the teacher and allow students to correct errors as they proceed through immediate feedback. This phase aims to verify students' comprehension and immediately correct their misconceptions. This in turn will promote student self-confidence, satisfaction, and collaborative programming. The activities are organized around the use of collaborative programming techniques described as follows:
   - The teacher gives a small programming problem to the students.
   - The students individually take a few minutes to think about the solutions.
   - Students are organized into groups of two or more, as they sit in the classroom, discuss and share their thoughts, and come up with a group solution. To improve students' logical programming thinking, the paper-and-pencil strategy (PPS) is required. The PPS is a handwriting technique that helps students solve problems as computers do and guides them to illustrate their solutions in a diagram, table, symbols, etc. [50]. On the other hand, during practical sessions, students sit in pairs in front of one computer and implement their elaborated solutions. While one types the code, the second observes, corrects, and improves the code quality. This situation has to be interchanged for the next activity or exercise.
   - The teacher approves or redirects the groups' proposals. This interaction will help teachers identify students' misconceptions which will later ease the preparation of concept inventories. Again, these misconceptions will also direct the elaboration of exercises that are closer to the student's zone of proximal development. These practices have to be continued until students master the strategies. But, if, after many attempts, the strategies prove futile, the teacher should go back to the modelling phase or move to the next phase.
4) *Non-guided programming:* This is an extension of the guiding practice phase, which is designed to promote memory retention and personal skill development by allowing students to develop their understanding to the highest possible level of learning mastery. These are programming activities that will allow students to work independently, ie without any support from other students and the teacher. Like the guided programming phase, these activities must be continued until students individually master the strategies. But if not mastered after several attempts, the teacher should go back to the guided programming or the modelling phase depending on his observation, or simply move to the next step otherwise.
5) *Report and transfer:* This is an evaluation stage where the teacher invites students to discuss what they have learned. This discussion enables students to check their understanding of the strategy and its relevance in a given context. This last element should enable students to discuss possible transfers of the strategy to other contexts or disciplines.

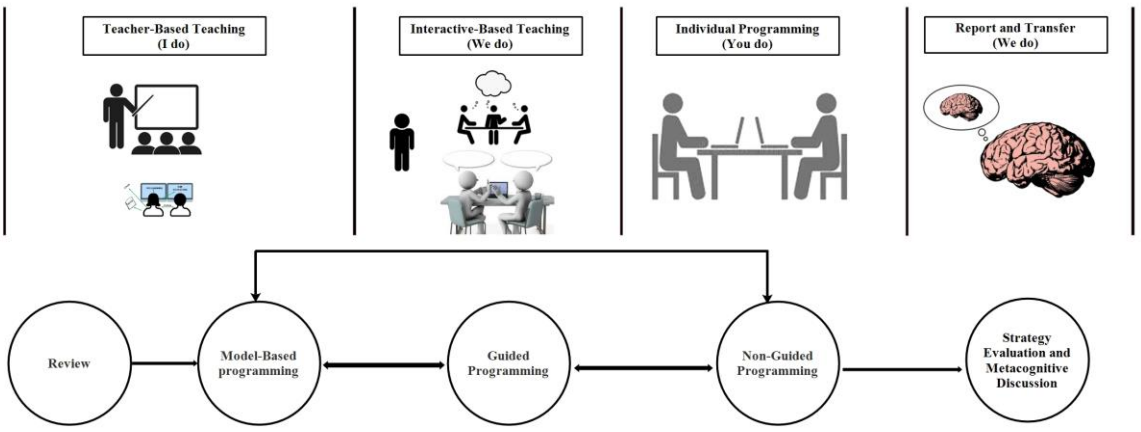

Fig. 2. IPEI framework

3.2 Discussion

As shown in Fig. 2, IPEI deals with the teaching of core programming concepts to novices based on strategy and the construction of a knowledge scheme. IPEI starts each lesson with a short review and introduces new features step by step. It then proceeds with guided activities followed by students working independently, and thereafter tests students' acquisitions frequently to know what they have retained, not just to assess them. Therefore, IPEI places the teacher at the centre, and its implementation requires the teacher to have quality preparation in terms of choice of concept, quality of explanation, research on students' misconceptions, the development of CIs, the choice of examples and non-examples, etc. Incorporating misconceptions and CI in the process of teaching IP would reduce students' frustration and discouragement, and later help the instructor have a good opinion of the student's understanding of basic programming concepts. It sequences skills logically, focuses instructions on essential content, and provides a wide range of examples and non-examples, primarily based on misconceptions of any kind. IPEI also allows the teacher, based on the results observed in one step, to return to the previous step for further explanation. The collaborative programming activity included in the IPEI stage is highly effective in promoting high-level cognitive skills in students.

Through this collaborative exercise, learners experience their limitations, inconsistencies, and beliefs. IPEI as an instructional method satisfies the nine teaching events described by GAGNE [51]: Increase retention and transfer; Assess performance; Provide feedback on performance; Highlight performance; Guide the learner; Present stimulus material; Stimulate recall of previous knowledge; Inform the learner of the objective; Gain attention. It bounds all the teacher's and learners' activities in the classes and can be applied in both lecture and practical classes, contrary to most other methods [25,52,55] which base their methods on the coding part. IPEI acknowledges that programming is about analysis, designing, coding, testing, and debugging. We assume that the implementation of this teaching method supported by Ben Newmark's principles [44] in any IP module would improve student motivation, engagement, and performance.

3.3 Evaluation of the EI framework

This section presents the experiment related to the application of the five steps of the IP-EI framework to the teaching of computer programming. The goal is to answer the second research question.

1) *Experimental requirements*: This explains how to run an experiment using the EI framework with first-year non-computing students in an IP class. Its goal is to help determine the viability of such an experiment. Thus, before conducting experiments, the instructor must adapt to the EI framework's phases. The instructor must engage in meticulous preparation, ensuring that they properly define the topic, goals, and anticipated outcomes for the students throughout the session. Preparing from the literature or any other manner students' misconceptions of the topic is also critical to the experiment's success. The experiment should cover the programming concepts required to carry out the various stages of the IE framework. To ensure the experiment's reliability and the students' effective presence, it will be carried out during regular class.
2) *Context*: The experiment with Group 1 involved first-year students as non-computing students enrolled in an elective module entitled CMPG121, structured programming. Fifty-eight (58) students were officially registered for this module. This module was delivered in the second semester of the 2023 academic year and lasts around 12 weeks, with a load of 66 hours. It is divided into lectures and laboratory sessions and takes place three times a week. The lecture lasts an hour and a half on the first two days (lecture session) and two and a half hours on the last day (practical session). This course is taught using the C++ programming language. On the other hand, the experiment with Group 2 involved first-year trainee teachers as non-computing students who are being trained to become high school teachers later on. This second experiment was applied in an introductory course to Python, which is a prerequisite for the module entitled INFO2122, Object-Oriented Programming with Python, taught in the second year. This course has about thirty (30) students and was delivered during the second semester of the 2023/2024 academic year. The course lasts two hours, with many hands-on exercises, and takes place twice a week.
3) *Intervention with Group 1*: The experiments were conducted during lectures and lab sessions with the participation of almost 45 students. At the beginning of each class, students were informed that the module would have to follow an experimental new teaching approach, the steps of which are then described, and their roles outlined. The reason for this is that students don't attend class regularly, so attendance lists are drawn up at the end of each lesson. The concept we worked on during the two weeks of experimentation was that of binary files. Before the experiment, we researched literature on students' misconceptions about binary files, which served to prepare non-examples. The experiment was easy to set up because the framework is defined by precise steps that are simple to follow. In the first day's implementation of the framework, which lasted an hour and a half, it was not possible to develop all five steps, as we realized that the students were not paying enough attention. This situation was observed during the modelling phase when the lecturer tried to involve the students through questions. None of them was able to repeat what the developed strategy consisted of. But after the teacher repeated the strategy, as they knew they were going to be questioned later, they became more engaged. The next steps were implemented in the next class, where everything went smoothly. The other

experiment sessions went smoothly as well. On the other hand, during the guided programming phase, groups of 3-5 students were randomly formed with the students' agreement.

Intervention with Group 2: Fifteen students were randomly selected to take part in the experiment. The teaching method was introduced to the participants at the beginning of the first class. Students' roles were clearly outlined for each stage of the framework. The IF statement and For loop were taught in a week-long experiment. Compared to the previous intervention, all stages of the teaching method were run throughout each course session. Participants reacted smoothly and were engaged during the experiment, which recorded no missed sessions.

4) *Challenges*: The challenges we faced when preparing the experiment with Group 1 were related to developing a good strategy, and finding out about students' misconceptions. We solved the strategy issue by introducing the binary files with the problem-based learning technique [14], led exclusively by the lecturer. Some of the most common misconceptions students have about binary files in C++ were identified by our programming experiences as teachers because we couldn't find them in the literature. A further challenge was to develop examples and exercises that matched students' interests. Indeed, most of these students were not interested enough when the examples and exercises were based on mathematics problems, which led us to frame our examples and exercises on problems from daily life. Another challenge was the lesson duration; there were times when it did not cover all the EI framework phases, and in such a case, during the next class, we continued with the remaining ones. From this experience, this makes more work for the teacher in preparation, but when it is done right, the method seems easy to implement, and the impact on student motivation is real. Unfortunately, in this experiment, we did not develop CIs. There were no major challenges related to the Group 2 experience.

5) *Impact*: The IP-EI framework, with its five steps, frames and guides the teaching process from start to finish. This experiment has shown us that it can be used as an effective technique for improving the traditional teaching method in programming teaching modules. Indeed, many methods exist to improve teaching IP modules, but the traditional method is still used by most teachers despite its inefficacy [56].

4. Survey on Students' Motivation

4.1 Methodology

To answer the above-mentioned research question, we conducted a survey following a quantitative research methodology, utilizing a structured questionnaire.

1) *Participants*: There were 49 randomly selected students, distributed as follows:
   - 34 students from Group 1, as attested by the class attendance lists elaborated during classes. This sample size corresponds to the exact number of students who attended all classes and agreed to take part in the survey during the last experiment class.
   - 15 students from Group 2, were randomly selected from the 30 students enrolled in the class.
2) *Data collection and analysis*: For data collection, this study distributed questionnaires to participants at the end of the experiments. The 21 questionnaire statements were developed based on the Instructional Material Motivation Survey (IMMS) [57]. As the IMMS statements have been developed based on the four components of the ARCS model, our questionnaire therefore comprises four sections corresponding to each of the ARCS model categories: attention, relevance, confidence, and satisfaction. Responses to the questions in each section were based on a 5-point Likert scale: Strongly disagree (1), Disagree (2), Fairly agree (3), Agree (4), and Strongly disagree (5). The data collected correspond to the responses of 49 students who completed the questionnaire to assess the impact of the proposed EI framework on motivation. On the other hand, we used descriptive statistics to analyze the quantitative data collected, utilizing SPSS software.
3) *Ethical considerations*: Ethical clearance was applied for and approved by the FNAS ethics committee before the research commenced. There were no cases of ethical violations.
4) *Reliability of the scale*: The internal consistency estimates for all 21 items of the questionnaire, as measured by Cronbach's alpha, indicate a satisfactory level of internal consistency, with a value of 0.78 for the whole scale.

4.2 Results and Analysis

This part describes the analysis and results of the data collected. We answered research question 2 by considering students' motivation across the four ARCS model categories, as well as each of the individual elements within these categories (see Table 1). We analyze each unit's outcomes in the categories as follows:

Table 1. Outcomes of the descriptive analysis of participants' motivation.

Code	Criteria	Mean	Mode	SD*
Attention		**3.88**	**4**	**0.81**
A1	The content has things that stimulated my curiosity	3.94	4	0.92
A2	The quality of feedback helped to hold my attention	3.69	4	1.00
A3	The variety of examples helped keep my attention	3.69	4	0.96
A4	The explanations were so abstract that it was hard to keep my attention	2.43	2	1.10
A5	The integration of misconceptions in the course content helped keep my attention	3.82	4	0.99
Relevance		**3.65**	**4**	**0.81**
R1	The way in which the content was presented conveyed the impression that the related programming concepts are worth knowing	3.73	4	0.91
R2	The strategies developed during classes were useful to me	3.47	4	1.14
R3	The examples or exercises matched my interest	3.35	3	0.78
R4	The examples or exercises were all relevant to me	3.67	3	0.94
R5	There were examples or exercises that showed me how programming solves a real-life problem	4.14	4	0.87
Confidence		**3.86**	**4**	**0.76**
C1	As I worked in pairs and individually as well on programming problems, I was confident that I could learn programming	4.27	5	0.78
C2	I felt confident in solving a real-life problem	3.61	4	0.93
C3	After attending classes for a while, I was confident that I would be able to pass a test or exam	3.65	4	0.97
C4	I could not understand anything after multiple attempts working on programming problems whether in pairs or individually	2.29	2	1.04
C5	The strategies were more difficult to understand	2.16	2	1.16
Satisfaction		**3.94**	**5**	**0.97**
S1	I enjoyed the classes so much that I would like to know more about programming	3.73	4	0.97
S2	This teaching approach improved my programming learning experience	3.49	4	1.12
S3	It was a pleasure to participate in such an experience	4.02	4	0.88
S4	Feedback provided during guided activities was well appreciated	3.71	4	1.04
S5	Completing the exercises individually gave me a satisfying feeling of accomplishment	3.96	4	0.89
S6	This teaching approach does not provide more satisfaction than traditional teaching	2.33	2	1.21

*SD: Standard deviation

1) *Attention*: This study assessed respondents' attention based on five statements, and analysis of the results presented in Fig. 3 shows that most participants agreed with the following points: the content has things that stimulated their curiosity; the quality of feedback helped them hold their attention; the variety of examples helped keep their attention; and the integration of misconceptions helped keep their attention. Conversely, with an average score of 2.43 (SD = 1.10), the majority disagreed that the explanations provided during classes were abstract and difficult to maintain their attention. According to the results in Table 1, the average attention score is 3.88 (SD= 0.81), indicating that the majority of participants agree that the EI framework captivated their attention.

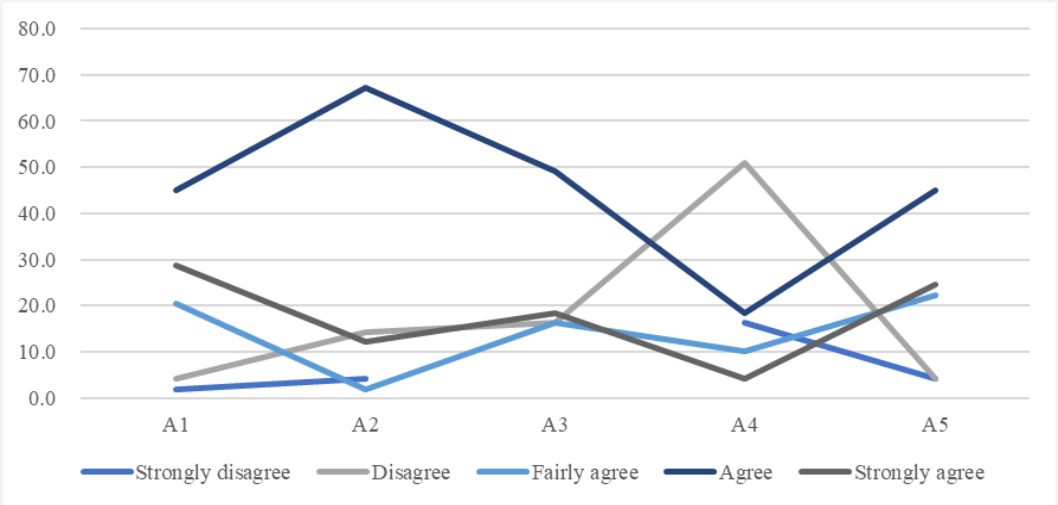

Fig. 3. Attention.

2) *Relevance:* This paper assessed participants' relevance based on five statements. Analysis of the results presented in Fig. 4 reveals that most participants agreed with statements R1, R2, R4, and R5, with the following respective average scores 3.73 (SD = 0.80), 3.47 (SD = 0.91), 3.67 (SD = 0.94), and 4.14 (SD = 0.87). On the other hand, most respondents fairly agreed with the statement R3, with an average score of 3.35 (SD=0.94), meaning that examples or exercises need to be improved to better hold students' interest and relevance.

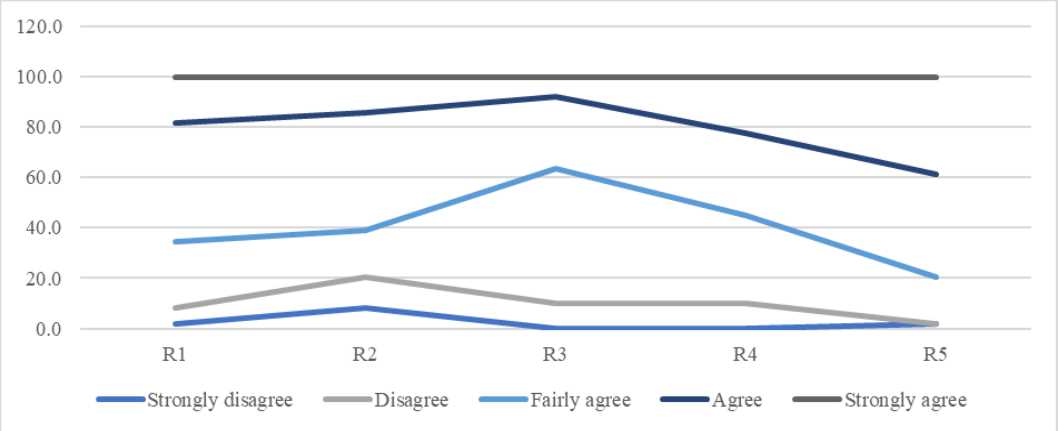

Fig. 4. Relevance.

3) *Confidence:* This study measured participants' attention on the strength of five criteria, and an examination of the findings as shown in Fig. 5 indicates that students mostly strongly agreed that the experience during the guided programming step improved their confidence in programming. The analysis of the results in Table 1 shows in detail that participants mostly agree with C1, C2, and C3, with the following respective average scores: 4.27 (SD = 0.78), 3.61 (SD = 0.93), and 3.65 (SD = 0.97). Regarding statements C4 and C5, the majority of participants expressed disagreement, with the value of 2 being the most frequently mentioned.

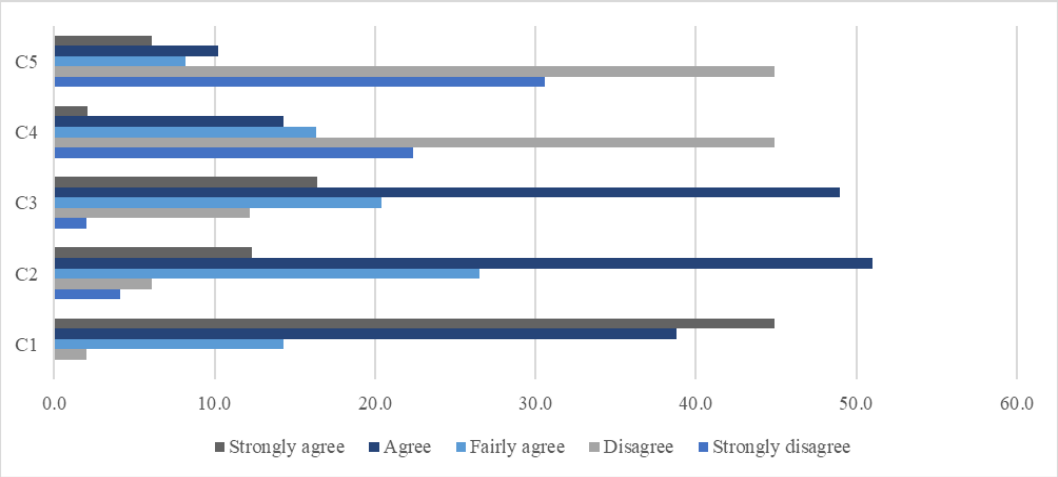

Fig. 5. Confidence.

4) *Satisfaction*: The examination of the results shown in Fig. 6 of the six criteria used to measure students' satisfaction after the experiments highlighted that the majority of participants at least agreed with S1, S2, S3, S4, and S5. According to the results in Table 1, the majority of participants rated these statements as 4. On the contrary, most of them disagreed that the novel teaching framework did not provide more satisfaction than traditional teaching, with the value of 2 being the most frequently mentioned.

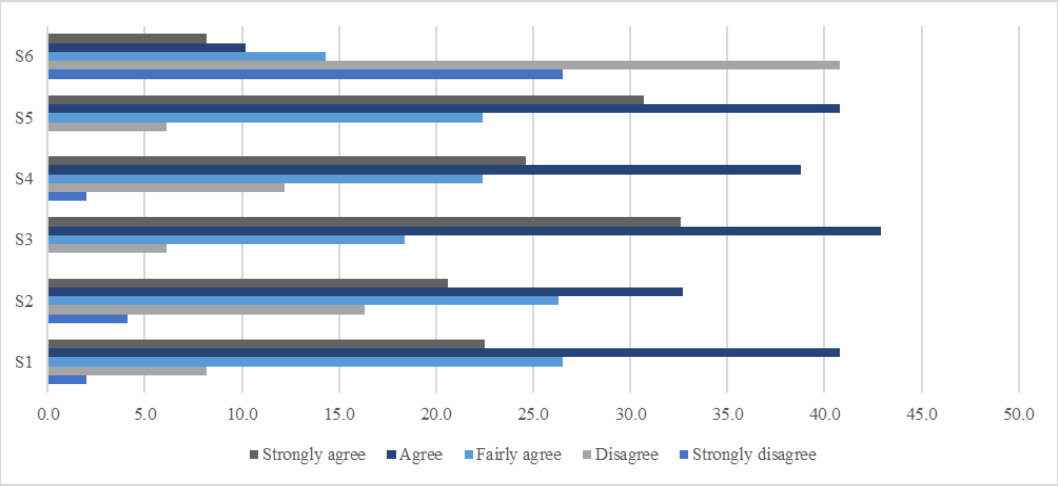

Fig. 6. Satisfaction.

5) *Correlation between ARCS model categories:* The Pearson inter-factor correlation coefficients obtained show a substantial positive link between all ARCS dimensions (see Table 2). The categories of relevance and satisfaction have the strongest association (r = 0.561), whereas attention and relevance have the lowest (r = 0.414).

Table 2. Correlation between ARCS dimensions

Scale	Attention	Relevance	Confidence	Satisfaction
Attention	1	0.414*	0.478*	0.418*
Relevance	0.414*	1	0.528*	0.561*
Confidence	0.478*	0.528*	1	0.440*
Satisfaction	0.418*	0.561*	0.440*	1

*Correlation is significant at the 0.01 level (2-tailed)

4.3 Discussion

This part discusses the results concerning some difficulties encountered by non-computing students, as documented in [1], and also discusses how the results align with or differ from previous research. An earlier study [1] into the difficulties encountered by non-computing students in programming showed that they were not attentive during lessons and were largely unconfident. As a response to this, the results of the previous section demonstrated that using the IPEI framework overcame these difficulties, as it significantly improved their attention and confidence (see Fig.7). This is probably due to the involvement of students through questions during the model-based programming phase and the feedback they received during the guided programming phase. Similarly, the integration of peer programming followed by individual programming in its stages is a positive contribution to teaching practices, and most participants found it a great source of confidence. On the other hand, although most participants found the implementation of the IPEI framework relevant, we note that a significant proportion of them fairly agree with the relevance of the exercises capable of addressing their interests. Other research found similar outcomes [9]. On the other hand, Horváth and Javorsk [7] observe a frequent mismatch between the teacher's explanations and the students' understanding, indicating a significant failure factor in education. The EI framework fixes this problem by looking at criteria R2 (mean = 3.47, SD = 1.14) and R5 (mean = 4.14, SD = 0.87). Furthermore, our findings indicate that quality feedback is critical for attracting students' attention (mean = 3.69, SD = 0.96) and increasing their satisfaction (mean = 3.71, SD = 1.04), which is consistent with Jiau et al.'s [8] findings on the influence of feedback on students' self-motivation. Additionally, the varying standard deviation values for the criteria in Table 1 suggest that the participants' answers are homogeneous, implying that their responses are very close to the computed averages. As a result, the EI framework improves student motivation in all four areas of the ARCS motivation model, as does Santana et al's mixed-context teaching method [10]. Unlike Souza and Bittencourt's problem-based learning technique [9], which solely affected the categories of attention and satisfaction. Moreover, this novel framework provided satisfaction to participants, as most of them enjoyed the classes so much, found the framework more efficient than the traditional teaching method, and were pleased to participate in such a teaching experience.

The development of a novel teaching approach to boost the skills of non-computer science students in the IP module has sparked renewed interest in computer science education [10,14]. This study contributes to this trend by proposing a novel teaching framework based on cognitivist theory, which aims to influence the motivation of

non-computing students who primarily take this course as an elective. The efficiency of the approach is dependent on the extent to which the instructor can effectively manage the time, prepare for the course, and successfully operate the classroom. If the modelling stage is not well organized, for example, we have discovered that there will not be enough time to finish the remaining framework stages. Despite the high levels of motivation demonstrated by the study's results, further research is necessary to validate and enhance the findings. This testing should take place over a semester, with a larger student population.

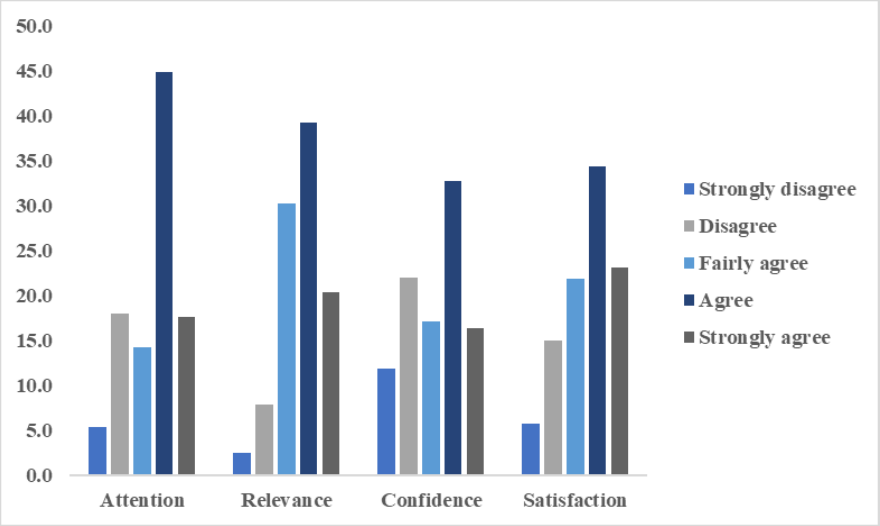

Fig. 7. ARCS-model summary results

4.4 Validity Threats

To ensure that the validity of the results presented above is not compromised, several measures have been taken. Our sample consists of 49 students out of a population of around 88, giving us a representativeness rate of 55.6%, which, according to Lokesh's theory [58], is representative of the population, since this rate is higher than 20%. Moreover, to avoid response bias, questions were asked about the various teaching activities that took place during the lectures. Also, we were willing to provide and clarify certain incomprehensible questions. In addition, to avoid measurement bias, responses were based on a 5-point Likert scale to accurately measure participants' consent. To ensure data quality, a questionnaire was given to each participant, and only volunteer students were surveyed, those who were present for the entire duration of the experiment. The results analysis showed that all participants answered all the questions, and we are therefore confident in the quality of the results.

5. Impact and Comparison with Existing Teaching Methods

The IPEI framework is a novel approach to teaching introductory programming modules. Explicit teaching is the foundation of this approach and has been very successful in the teaching of mathematics, foreign languages, and so on. IPEI has five steps, and contrary to other IP teaching methods [2,4,5,8,42,55,57], bounds all the activities of learners and teachers during classes. The IPEI approach can be used for both lectures and practical classes, which is not the case with most available IP teaching methods that instead emphasize coding or problem-solving activities. Furthermore, compared with traditional and other IP teaching approaches, IPEI focuses more on teaching strategies to better grasp programming content and prevent shallow learning. For example, as stated by Malik et al., the traditional approach favours surface rather than deep learning, as problem-solving or content comprehension strategies are not highlighted [54]. On the other hand, the IPEI approach requires a lot from teachers before and during classes, starting with content preparation, research and elaboration of misconceptions, and strategy preparation. The preparation of misconceptions and strategies is not common to most existing IP teaching methods. Table 3 provides a comparison of the IPEI approach with some existing IP teaching methods (rows in the table) based on some selected criteria (columns in the table). According to Table 2, among other observations, all the pedagogical approaches use constructivism as a learning theory and are aimed primarily at CS students, unlike IPEI, which uses cognitivism. This study assumes that the IPEI approach can significantly boost the traditional teaching method used extensively in the teaching of introductory programming modules.

Furthermore, the academic values of this study are as follows: (1) it addressed a significant gap in the existing literature by focusing on non-computing students, as they are often overlooked in research [1,14], (2) it introduced a cognitive-based teaching framework based on EI to provide a novel approach to teaching programming, and (3) it involved students from two different universities, adding a cross-cultural perspective to the study and paving the way

for more global and inclusive studies in the future. Similarly, in terms of application implications, the study enhanced students' motivation, resulting in improved student engagement in IP modules. Educators can easily adopt the concrete teaching strategies it provided in their classrooms. The proposed method could make programming more accessible and enjoyable for non-computing students, potentially encouraging a more diverse group of students to explore computing fields. The findings of this study could be used to inform curriculum development and teaching practices in IP modules, particularly those with a diverse student body, including non-computing majors. Lesson plans and teaching materials could incorporate our proposed teaching framework. In addition, the positive effect on student motivation could lead to higher course completion rates and better student outcomes. In general, the study offers valuable insights that could benefit educators and students alike.

Table 3. Comparison of IPEI with some IP teaching methods.

Teaching methods	Learning theory	Target population	Features
IPEI approach	Cognitivism	Non-CS students with no prior programming or mathematics experiences, slow-paced students	Includes 5 steps and promotes motivation, and self-efficacy. The teacher is active or a Facilitator. The learner is active and passive depending on the step we are in. The method can be used for both practical and lecture classes and promote students' motivation.
Traditional method	Behaviourism	All students	Promotes performance. The teacher is active and the learner is passive. The method can be used for both practical and lecture classes.
Low-level approach [55]	Constructivism	First-year students	This approach is before the beginning of the programming classes and promotes performance. The teacher is the facilitator and the learner is active. The method can be used for lecture classes.
Agile process [56]	Constructivism	First-year programming with no prior programming experience	Includes 4 steps with teacher feedback and promotes motivation and engagement. The teacher is the facilitator and the learner is active. The method can be used for practical classes.
ADRI model [57]	N/A	CS students	Includes 4 steps and promotes more practice of programming. The method can be used for both practical and lecture classes.
Problem-based learning (PBL) [9]	Constructivism	CS students	Promotes problem-solving skills, critical thinking, motivation and engagement. The method can be used for practical classes.
Mixed-methods survey design (Gamification with Digital Badges) [4,40,41]	Constructivism	Mostly CS students	Promotes motivation. The method can be used for practical classes.
mixed contexts teaching approach [6]	N/A	Non-CS (Civil Engineering students)	Promotes motivation.
Holistic approach (Python as the first programming language, project-oriented and PBL, multimedia resources, and evaluation rubrics) [6]	Constructivism	CS	Promotes engagement. The teacher is the facilitator, and the learner is active. The method can be used for practical and lecture classes.
Peer review [8]	Constructivism	CS	Promotes engagement and performance. The teacher's role is not mentioned and the active learner. The method can be used for practical classes.

6. Conclusion

In this article, we introduced a new way of teaching computer programming called the EI framework, designed to boost the motivation of non-computing students. The EI framework, based on cognitivist theory, consists of five phases: reviewing, model-based programming teaching, guided programming, non-guided programming, and report and transfer. What sets it apart is that it can be used for both lectures and practical sessions, unlike other methods that mainly focus on coding and are student-centric. Our approach, IPEI, is teacher-centric, offering significant guidance to students. Two case studies were performed to gauge the new approach's effectiveness, using a questionnaire with 49 students in the CMPG121 module at FNAS of the North-West University, South Africa, and INFO212 at the HTTC, University of Yaoundé 1, Cameroon. The results showed that the EI framework positively impacted students' motivation in terms of attendance, relevance, confidence, and satisfaction, according to the ARCS model categories. Despite these positive

findings, there's a need to create exercises and examples based on real-life problems to engage students more effectively. Investing time in preparing high-quality strategies and addressing misconceptions is crucial for better outcomes. This paper contributes to CS education research by introducing innovative teaching methods to enhance student's skills in IP modules. Future research should experiment with this method across various programming concepts in different institutions to validate and improve the results as needed.

References

[1] A. Mbiada et al., 'A Comparative Study of Computer Programming Challenges of Computing and Non-Computing First-Year Students', *Indonesia. J. Comput. Sci.*, vol. 12, no. 4, 2023.

[2] E. Hong et al., 'Effects of explicit instructions, metacognition, and motivation on creative performance', *Creat. Res. J.*, vol. 28, no. 1, pp. 33–45, 2016.

[3] A. L. Ribeiro et al., 'Análise da Motivação em um Estudo Integrado de Programação Baseado em PBL', in *Anais do XXVI Workshop sobre Educação em Computação*, 2018.

[4] K. Longi and others, 'Exploring factors that affect performance on introductory programming courses', 2016.

[5] A. Forte and M. Guzdial, 'Computers for communication, not calculation: Media as a motivation and context for learning', in *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the*, 2004, pp. 10-pp.

[6] O. S. Pabón and L. M. Villegas, 'Fostering motivation and improving student performance in an introductory programming course: An integrated teaching approach', *Rev. EIA*, vol. 16, no. 31, pp. 65–76, 2019.

[7] R. Horváth and S. Javorskỳ, 'New teaching model for Java programming subjects', *Procedia-Soc. Behav. Sci.*, vol. 116, pp. 5188–5193, 2014.

[8] H. C. Jiau et al., 'Enhancing self-motivation in learning programming using game-based simulation and metrics', *IEEE Trans. Educ.*, vol. 52, no. 4, pp. 555–562, 2009.

[9] S. M. Souza and R. A. Bittencourt, 'Motivation and engagement with PBL in an introductory programming course', in *2019 IEEE Frontiers in Education Conference (FIE)*, 2019, pp. 1–9.

[10] B. L. Santana et al., 'Motivation of engineering students with a mixed-contexts approach to introductory programming', in *2018 IEEE Frontiers in Education Conference (FIE)*, 2018, pp. 1–9.

[11] L. Facey-Shaw et al., 'Do badges affect intrinsic motivation in introductory programming students?', *Simul. Gaming*, vol. 51, no. 1, pp. 33–54, 2020.

[12] A. Carbonaro, 'Good practices to influence engagement and learning outcomes on a traditional introductory programming course', *Interact. Learn. Environ.*, vol. 27, no. 7, pp. 919–926, 2019.

[13] P. A. Kirschner et al., 'Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching', *Educ. Psychol.*, vol. 41, no. 2, pp. 75–86, 2006.

[14] A. K. Mbiada et al., 'Introductory Computer Programming Teaching and Learning Approaches: Review', in *2022 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, 2022[Online]. Availablehttps://doi.org/10.1109%2Ficecet55527.2022.9873427.

[15] C. Gauthier et al., 'L'enseignement explicite', 2007.

[16] A. L. Archer and C. A. Hughes, 'Explicit Instruction: Effective and efficient teaching (what works for special-needs Learners)', *J. Spec. Educ.*, vol. 36, no. 4, pp. 186–205, 2011.

[17] J. R. Hollingsworth and S. E. Ybarra, *Explicit direct instruction (EDI): The power of the well-crafted, well-taught lesson*. Corwin Press, 2009.

[18] J. M. Keller, 'Development and use of the ARCS model of instructional design', *J. Instr. Dev.*, vol. 10, no. 3, pp. 2–10, 1987.

[19] C. Iriarte and S. Bayona, 'IT projects success factors: a literature review', *Int. J. Inf. Syst. Proj. Manag.*, vol. 8, no. 2, pp. 49–78, 2020.

[20] R. M. Ryan and E. L. Deci, 'Intrinsic and extrinsic motivations: Classic definitions and new directions', *Contemp. Educ. Psychol.*, vol. 25, no. 1, pp. 54–67, 2000.

[21] G. Nel and L. Nel, 'Motivational value of code. Org's code studio tutorials in an undergraduate programming course', in *ICT Education: 47th Annual Conference of the Southern African Computer Lecturers' Association, SACLA 2018, Gordon's Bay, South Africa, June 18–20, 2018, Revised Selected Papers 47*, 2019, pp. 173–188.

[22] D. Bédard et al., 'Problem-based and project-based learning in engineering and medicine: determinants of students' engagement and persistance', *Interdiscip. J. Probl.-Based Learn.*, vol. 6, no. 2, pp. 7–30, 2012.

[23] W. B. Schaufeli et al., 'Utrecht work engagement scale-9', *Educ. Psychol. Meas.*, 2003.

[24] S. Wiedenbeck et al., 'Factors affecting course outcomes in introductory programming.', in *PPIG*, 2004, p. 11.

[25] C. O'Malley and A. Aggarwal, 'Evaluating the Use and Effectiveness of Ungraded Practice Problems in an Introductory Programming Course', in *Proceedings of the Twenty-Second Australasian Computing Education Conference*, 2020, pp. 177–184.

[26] G. Sharma, 'Computer Programming Comp103: Who Does Better?', 2019.

[27] K. S. Taber, *Modeling learners and learning in science education*. Springer, 2013.

[28] Y. Qian et al., 'Teachers' perceptions of student misconceptions in introductory programming', *J. Educ. Comput. Res.*, vol. 58, no. 2, pp. 364–397, 2020.

[29] A. K. Veerasamy et al., 'Identifying novice student programming misconceptions and errors from summative assessments', *J. Educ. Technol. Syst.*, vol. 45, no. 1, pp. 50–73, 2016.

[30] Ž. Žanko et al., 'Mediated transfer: impact on programming misconceptions', *J. Comput. Educ.*, pp. 1–26, 2022.

[31] F. Johnson et al., 'Experience Report: Identifying Unexpected Programming Misconceptions with a Computer Systems Approach', in *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1*, 2022, pp. 325–330.

[32] F. Johnson et al., 'Analysis of student misconceptions using Python as an introductory programming language', in *Proceedings of the 4th Conference on Computing Education Practice 2020*, 2020, pp. 1–4.
[33] Ž. Žanko et al., 'Analysis of school students' misconceptions about basic programming concepts', *J. Comput. Assist. Learn.*, vol. 38, no. 3, pp. 719–730, 2022.
[34] A. Altadmri and N. C. Brown, '37 million compilations: Investigating novice programming mistakes in large-scale student data', in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 2015, pp. 522–527.
[35] C. Taylor et al., 'Computer science concept inventories: past and future', *Comput. Sci. Educ.*, vol. 24, no. 4, pp. 253–276, 2014.
[36] G. L. Herman, 'The development of a digital logic concept inventory', PhD Thesis, University of Illinois at Urbana-Champaign, 2011.
[37] J. Libarkin, 'Concept inventories in higher education science', in *BOSE Conf.*, 2008.
[38] [V. L. Almstrum et al., 'Concept inventories in computer science for the topic discrete mathematics', in *ACM SIGCSE Bulletin*, 2006, vol. 38, pp. 132–145.
[39] D. C. Leonard, *Learning theories: A to Z*. Bloomsbury Publishing USA, 2002.
[40] J. Piaget, 'Part I: Cognitive Development in Children–Piaget Development and Learning.', *J. Res. Sci. Teach.*, vol. 40, 2003.
[41] J. M. Fletcher et al., *Learning disabilities: From identification to intervention*. Guilford Publications, 2018.
[42] M. Pressley and K. R. Harris, 'Cognitive strategies instruction: From basic research to classroom instruction', *J. Educ.*, vol. 189, no. 1–2, pp. 77–94, 2009.
[43] M. Bocquillon et al., 'Faut-il utiliser l'enseignement explicite en tout temps? Non... mais oui!', *Apprendre Enseigner Aujourd'hui*, vol. 8, no. 2, pp. 25–28, 2019.
[44] Newmark, 'Ten principles for great explicit teaching'. [Online]. Availablehttps://bennewmark.wordpress.com/2017/10/07/ten-principles-for-great-explict-teaching/.
[45] M. Rogers et al., 'Exploring personalization of gamification in an introductory programming course', in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 2021, pp. 1121–1127.
[46] R. Smiderle et al., 'impac', *Smart Learn. Environ.*, vol. 7, no. 1, pp. 1–11, 2020.
[47] B. Hartanto et al., 'Enhancing the student engagement in an introductory programming: a holistic approach in improving the student grade in the informatics department of the University of Surabaya', *Dalam Commun. Comput. Inf. Sci.*, vol. 516, pp. 493–504, 2015.
[48] R. A. Alturki and others, 'Measuring and improving student performance in an introductory programming course', *Inform. Educ.- Int. J.*, vol. 15, no. 2, pp. 183–204, 2016.
[49] J. Kasurinen and U. Nikula, 'Lower dropout rates and better grades through revised course infrastructure', in *Proceedings of the 10th International Conference on Computers and Advanced Technology in Education*, 2007, pp. 152–157.
[50] K. Kwon, 'Novice programmer's misconception of programming reflected on problem-solving plans', *Int. J. Comput. Sci. Educ. Sch.*, vol. 1, no. 4, pp. 14–24, 2017.
[51] R. M. Gagne, 'Instruction and the conditions of learning', *Psychol. Sch. Learn. Views Learn.*, vol. 1, pp. 153–175, 1974.
[52] T. H. Spangsberg and M. Brynskov, 'Code-labelling: A teaching activity encouraging deep learning in a non-STEM introductory programming course', in *2017 12th International Conference on Computer Science and Education (ICCSE)*, 2017, pp. 95–100.
[53] F. J. Gallego-Durán et al., 'A low-level approach to improve programming learning', *Univers. Access Inf. Soc.*, pp. 1–15, 2020.
[54] S. H. Edwards et al., 'The Relationship Between Voluntary Practice of Short Programming Exercises and Exam Performance', in *Proceedings of the ACM Conference on Global Computing Education*, 2019, pp. 113–119.
[55] B. Pasian, 'The Constructive Research Approach: Problem Solving for Complex Projects', in *Designs, Methods and Practices for Research of Project Management*, Routledge, 2016, pp. 125–136.
[56] K. Daungcharone et al., 'Smart Learning Environment to Augment the Learners' Programming Skills', in *2020 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON)*, 2020, pp. 293–297.
[57] W. Huang et al., 'A preliminary validation of Attention, Relevance, Confidence and Satisfaction model-based Instructional Material Motivational Survey in a computer-based tutorial setting', *Br. J. Educ. Technol.*, vol. 37, no. 2, pp. 243–259, 2006.
[58] K. Lokesh, *Methodology of educational research*. Vikas Publishing House, 1984.
[59] S. I. Malik et al., 'Promoting Algorithmic Thinking in an Introductory Programming Course.', *Int. J. Emerg. Technol. Learn.*, vol. 14, no. 1, 2019.
[60] B. Isong, 'A Methodology for Teaching Computer Programming: first year students' perspective', *Int. J. Mod. Educ. Comput. Sci.*, vol. 6, no. 9, p. 15, 2014.
[61] S. I. Malik and J. Coldwell-Neilson, 'A model for teaching an introductory programming course using ADRI', *Educ. Inf. Technol.*, vol. 22, no. 3, pp. 1089–1120, 2017.

Authors' Profiles

Alain Kabo Mbiada, PhD candidate (Computer Science), Department of Computer Science of North-West University, Mafikeng, South Africa. Areas of scientific interests: Computer Science Education, Educational Technology, and Software Engineering.

Bassey Isong, he received his BSc degree in Computer Science from the University of Calabar, Nigeria in 2004, MSc. degrees in Computer Science and Software Engineering from the Blekinge Institute of Technology, Sweden in 2008 and 2010 respectively, and a PhD degree in Computer Science from the North-West University, in 2014. His research interests include and are not limited to Software Engineering, Software Defined Networks, the Internet of Things and Low Power Wide Area Networks, Machine Learning, and Blockchain. He is currently an Associate Professor in the Department of Computer Science, at North-West University, Mafikeng Campus, South Africa. He is also a member of the IEEE Computer, Communication, and Education Societies as well as ACM.

How to cite this paper: Alain Kabo Mbiada, Bassey Isong, "Explicit Instruction-based Methodology for Teaching Introductory Computer Programming", International Journal of Modern Education and Computer Science(IJMECS), Vol.17, No.1, pp. 1-16, 2025. DOI:10.5815/ijmecs.2025.01.01