Journal of Research in Mathematics, Science, and Technology Education
ISSN 3047-5163 (online) & 3047-6933 (print)
October 2025, Vol. 2, No. 2, pp. 103–118

Research Article

Implementing Whole Brain Teaching Within A Pre-Service Teachers' Introductory Programming Instruction: An Action Research Study

Fatimah Yetunde Akinrinola[1], **Nofiu Oriyomi Iposu**[1]

[1]Department of Computer Science Education, College of Information and Technology Education, Lagos State University of Education (LASUED), Lagos, Nigeria

Abstract
This study investigated the implementation of whole brain teaching within a constructivist teaching strategy in pre-service teachers' programming education, towards enhancing their knowledge of procedural programming. From a qualitative paradigm lens, hermeneutic phenomenology guided the inquiry. The research strategy involves two action research cycles involving fifty-eight pre-service teachers purposively sampled over two academic sessions. The Whole Brain instructional plan was designed to facilitate the programming instruction. Data were collected through surveys, interviews, and classroom observations. The findings generated two themes: programming intervention promoted student engagement and pre-service teachers' teaching strategies through self-confidence in problem-solving and self-regulation of learning. These results contributed to the literature on pedagogical innovations in the teaching of programming and provided recommendations for enhancing pre-service teachers' programming learning with a focus on holistic brain development through the implementation of the whole-brain programming walk-around and instructional plan for programming lessons. For transparency in research, it is necessary to state that even though the data for this study were collected in 2016 and 2017, the data are still significant within the context of programming education. The findings of this study support current literature on constructivist learning and also address gaps in the implementation of whole-brain learning for programming teaching. The findings of this study cannot be generalized due to the nature of action research studies, but can be replicated in another context. Further exploration with larger sample sizes and diverse contexts could bolster the robustness and generalizability of these findings.

Keywords: Whole-Brain Teaching and Learning, Programming Walk-Around, Cooperative Learning, Collaborative Learning, Instructional Scaffolding

✉ Correspondence
Fatimah Yetunde Akinrinola
akinrinolafy@lasued.edu.ng

Received
December 20, 2024
Accepted
May 13, 2025
Published
October 1, 2025

Citation: Akinrinola, F. Y., & Iposu, N. O. (2025). Implementing whole brain teaching within a pre-service teachers' introductory programming instruction: An action research study. *Journal of Research in Mathematics, Science, and Technology Education, 2*(2), 103–118.

DOI: 10.70232/jrmste.v2i2.32

1. INTRODUCTION

Programming, an essential skill for computer science students, is a complex construct. Understanding programming proves to be a challenging and sometimes tedious endeavor, especially for novice students globally (Cheah, 2020; Dasgupta & Hill, 2017; Medeiros et al., 2018). Novice students encounter difficulties comprehending the complexity of programming concepts, attributing their struggles to the intricate nature of programming instruction, unfamiliar syntax, and problem-solving abilities (Koulouri et al., 2016; Topalli & Cagiltay, 2018; Yurdugül & Aşkar, 2013). These challenges collectively contribute to student failure, attrition, and dropout rates in programming courses (Bennedsen & Caspersen, 2019; Hawlitschek et al., 2020; Law et al., 2010). However, its relevance in society cannot be overemphasized, because programming provides a vital foundation for students to excel both in the industry and the modern workplace. Meanwhile, achieving success in programming often rests on the teacher's pedagogical methods (Medeiros et al, 2019). Teachers often achieve less success than expected (Bennedsen & Caspersen, 2019; Berssanette & de Francisco, 2021; Simon & Hanks, 2008). For instance,

(Berssanette & de Francisco, 2021) stressed that problems related to lack of success in programming teaching originate from intrinsic teachers' practices developed during the teaching/learning process. In addition, the traditional lecture method is commonly used in different contexts, and teachers seldom diversify their pedagogical approaches to bridge the gap between abstract programming concepts and real-world applications. Students are exposed to theoretical programming teaching with limited exposure to practical applications (Isong, 2014), leaving students unskilled and unprepared for the future workforce. This teaching approach has been questioned in all facets of knowledge.

Furthermore, the spike in higher education institutions (HEIs) enrolment has presented challenges in managing student diversity and varying learning preferences in the classroom. Therefore, educators must engage learning with pedagogical practices that accommodate the varying diversities within the classroom (Opdecam et al., 2014). Therefore, Entwistle et al. (2010) and Entwistle et al. (2002) suggest that teachers' understanding of students' diversities can guide them in selecting appropriate strategies for designing instruction that meets individual needs. Importantly, the 21st century educational landscape has witnessed a paradigm shift in pedagogical approaches, driven by the integration of technology and the growing recognition of the need to nurture holistic learning experiences, and encouraging students to be active learners and participants in creating information, and generating new ideas (Luna Scott, 2015) for an engaging learning experience (McLoughlin & Lee, 2008). Cultivating an awareness of learners' individual learning preferences becomes crucial for effective teaching and successful learning (Pritchard, 2017). It is worth noting that, as the modern classroom becomes increasingly digitalized and interconnected, educators are tasked not only with imparting subject knowledge but also with fostering essential skills that encompass critical thinking, creativity, collaboration, and adaptability (Van Laar et al., 2020). This integration calls for an innovative pedagogical framework that goes beyond traditional methods, aiming to engage and stimulate the entire spectrum of a novice pre-service teacher's cognitive capacity.

This implies that a comprehensive revamp of programming pedagogy that fosters holistic learning through programming interventions is needed. One of such interventions is Whole Brain Teaching (Elfiky, 2022). Whole Brain teaching emphasizes that individuals tend to have preferences for certain thinking modes over others, and this influences their learning, problem-solving, and decision-making approaches (Herrmann, 1996). Although a series of research (Jakovljevic, 2003; Malliarakis et al., 2013; Thota & Whitfield, 2010) fostering holistic learning exists in literature, programming environments tailored to match the unique learning preferences of learners are scarce. This study, therefore, provides an evidence report on a computer programming intervention with a particular focus on the holistic development of novice pre-service teachers.

2. CONCEPTUAL FRAMEWORK

The literature on whole brain teaching and learning (Herrmann, 1995), constructivist teaching strategies (D. Lee et al., 2018), such as collaborative learning (Tan et al., 2022), scaffolding (Ahmadi & Motaghi, 2021), and cooperative learning (Johnson & Johnson, 2018) formed the conceptual framework for this study. Whole brain teaching and learning, and constructivist teaching strategies were combined to facilitate the programming instruction. The study's research question is derived from a comprehensive discussion of each literature base and conceptual framework.

2.1. Whole Brain Teaching and Learning Framework

The recent surge in student admissions has posed challenges due to the increasing diversity among students in Nigerian higher education institutions, highlighting the need for teaching practices that accommodate varied learning characteristics (Opdecam et al., 2014). To address this, Ned Herrmann developed the *Whole Brain Model,* which identifies four cognitive modes associated with brain hemispheres: l*ogical (A quadrant), analytical (B quadrant), relational (C quadrant), and imaginative (D quadrant).* These modes collectively define an individual's thinking profile. Building on Herrmann's model, the *Whole Brain Teaching and Learning (WBTL)* framework integrates these cognitive modes into educational practices, aligning with constructivist learning theory (Bawaneh et al., 2011). WBTL emphasizes engaging all brain facets to create inclusive and effective learning environments (Bawaneh, Abdullah, et al., 2011; Healy et al., 2018). Classroom strategies include organizing students into teams, using creative tasks, and fostering

collaboration to accommodate diverse learning preferences. Herrmann's walk-around model, a subset of the WBTL, enables facilitators to navigate through the framework effectively. This framework accommodates students' learning preferences organized into teams of four to six (Bawaneh, Abdullah, et al., 2011), within the instructional time. The use of creative materials for both individual and group tasks encourages learners to move beyond their comfort zones, thereby fostering collaboration among groups. This cognitive understanding shapes teaching strategies and allows educators to tailor their methods to accommodate individual learners' preferences for learning. The WBTL framework has been successfully applied across various fields, such as physics, mathematics, business management, and language studies (Bawaneh et al., 2011; De Boer et al., 2013; Kirstein & Kunz, 2016; Santoso, 2016; Sontillano, 2018), respectively, to improve learning outcomes. However, its application in programming education remains underexplored.

2.2. Constructivist Strategies for Teaching Programming

Higher education institutions are adopting active learning environments that prioritize student-centered teaching methodologies like constructivism (D. Lee et al., 2018), which places the learner at the forefront of the educational process. Constructivism is therefore crucial for programming skills acquisition.

2.2.1. Collaborative Learning

Collaborative learning (CL), grounded in Vygotsky's socio-cultural theory of learning (1978), underscores the importance of social interactions in the learning process. CL involves participants working together to solve shared problems or achieve results beyond individual capabilities (Herrera-Pavo, 2021; Tan et al., 2022). It encourages students to take responsibility for their learning and support their peers, offering benefits such as brainstorming, peer interaction, resource sharing (Tan et al., 2022), critical thinking, problem-solving (Warsah et al., 2021), deep learning, retention, and improved academic performance (Qureshi et al., 2023). Pair programming (PP) is a notable form of CL in programming, where two individuals share a single keyboard, alternating roles as driver and navigator to collaboratively code and take ownership of the task (Denner et al., 2012; Watkins & Watkins, 2009). Traditional human-human PP and human-AI PP (e.g., with GitHub Copilot) have shown promise in programming education (Dakhel, A. M et al., 2023; Ma et al., 2023). This study focuses on traditional pair programming. PP has demonstrated its effectiveness in enhancing programming knowledge, fostering computational thinking practices (Su et al., 2024), and promoting inclusive learning, collaboration, and meaningful discourse, even in virtual settings (Lubarda et al., 2024). Additionally, integrating collaborative scripts into PP has been shown to improve collaboration, mathematical achievements, and computational thinking skills (Ma et al., 2023).

2.2.2. Instructional Scaffolding

Programming education involves complex constructs, requiring instructional scaffolding to support students both cognitively and motivationally when tackling intricate problems (C. Kim et al., 2022; Näykki et al., 2021). Scaffolding reorganizes students' understanding by providing selective support, broadening options, and facilitating task completion (Belland, 2017; Kim et al., 2020). It can take three forms: one-to-one (teacher support), peer scaffolding (from peers of similar or greater ability), and computer-based scaffolding (tools embedded in curricula or used externally (Belland, 2017). Scaffolding in programming helps students manage learning processes (Su et al., 2024), improves algorithmic and debugging skills (Angeli, 2022; Tikva & Tambouris, 2023), and enhances problem-solving abilities and task completion time (Hou et al., 2022; Margulieux & Catrambone, 2021). For example, Zhang et al (2023) reported in their study that instructors reported that scaffolding effectively develops programming and computational thinking skills.

2.3. Cooperative Learning

Cooperative learning (CL) is a structured educational approach emphasizing group-based problem-solving under instructor supervision (Johnson & Johnson, 2018). It fosters collaborative skills through peer interaction (Ghavifekr, 2020). Despite its benefits, programming education often relies on individualized methods, creating a research gap in exploring group learning approaches, even though novice programmers frequently depend on peer assistance (Garcia, 2021). Effective CL implementation requires five components: *positive interdependence, individual accountability, group processing, social and interpersonal skills,* and *face-to-face interaction* (Johnson & Johnson, 2018). Positive interdependence, the core of CL, ensures task distribution and accountability, enhancing collaboration (Johnson & Johnson, 2018). Group processing involves instructor guidance for productivity and appropriate behavior, supported by skills like leadership, conflict resolution, and effective communication (Johnson & Johnson, 2021). Face-to-face interaction is also vital for fostering cooperation. Educational strategies such as *think-pair-share* (Saka, 2020; Sharma & Saarsar, 2018) and the *jigsaw technique* (Garcia, 2021) are common in CL and improved novices' attitudes and self-efficacy in programming. Similarly, Cecchini et al. (2020) reported significant improvements in motivation, knowledge, and responsibility among pre-service teachers using CL. However, poorly managed groups can hinder learning, as highlighted by Le Roux (2011) and D. Lee et al. (2018).

In summary, this study integrates the Whole Brain Teaching and Learning framework (WBTL) with constructivist strategies, emphasizing their combined potential to address diverse student learning preferences and challenges in programming education. This synergy aims to enhance instructional effectiveness and learning outcomes, particularly for complex programming concepts, through active participation. The fusion of WBTL, collaborative learning, instructional scaffolding, and cooperative techniques is designed to boost student engagement, academic achievement, and comprehension of programming concepts (Cecchini et al., 2020; Lubarda et al., 2024; Su et al., 2024). These strategies are anticipated to significantly enrich the learning experiences of pre-service teachers.

2.4. Research Question

How can whole brain teaching infused with constructivist teaching strategies be integrated into pre-service teachers' programming education to address diverse learner preferences, towards enhancing their knowledge of procedural programming?

3. METHODOLOGY

3.1. Research Paradigm and Strategy

Interpretive paradigm and hermeneutic phenomenology (Klein & Myers, 1999) as a method of inquiry guided the study. The researchers employed Action research (AR) as the research strategy through two cycles of practical AR (Du Toit, 2009a).

3.2. Population and Sampling

The study targeted all first-year computer science students from a Nigerian college of education. Using purposive sampling, 58 students—both male and female—were selected because procedural programming is a mandatory component of the curriculum and a graduation requirement. The participants include male and female first-year computer science pre-service teachers during the first semester of the 2015/2016 and 2016/2017 academic sessions. We understand the age of this data and further educate the reader that the data was collected for a whole semester, which shows the robustness and rigour involved in the data collection process. In addition, this study is still relevant to the context of the study, offering valuable insights that drive ongoing development in the field of programming education. In both first and second cycles, 24 and 34 first-year students, respectively, with 23 male and 35 female participants, were participants of the study. They were distributed in the southwestern part of Nigeria with Lagos (79.4%), Osun (16.7%), Oyo and Ogun (11.91%), Ondo (10.9%), Delta (5.9%), and Kogi, Benue, Edo, and Enugu (2.94%). The students are aged between 17 and 24 years.

3.3. Data Collection

The data collection process spanned the whole first semester of each of the two academic sessions. Data was collected through questionnaires, classroom observations, and interviews. The validated Herrmann Brain Dominance Instrument (HBDI) (Hermann, 1996) and a simulated simplified version (Ngozo, 2012) were used to assess pre-service teachers' learning preferences, with the first instrument containing 120 items and the second a 24-item questionnaire with an associated scoring key. This instrument is scored in the USA by the HBDI practitioner with interpretations for easy understanding. As with the validated HBDI, the simulated HBDI questionnaire supports four quadrants representing A as learning that favours analytical, logical, and fact-based information, B as a sequential, controlled, or linear approach to learning, C as information supporting interpersonal, feeling-based, and emotional, and D as a holistic and conceptual approach to thinking. It utilizes a 5-point Likert-scale of SD-strongly disagree, D-disagree, N-Neutral, A-Agree, and SA-strongly agree. Scoring this profile involves adding up all the total scores for each of the four quadrants. A high score from any of the quadrants represents a strong preference, while quadrants with the same high score represent a double dominance.

The dual use of these instruments was due to the high cost of the online HBDI survey and recommendations against relying on a single instrument for assessing pedagogical change (Coffield et al., 2004). Participatory observations provided insights into group dynamics in each programming classroom, recorded on video over a 2-hour duration. Interviews were conducted to explore participants' lived experiences. The selection of six and eight pre-service teachers for the interviews in both the first and second cycles, respectively, was voluntary, with each interview spanning an hour. The first researcher acted as participant and human instrument. This combined role may have introduced observer error and bias in the collected data. These biases were mitigated by engaging deeply with the data and reflecting on the observations made. Table 1 gives a summary of the data collected for the study.

Table1. Summary of Data Collected and Participant Selection for First and Second Cycles

Data Collection Strategy	No of Participants	Place and Time	Duration	Research Instrument	Role of the Researcher
Questionnaire	22 students	At school	Depending on the student's pace	Validated HBDI instrument	Guided the students
	36 students			Simulated HBDI	Guided the students
Questionnaire	1 lecturer	Outside the research area	30minutes	HBDI instrument	Fill questionnaire
Observation	58 students	During the classroom session	2 hours	Video, field notes	Participant Observer
Interview	14 students	At school	1 hour per student	Interview protocol	Interviewer

3.4. Research Procedure

The research procedure followed the five stages of the visionary AR model (Du Toit, 2009b), designed in line with the Teaching and Learning Process Framework (TLPF) of (Tijani et al., 2020). The framework ensures a systematic approach to designing educational experiences, with each phase of learning leading to the next. A validated HBDI survey was used to identify the participant researcher's learning preference who exhibits a double dominance in quadrants B and C, with secondary preferences in quadrants A and D, which are at times functional (Figure 1).

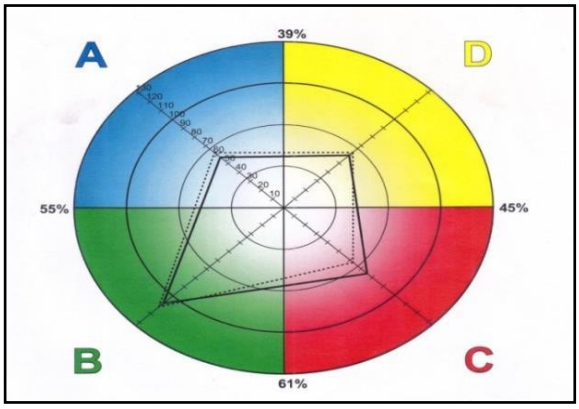

Figure 1. Researcher's Profile

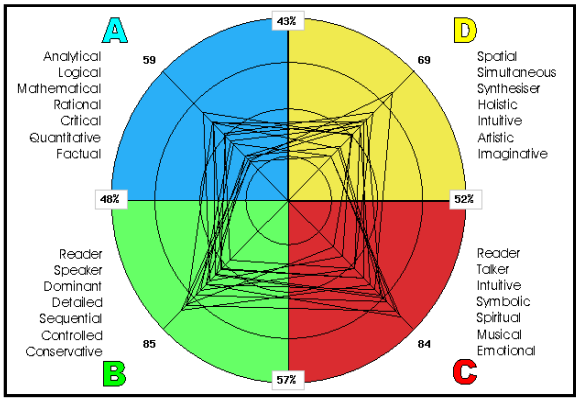

Figure 2. Composite Group of Participants' Preferences

Source: Herrmann International (2016)

Based on her strong preference in B and C, programming facilitation may overlook qualities like creativity, active imagination, and critical thinking as inherent in quadrant A and D. This suggests potential bias in designing learning experiences. Therefore, students' preferences, expectations, and challenges must be understood in order to minimize biases during the learning design process. Notably, 13 participants were randomly selected to complete the validated HBDI survey in the first AR cycle (Figure 2). A Hermann practitioner prepared and interpreted the researcher and participant's profiles for easy understanding. This was used to understand the pre-service teachers' (PSTs) learning preferences and not to evaluate their performance. Therefore, the participants showed strong preferences in B and C, which aligns with the researcher's strong preference. In the second cycle, a simulated HBDI was first used on all 34 students in the programming classroom, and the results showed strongly favored quadrants in B, C, and D, while quadrant A was less represented. Later, nine students' profiles were also determined through the validated HBDI, and their preference for learning programming lies strongly in quadrants A, B, and D. The programming instructions were then designed to accommodate all preferences by adopting a flexible approach in planning and facilitating lessons. This was to ensure engagement from all four brain quadrants throughout the learning process using the programming walk-around (Figure 3).

Instruction was facilitated by first exposing the PSTs to Scratch programming before moving to procedural QBASIC programming.

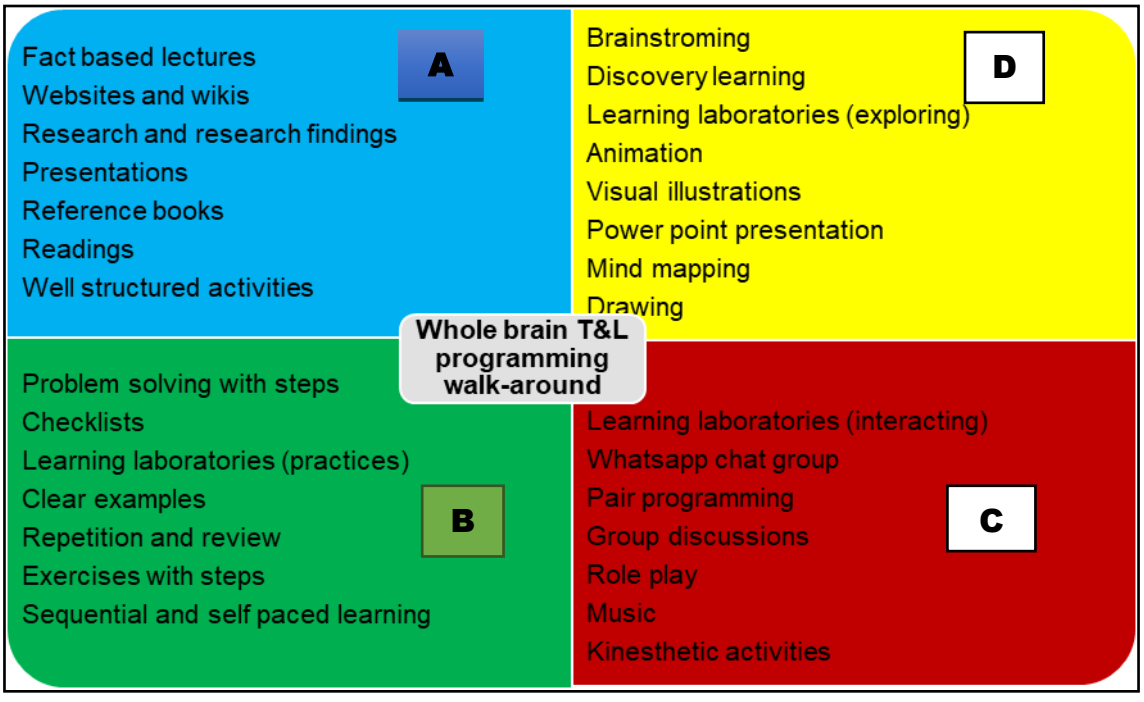

Figure 3. Programming Walk-Around

The instructional plan was conscientiously refined to effectively convey programming concepts over two cycles (see Table 2, adopted from Tijani, 2020). With exposure to diverse programming problems, the instructional plan was adjusted based on the PSTs' feedback, contextual considerations, and the researcher's reflections after each lesson. This plan embraced a holistic methodology, incorporating various learning activities that aligned with the intervention, thereby establishing a unique educational theory applicable in multiple contexts. The PSTs were able to construct their own understanding from the programming activities grounded in constructivist learning activities earlier discussed, and which are aimed at achieving the defined learning objectives, as illustrated in the programming walk-around. To promote collaboration among groups of similar preferences, participants were initially grouped into homogeneous teams of four. After several weeks, these groups were reorganized into heterogeneous groups to encourage interaction with peers with different cognitive preferences, thus fostering skill development in less-favored quadrants. The instructional plan continued to utilize a whole-brain approach, integrating specific learning activities tailored to each cognitive quadrant as needed.

3.5. Validity and Reliability

To ensure validity, a validated HBDI was used to determine the researcher's and some of the learners' preferences for learning, while the simulated HBDI maintains the contents of the validated HBDI. The 24-item simulated HBDI questionnaire was then subjected to a test-retest method through a pilot study on other students that are not participants of the study. The Cronbach-alpha reliability coefficient obtained was 0.87, which was considered reliable for this study.

3.6. Ethics and Quality Assurance

Official permission to conduct the study was obtained from the institution where the data were collected. Anonymity was rigorously safeguarded, and pseudonyms were substituted for the participants' real names. Validity was ensured by allowing participants to respond during interviews such that comprehensive qualitative data were obtained. Their responses were cross-verified for accuracy. Trustworthiness was achieved through triangulation, member checking, analyzing negative cases, and consistently engaging in reflective analysis (Creswell, 2014).

3.7. Data Analysis

Data analysis was conducted using thematic analysis. Lived experiences of the participants in the procedural programming classroom were transcribed, coded, and analyzed using deductive and inductive thematic methods (Braun & Clarke, 2006).

4. FINDINGS

Findings from this study are discussed under two themes: *programming intervention promoted student engagement and pre-service teachers' programming strategies* below.

We first present the learning preferences of the participants in the programming classroom using quantitative data. Table 2 gives a summary of the students' learning preferences in the programming classroom.

Table 2. Students' Learning Preferences in the Programming Classroom

Quadrants	Composite HBDI Quadrants				Simulated HBDI Quadrants					
	A	B	C	D	A	B	C	D	AC	AD
AR Cycle 1	48%	57%	52%	43%	-	-	-	-	-	-
AR Cycle 2	77%	80%	77%	69%	19%	28%	38%	9%	3%	3%

The results indicate that students' preferences for learning programming, as assessed by the HBDI, are: B (57%, 80%), C (52%, 77%), A (48%, 77%), and D (43%, 69%).

Table 2. Sample Programming Instructional Plan

Weeks of Lessons	Block/Text-Based Programming	Topic Taught	Whole Brain Approach	Quadrant	Constructivist Learning Activities in the Classroom
Week 1	Nil	Consent letters, familiarisation, filling of HBDI instrument, and other deliberations	Interaction/discussions	C	Learner control, social interaction.
Week 2	Scratch	Thinking for computers (Algorithms)	Brainstorming, clear examples, lecture and poem	A, B, C and D	Collaborative learning, brainstorming, teacher as facilitator, social interaction, meaning construction, authentic activities, multiple perspectives, situated learning, real-life examples, group learning.
Week 5	Scratch	Algorithm with decision and repetition	Presentation, brainstorming, self-paced, group discussion, exercises with steps, feedback on assignment	B, C and D	Collaborative learning, authentic activities, teacher as facilitator, social interaction, meaning construction, multiple perspectives, problem solving, previous knowledge construction, real-life problems, individual learning, group learning, rubric, cognitive conflict, learner control.
Week 11	Scratch	Repetition	Powerpoint presentation, practical practice (interacting), group discussions, pair programming, animations	B, C and D	Collaborative learning, authentic activities, teacher as facilitator, social interaction, meaning construction, multiple perspectives, problem solving, previous knowledge construction, meaning construction, reflective teaching.
Week 13	Scratch	Variables	Power point presentation, animations, practical (exploring and interacting), group discussions, visual illustrations, pair programming	C and D	*As L7 above* with error consideration.
Week 16	QBASIC	Writing of small program (control structures)	Lecture, practical (practice), group work, sequential exercises	A and B	* As Week 5 above
Week 18	QBASIC	Looping	Lecture, practical (practice), sequential exercises, discussions	A, B and C	* As Week 5 above

Simulated HBDI shows preference C (38%), and B (28%) are mildly present, while other quadrants are very small in percentage, indicating distributed preferences among the four quadrants. In addition, the results from the simulated HBDI tool confirmed the dominance of quadrants B and C among programming students. One important point regarding PSTs' preferences for learning is that the B and C quadrants are overly represented in the two action research cycles. This suggests that the programming class is predominantly dominated by PSTs who prefer organized, sequential, planned, detailed explanations of programming concepts, interpersonal, emotional, feeling-based, and kinesthetic activities.

4.1. Programming Intervention Promoted Student Engagement

This theme focuses on the PSTs' perspectives on how the programming intervention promoted engagement, as explained under two categories: whole brain grouping enhanced learning, and constructivist teaching strategies encouraged engagement in learning.

The participants highlighted that the adoption of whole brain quadrant grouping significantly contributed to their learning experience. They pointed out that the instructor employed diverse approaches to convey the programming concepts effectively. Techniques such as dramatization, role-playing, and poetry served as instructional tools, aiding the activation of the students' mental frameworks for challenging programming ideas. In addition, incorporating class presentations facilitated an engaged learning atmosphere for programming. Participants emphasized that the utilization of whole brain grouping not only enhanced their understanding of programming but also granted them deeper self-awareness. The subsequent excerpts extracted from interviews and students' reflective learning journals corroborate this finding:

> *"I was able to mix with my friend …even the way I can't learn I was able to learn in that way…" (Vanessa).*
>
> *"I can say that also what helped me is the A aspect, and the B aspect…" (Joy)*
>
> *"I came to know about some of my weaknesses and what am very good at…"* (Farai)

Constructivist teaching methods, such as collaborative discussions and brainstorming within quadrant groups and the jigsaw method, further enriched the classroom environment. Consequently, through collective efforts, the PSTs found the class captivating due to the engaging group interactions and debates inherent in the learning process. In addition, the jigsaw enabled them to maintain interest and easily address problems within their peer groups, reducing the need to consult the instructor frequently. Notably, both the educator and students participated in scaffolding during programming instruction. This collaborative support structure allowed proficient students to assist those struggling, facilitating their progress beyond their comfort zone. Similarly, through collaborative effort, the participants engaged in social interactions while working within groups during programming tasks. They fostered a curious disposition in the programming classroom, driven by discussions between the facilitator and the PSTs. This enabled connections and open communication where introverted individuals acquired the ability to engage socially and enhance their capacity to explain ideas during programming tasks. To validate these observations, participants provided the following perspectives:

> "… *has made me to improve both socially and mentally*" (Joy).
>
> *"I learnt and understand … and relate (ing) with others help in learning greatly" (Bassey);*
>
> *"I gain a lot about this five-repetition structure in flowchart because the class was interactive" (Mercy).*

To further enhance programming learning, research activities were introduced and served as a means for students to delve into the realms of the world from their viewpoints. This process introduced diverse experiences into the classroom environment, leading to comprehensive discussions aimed at collectively constructing knowledge. For example, "… *when we did research, it was interesting, it was inviting*" (Percy). Therefore, constructivist strategies introduced in the programming class promoted engagement where students were actively involved and committed to learning programming.

Further examination delved into the influence of group work and pair programming. Quadrant grouping was both advantageous and adverse. The study's findings revealed a mixed outcome, where some groups collaborated effectively, while others encountered difficulties, particularly during project assignments. Participants expressed reservations about the impact of grouping on their learning experiences. They elaborated on the challenges arising from the reshuffling of students from distinct quadrants, having been accustomed to peers possessing similar preferences. This reshuffling, they noted, led to disruptions as the groups became composed of students with diverse cognitive preferences, hampering effective communication. In affirmation of these findings, Peter, and Nancy commented:

"*I discovered that in my group, there are these set of people, I don't know maybe their quadrant is A, …*

but the way they came up with solution is quite different. So, I find it difficult coping with them…"

Scratch programming is ineffective among group members".

The disposition and conduct of group participants also had an adverse influence on the programming learning process. Within groups, there were instances where some members relinquished their responsibilities to others, while certain individuals completed assignments in isolation from their group peers. Furthermore, students often found it challenging to express themselves openly within groups due to the dominating behavior of certain members. Consequently, the dynamics of grouping led to internal struggles, resulting in discontent among some members. Notwithstanding the documented negative effects of grouping, students managed to develop an adaptive approach by gradually adjusting to the diverse attitudes and behaviors of fellow group members. In contrast to group learning, pair programming was preferred. Drawing from their viewpoints, they noted that pair programming promoted collaboration by involving only two participants rather than four. This encouraged unrestricted expression and a sense of accountability for individual actions. Students found pair programming intriguing and expressed a preference for it over participating in larger groups. These excerpts from interviews provide validation for these findings:*" … after some practice we adapted to each other, we worked together" (Joyce); "I prefer pair than group…I know how I will express myself more than when we are like four…".*

4.2. Pre-Service Teachers' Programming Strategies

This theme explains the strategies employed by PSTs in learning programming as unpacked under two categories viz-a-viz problem-solving and self-regulation of learning.

Using *problem-solving*, PSTs employed their mental processes to tackle challenges as part of their programming concept learning. This strategy encompassed brainstorming and the creation of effective group interactions during programming tasks, which was evident when tackling assignments like the multiplication table problem. Beyond employing brainstorming and group dynamics, students highlighted that their consistent engagement in problem-solving activities within the programming classroom fostered their self-confidence in addressing complex programming assignments. This competence, in turn, enabled them to assist their peers and further nurtured a heightened sense of responsibility toward their programming education. To illustrate this point, one participant shared: "*because I was opportune to solve problems, if I get home, I will do my assignments*" (Mercy).

Self-regulated learning refers to students managing their thoughts, emotions, and actions in relation to achieving their goals (Schunk, 2020). Through *self-regulation,* they established objectives for overseeing their programming learning process through a range of methods, including seeking assistance from both their instructor and peers, as well as demonstrating a commitment to learning. This commitment was evident in actions such as raising their hand to seek clarification when faced with a challenging concept, waiting after class to finish assignments, and ensuring the completion of a task. The following excerpts, extracted from classroom observations and students' reflective journals, corroborate and reinforce the finding.

"Farai assisted Alfred and his partner. Queen also assisted students in group E, and they were happy when the program was able to run" (classroom observation). ,

"*… as we are in a group, we used to help ourselves*" (Uchenna).

5. DISCUSSION

The study examined one research question: how can whole brain teaching infused with constructivist teaching strategies be integrated into a pre-service college students' programming education to address diverse learner preferences, towards enhancing their knowledge of procedural programming?

Findings through qualitative hermeneutic data analysis from two cycles of action research generated two themes from the analysed data: *programming intervention encouraged student engagement,* and *pre-service teachers' programming strategies*.

First, the study revealed that students exhibited brain activity in all four quadrants, thus challenging the prevailing notion that students who learn programming are solely left-brained. This finding supported Shin et al. (2022), who stressed engaging students in the whole-brain activity. Second, the study's outcomes also indicated that constructivist strategies encouraged learning engagement in the programming classroom among the pre-service teachers (*theme 1*). Constructivist strategies were instrumental in promoting student engagement in programming. Johnson & Johnson (2018) stressed that *engagement* is realized when students demonstrate a dedicated effort, genuine interest, and concentration in completing a task, while applying cognitive reasoning to accomplish their objectives. This was demonstrated among the pre-service teachers as groups were actively engaged in classroom participation through group discussions, aligning with the findings of Tan et al (2022) and Lubarda et al. (2024). Collaboration and metacognitive awareness in the learning process (Johnson & Johnson, 2018) were exhibited among group members of differing learning preferences. Learning programming led to enhanced decision-making and cognitive reasoning within groups (Cecchini et al., 2020). However, the heterogeneous composition of groups, with varying learning preferences, sometimes led to distractions, confusion, conflicts, and reduced cooperation, which concurred with Lee et al. (2018). This negatively impacted the understanding of the algorithm and pseudocode in some students. These findings are in line with existing literature stating that cooperative grouping does not always benefit all students (Healy et al., 2018). While some students supported the grouping approach, others felt it hindered their active involvement and generated confusion within groups, resulting in a lack of cooperation. It is believed that programming instruction involving a complete whole brain grouping from the commencement of the lessons in conjunction with the programming walk-around, might prove a better experience. Nevertheless, the pre-service teachers favoured pair programming over grouping because it fostered cooperation, accountability, and respect for individual perspectives, as found by Ma et al (2023) and Lubarda et al (2024). This suggests that learning through paired programming and group activities could promote knowledge construction (Su et al., 2024) through deep social interactions, inquisitive and reflective mindset. This approach proved beneficial for introverted students. While pair programming was advantageous, the effectiveness of PP for comprehension relies heavily on the individual's participation. The actions of a distracted partner may hinder the progress of a dedicated partner. Although this behavior was not specifically noted by the participants, it presents an opportunity for exploration in future research.

Furthermore, findings demonstrated that students exhibited diverse programming strategies for learning (*theme 2*). Students exposed to diverse programming problems gained self-confidence in individual *problem-solving* and by transferring skills from visual representations in Scratch to text-based procedural programming. The result resonated with Grover et al (2015) and Olsson & Granberg (2024). The pre-service teachers also developed problem-solving strategies, including analogical thinking, brainstorming, argumentation, and devising group dynamics, aligning with Hazzan et al (2015). Moreover, social interaction through group discussions facilitated knowledge construction, and metacognitive and cognitive strategies to oversee, control, and adjust their learning processes throughout the programming course (Silver et al, 2024). They placed significant importance on mastering programming by challenging themselves with more complex tasks and organizing group tutorials independently (Çakıroğlu & Öztürk, 2017). They demonstrated a clear focus on goal achievement, actively sought assistance, applied knowledge across different learning areas, maintained a positive emotional perspective, and embraced a comprehensive understanding of programming. Self-regulated learning refers to the ability of students to manage their thoughts, emotions, and actions in relation to achieving their goals (Schunk, 2020). The finding established that pre-service teachers' *self-regulatory* practices enhanced their programming learning experience, which corroborates with previous studies of Lee et al. (2010) and Shin et al. (2022).

6. CONCLUSION

This study underscored the effectiveness of a holistic teaching intervention in programming education, leveraging WBTL, cooperative learning, instructional scaffolding, and collaboration. It advances the understanding of pre-service teachers' learning preferences and confirms the effectiveness of the intervention in creating engagement in first-year pre-service teachers' knowledge of programming from the visual to the procedural. In addition, students learned programming through problem-solving strategies and self-regulation. Nonetheless, the study acknowledged the complexities of group dynamics and the need for tailored strategies to accommodate varying learning diversities present in programming classrooms, thereby fostering inclusive and effective programming education. The study made two contributions to research. Theoretically, this study contributes to the existing literature on learning strategies, including problem-solving and self-regulated learning, which are critical for programming education. It also provides empirical support for research advocating constructivist teaching strategies in programming instruction. The instructional plan developed in this study (Table 2) serves as a tangible research outcome, further enriching the literature on instructional design for programming courses. Moreover, given the limited research on the application of whole-brain theory in programming instruction, this study addresses a notable gap. Considering the enhancement of both teaching practices and student learning in programming, the theoretical contributions of this study are particularly relevant, especially since the instruction of programming to first-year pre-service teachers in Nigeria has received limited attention. Practically, programming facilitators can utilise the simulated HBDI to understand learners' preference for learning programming and use this understanding to design instruction based on the whole-brain programming walk-around (Figure 3) and the instructional plan (Table 2). We therefore recommend the use of a programming walk-around in conjunction with the instructional plan to be adapted or adopted by programming educators for designing programming lessons that meet diverse students' needs. This, we believe, will encourage engagement in learning programming and enable students to exhibit individual strategies for learning programming, which may therefore change the narrative that programming is hard and difficult to understand. Due to the non-generalisability of action research studies, the findings of this study cannot be generalized but can be replicated in another context. Further exploration with larger sample sizes and diverse contexts could bolster the robustness and generalizability of these findings.

Acknowledgment. We thank the management of Michael Otedola College of Primary Education (now Lagos State University of Education), Lagos State, Nigeria, for providing the necessary facilities and resources to conduct the research. Likewise, students of the institution during the 2015/2016 and 2016/2017 academic sessions are appreciated for being part of the research. Special acknowledgement also goes to the supervisors of the research at the PhD level.

Research Ethics. This study received ethics approval on 18th April, 2016, from the Faculty of Education, University of Pretoria, South Africa, with reference number SM 15/11/02, and all procedures for data collection were in accordance with the applicable ethics laws of the institution.

Data Availability Statement. All data can be obtained from the corresponding author via email.

Conflicts of Interest. No conflict of interest was recorded for the research reported in this manuscript.

Funding. The study received the Tertiary Trust Fund (TETFUND) funding.

REFERENCES

Ahmadi S. M. & Motaghi, F. (2021). Cognitive vs metacognitive scaffolding strategies and EFL learners' listening comprehension development. *Language Teaching Research*. https://doi.org/10.1177/13621688211021821

Angeli, C. (2022). The effects of scaffolded programming scripts on pre-service teachers' computational thinking: Developing algorithmic thinking through programming robots. *International Journal of Child-Computer Interaction*, *31*(100329). https://doi.org/10.1016/j.ijcci.2021.100329

Bawaneh, A. K. A., Abdullah, A. G. K., Saleh, S., & Yin, K. Y. (2011). Jordanian students' thinking styles based on Herrmann whole brain model. *International Journal of Humanities and Social Science*, *1*(9), 89–97.

Bawaneh, A. K. A., Nurulazam Md Zain, A., & Salmiza, S. (2011). The Effect of Herrmann Whole Brain Teaching Method on Students' Understanding of Simple Electric Circuits. *European Journal of Physics Education*, *2*(2), 1–23.

Belland, B. R. (2017). *Instructional scaffolding in STEM education: Strategies and efficacy evidence.* Springer.

Bennedsen, J. & Caspersen, M. E. (2019). Failure rates in introductory programming: 12 years later. *ACM Inroads*, *10(2)*, 30–36. https://doi.org/10.1145/3324888

Berssanette, J. H., & de Francisco, A. C. (2021). Active learning in the context of the teaching/learning of computer programming: A systematic review. *Journal of Information Technology Education. Research*, *20*(201). http://dx.doi.org/10.28945/4767

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, *3*(2), 77–101. https://doi.org/10.1191/1478088706qp063oa

Çakıroğlu, U & Öztürk, M. (2017). Flipped Classroom with Problem Based Activities: Exploring Self-regulated Learning in a Programming Language Course. *Journal of Educational Technology & Society*, *20*(1), 337–349. http://www.jstor.org/stable/jeductechsoci.20.1.337

Cecchini, J. A, Fernandez-Rio, J, Mendez-Gimenez, A, Gonzalez, C., Sanchez-Martínez, B, & Carriedo, A. (2020). High versus low-structured cooperative learning. Effects on prospective teachers' regulation dominance, motivation, content knowledge and responsibility. *Journal of Teacher Education*, *44*(4), 486–501. https://doi.org/10.1080/02619768.2020.1774548

Cheah, C. S. (2020). Factors Contributing to the Difficulties in Teaching and Learning of Computer Programming: A Literature Review. *Contemporary Educational Technology*, *12*(2). https://doi.org/10.30935/cedtech/8247

Coffield, F., Ecclestone, K., Hall, E., & Moseley, D. (2004). *Learning styles and pedagogy in post-16 learning: A systematic and critical review*. Learning and Skills Research Centre.

Creswell, J. W. (2014). *A concise introduction to mixed methods research*. SAGE.

Dakhel, A. M, Majdinasab, V, Nikanjam, A., Khomh, F, Desmarais, M. C, & Jiang, Z. M. J. (2023). Github copilot AI pair programmer: Asset or liability?. *Journal of Systems and Software, 203*.

Dasgupta, S., & Hill, B. M. (2017). *Learning to code in localized programming languages*. Proceedings of the Fourth (2017) ACM Conference on Learning (pp.33–39). https://doi.org/10.1145/3051457.3051464

De Boer, A. L, Du Toit, P, Scheepers, D., & Bothma, T. (2013). *Whole Brain® Learning in higher education. Evidence-based practice.* Elsevier.

Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education*, *58*(1), 240–249. https://doi.org/10.1016/j.compedu.2011.08.006

Du Toit, P. (2009). *An action research approach to monitoring one's professional development as manager.* Foundation for Professional Development.

Elfiky, D. E. G. (2022). The Effect of a Whole Brain Teaching Based Instruction on Developing Number Competencies and Arithmetic Fluency in Kindergarten Children. *International Journal of Instruction*, *15*(1), 673–684. https://doi.org/10.29333/iji.2022.15138a

Entwistle, N., Hughes, J. C., & Mighty, J. (2010). Taking stock: An overview of research findings. *Research on Teaching and Learning in Higher Education*, 15–51.

Entwistle, N., McCune, V., & Hounsell, J. (2002). *Approaches to studying and perceptions of university teaching-learning environments: Concepts, measures and preliminary findings* (No. Occasional report; 1, pp. 1–9).

Garcia, M. B. (2021). Cooperative learning in computer programming: A quasi-experimental evaluation of Jigsaw teaching strategy with novice programmers. *Education and Information Technologies*, *26*(4), 4839–4856. https://doi.org/10.1007/s10639-021-10502-6

Ghavifekr, S. (2020). Collaborative Learning: A Key to Enhance Students' Social Interaction Skills. *MOJES: Malaysian Online Journal of Educational Sciences*, *8*(4), 9–21.

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, *25*(2), 199–237. https://doi.org/10.1080/08993408.2015.1033142

Hawlitschek, A., Köppen, V., Dietrich, A., & Zug, S. (2020). Drop-out in programming courses–prediction and prevention. *Journal of Applied Research in Higher Education*, *12*(1), 124–136. http://dx.doi.org/10.1108/JARHE-02-2019-0035

Healy, M., Doran, J., & McCutcheon, M. (2018). Cooperative learning outcomes from cumulative experiences of group work: Differences in student perceptions. *Accounting Education*, *27*(3), 286–308. http://dx.doi.org/10.1080/09639284.2018.1476893

Herrera-Pavo, M. Á. (2021). Collaborative learning for virtual higher education. *Learning, Culture and Social Interaction*, *28*, 100437. https://doi.org/10.1016/j.lcsi.2020.100437

Herrmann, N. (1995). *The creative brain.* Quebecor Printing Book Group.

Herrmann, N. (1996). *The whole brain business book.* McGraw Hill

Hou, X., Ericson, B. J., & Wang, X. (2022). Using adaptive parsons problems to scaffold write-code problems. *ACM Conference on International Computing Education Research,* 15–26. http://dx.doi.org/10.1145/3501385.3543977

Isong, B. (2014). A Methodology for Teaching Computer Programming: First year students' perspective. *International Journal of Modern Education and Computer Science*, *6*(9), 15. https://doi.org/10.5815/ijmecs.2014.09.03

Jakovljevic, M. (2003). *Concept mapping and appropriate instructional strategies in promoting programming skills of holistic learners*. 308–315. https://dl.acm.org/doi/10.5555/954014.954048

Johnson, D. W., & Johnson, R. T. (2018). Cooperative learning: The foundation for active learning. *Active Learning—Beyond the Future*, 59–71. https://doi.org/10.5772/intechopen.81086

Johnson, D. W., & Johnson, R. T. (2021). Learning together and alone: The history of our involvement in cooperative learning. In *Pioneering perspectives in cooperative learning* (pp. 44–62). Routledge.

Kim, C., Vasconcelos, L., Belland, B. R., Umutlu, D., & Gleasman, C. (2022). Debugging behaviors of early childhood teacher candidates with or without scaffolding. *International Journal of Educational Technology in Higher Education*, *19*(1), 26. http://dx.doi.org/10.1186/s41239-022-00319-9

Kim, N. J., Belland, B. R., Lefler, M., Andreasen, L., Walker, A., & Axelrod, D. (2020). Computer-based scaffolding targeting individual versus groups in problem-centered instruction for STEM education: Meta-analysis. *Educational Psychology Review*, *32*, 415–461. https://doi.org/10.1007/s10648-019-09502-3

Kirstein, M., & Kunz, R. (2016). A whole brain® learning approach to an undergraduate auditing initiative–An exploratory study. *Meditari Accountancy Research*, *24*(4), 527–544. https://doi.org/10.1108/MEDAR-02-2014-0029

Klein, H. K., & Myers, M. D. (1999). A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Quarterly*, *23*(1), 67–93. https://doi.org/10.2307/249410

Koulouri, T., Lauria, S., & Macredie, R. D. (2016). Do (and say) as I say: Linguistic adaptation in human–computer dialogs. *Human–Computer Interaction*, *31*(1), 59–95. https://doi.org/10.1080/07370024.2014.934180

Law, K. M., Lee, V. C., & Yu, Y.-T. (2010). Learning motivation in e-learning facilitated computer programming courses. *Computers & Education*, *55*(1), 218–228. https://doi.org/10.1016/j.compedu.2010.01.007

Le Roux, I. (2011). New large class pedagogy: Developing students' whole brain thinking skills. *Procedia-Social and Behavioral Sciences*, *15*, 426–435. https://doi.org/10.1016/j.sbspro.2011.03.116

Lee, D., Morrone, A. S., & Siering, G. (2018). From swimming pool to collaborative learning studio: Pedagogy, space, and technology in a large active learning classroom. *Educational Technology Research and Development*, *66*, 95–127. https://doi.org/10.1007/s11423-017-9550-1

Lee, H., Han, Y.T, & Chen, M.P. (2010). The effect of scaffolding support on programming performance and the use of self-regulation in learning computer programming. *Enhancing and Sustaining New Knowledge Through the Use of Digital Technology in Education,* 517–524. https://doi.org/10.1108/ITSE-04-2021-0074

Lubarda, M., Phan, A., Schurgers, C., Delson, N., Ghazinejad, M., Baghdadchi, S., Minnes, M., Kim, M., Pilegard, C., Relaford-Doyle, J., Sandoval, C. L., & Qi, H. (2024). Virtual pair programming and online oral exams: Effects on social interaction, performance, and academic integrity in a remote computer programming course. *Computer Science Education*, 1–41. https://doi.org/10.1080/08993408.2024.2344401

Luna Scott, C. (2015). *The futures of learning 2: What kind of learning for the 21st century?* UNESCO.

Ma, Q., Wu, T., & Koedinger, K. (2023). Is AI the better programming partner? Human-human pair programming vs. Human-ai pair programming. *arXiv Preprint arXiv:2306.05153*.

Malliarakis, C., Satratzemi, M., & Xinogalos, S. (2013). A holistic framework for the development of an educational game aiming to teach computer programming. *University of Macedonia*, 359–369.

Margulieux, L. E., & Catrambone, R. (2021). Scaffolding problem solving with learners' own self explanations of subgoals. *Journal of Computing in Higher Education*, *33*(2), 499–523. https://doi.org/10.1007/s12528-021-09275-1

McLoughlin, C., & Lee, M. J. (2008). The three p's of pedagogy for the networked society: Personalization, participation, and productivity. *International Journal of Teaching and Learning in Higher Education*, *20*(1), 10–27.

Medeiros, R. P., Ramalho, G. L., & Falcão, T. P. (2018). A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*, *62*(2), 77–90. https://doi.org/10.1109/TE.2018.2864133

Näykki, P., Isohätälä, J., & Järvelä, S. (2021). "You really brought all your feelings out"–Scaffolding students to identify the socio-emotional and socio-cognitive challenges in collaborative learning. *Learning, Culture and Social Interaction*, *30*, 100536. https://doi.org/10.1016/j.lcsi.2021.100536

Ngozo, B. P. (2012). *Dynamics of learning style flexibility in teaching and learning*. University of Pretoria (South Africa).

Olsson, J., & Granberg, C. (2024). Teacher-student interaction supporting students' creative mathematical reasoning during problem solving using Scratch. *Mathematical Thinking and Learning*, *26*(3), 278–305. https://doi.org/10.1080/10986065.2022.2105567

Opdecam, E., Everaert, P., Van Keer, H., & Buysschaert, F. (2014). Preferences for team learning and lecture-based learning among first-year undergraduate accounting students. *Research in Higher Education*, *55*, 400–432. https://doi.org/10.1007/s11162-013-9315-6

Pritchard, A. (2017). *Ways of learning: Learning theories for the classroom*. Routledge.

Qureshi, M. A., Khaskheli, A., Qureshi, J. A., Raza, S. A., & Yousufi, S. Q. (2023). Factors affecting students' learning performance through collaborative learning and engagement. *Interactive Learning Environments*, *31*(4), 2371–2391. https://doi.org/10.1080/10494820.2021.1884886

Saka, A. O. (2020). Learning to Write Programs using Think-Pair-Share Programming Strategy: What are the Students' Perceptions and Experiences? *Journal of Educational Sciences*, *4*(4), 705–717. https://doi.org/10.31258/jes.4.4.p.705-717

Santoso, D. (2016). Improving the Students' Spiritual Intelligence in English Writing through Whole Brain Learning. *English Language Teaching*, *9*(4), 230–238. https://doi.org/10.5539/elt.v9n4p230

Schunk, D. H. (2020). *Learning theories: An educational perspective*. Pearson.

Sharma, H. L., & Saarsar, P. (2018). TPS (think-pair–share): An effective cooperative learning strategy for unleashing discussion in classroom interaction. *International Journal of Research in Social Sciences*, *8*(5), 91–100.

Shin, D. D., Lee, M., & Bong, M. (2022). Beyond left and right: Learning is a whole-brain process. *Theory into Practice*, *61*(3), 347–357. https://doi.org/10.1080/00405841.2022.2096386

Simon, B., & Hanks, B. (2008). First-year students' impressions of pair programming in CS1. *Journal on Educational Resources in Computing, 7*(4), 1–28. https://doi.org/10.1145/1316450.1316455

Sontillano, R. D. (2018). Impact of whole brain teaching based instruction on academic performance of grade 8 students in Algebra: Compendium of WBT-based lesson plans. *International Journal of Teaching, Education and Learning*, *2*(2), 98–114. https://doi.org/10.20319/pijtel.2018.22.98114

Su, Y.-S., Wang, S., & Liu, X. (2024). Using Epistemic Network Analysis to Explore Primary School Students' Computational Thinking in Pair Programming Learning. *Journal of Educational Computing Research*, *62*(2), 559–593. https://doi.org/10.1177/07356331231210560

Tan, S. C., Lee, A. V. Y., & Lee, M. (2022). A systematic review of artificial intelligence techniques for collaborative learning over the past two decades. *Computers and Education: Artificial Intelligence*, *3*, 100097. https://doi.org/10.1016/j.caeai.2022.100097

Thota, N., & Whitfield, R. (2010). Holistic approach to learning and teaching introductory object-oriented programming. *Computer Science Education*, *20*(2), 103–127. https://doi.org/10.1080/08993408.2010.486260

Tijani, F., Callaghan, R., & de Villers, R. (2020). An investigation into pre-service teachers' experiences while transitioning from Scratch programming to procedural programming. *African Journal of Research in Mathematics, Science and Technology Education*, *24*(2), 266–278. https://doi.org/10.1080/18117295.2020.1820798

Tikva, C., & Tambouris, E. (2023). The effect of scaffolding programming games and attitudes towards programming on the development of Computational Thinking. *Education and Information Technologies*, *28*(6), 6845–6867. https://doi.org/10.1007/s10639-022-11465-y

Topalli, D., & Cagiltay, N. E. (2018). Improving programming skills in engineering education through problem-based game projects with Scratch. *Computers & Education*, *120*, 64–74. https://doi.org/10.1016/j.compedu.2018.01.011

Van Laar, E., Van Deursen, A. J., Van Dijk, J. A., & De Haan, J. (2020). Determinants of 21st-century skills and 21st-century digital skills for workers: A systematic literature review. *Sage Open*, *10*(1), 2158244019900176. http://dx.doi.org/10.1177/2158244019900176

Warsah, I., Morganna, R., Uyun, M., Afandi, M., & Hamengkubuwono, H. (2021). The impact of collaborative learning on learners' critical thinking skills. *International Journal of Instruction*, *14*(2), 443–460. https://doi.org/10.29333/iji.2021.14225a

Watkins, K. Z., & Watkins, M. J. (2009). Towards minimizing pair incompatibilities to help retain under-represented groups in beginning programming courses using pair programming. *Journal of Computing Sciences in Colleges*, *25*(2), 221–227.

Yurdugül, H., & Aşkar, P. (2013). Learning programming, problem solving and gender: A longitudinal study. *Procedia-Social and Behavioral Sciences*, *83*, 605–610. https://doi.org/10.1016/j.sbspro.2013.06.115

Zhang, J.-H., Meng, B., Zou, L.-C., Zhu, Y., & Hwang, G.-J. (2023). Progressive flowchart development scaffolding to improve university students' computational thinking and programming self-efficacy. *Interactive Learning Environments*, *31*(6), 3792–3809. http://dx.doi.org/10.1080/10494820.2021.1943687