



American Journal of Development Studies and Innovations (AJDSI)

VOLUME 1 ISSUE 1 (2026)



PUBLISHED BY
E-PALLI PUBLISHERS, DELAWARE, USA

Effects of Python-Integrated Mathematics Instruction on Mathematics Achievement, Mathematical Problem-Solving Skills, and Computational Thinking among Senior Secondary School Students in Ondo State, Nigeria

Olurotimi David ADULOJU^{1*}, OluKayode Solomon ABODERIN², Adesoji Olubunmi OMONIYI¹, Raphael Olufemi AKINYEDE³

Article Information

Received: July 07, 2026

Accepted: September 01, 2026

Published: October 22, 2026

Keywords

Computational Thinking, Mathematics Achievement, Mathematical Problem-Solving, Mathematics Education, Python-Integrated Mathematics Instruction, Secondary School Students, Technology-Integrated Learning

ABSTRACT

This study investigated the effect of Python-integrated Mathematics instruction on Mathematics achievement, mathematical problem-solving skills and computational thinking skills among Senior Secondary School Two (SS2) students in Ondo State, Nigeria. Six research questions and six null hypotheses guided the study. A quasi-experimental pre-test-post-test non-equivalent control group design was adopted. The sample comprised 160 SS2 students, with 80 students assigned to the Python-integrated Mathematics group and 80 to the conventional Mathematics instruction group. Data were collected using the Mathematics Achievement Test (MAT), Mathematical Problem-Solving Test (MPST) and Computational Thinking Skills Test (CTST), with reliability coefficients of 0.80, 0.78, and 0.82, respectively. Mean and standard deviation were used to answer the research questions, while independent-samples t-test and Analysis of Covariance (ANCOVA) were used to test the hypotheses at the 0.05 level of significance. The baseline results showed no significant differences between the groups in Mathematics achievement, $t(158) = 0.845$, $p = .399$; mathematical problem-solving, $t(158) = 0.681$, $p = .497$; and computational thinking skills, $t(158) = 0.678$, $p = .499$. Following the intervention, Python-integrated Mathematics instruction had significant effects on Mathematics achievement, $F(1,157) = 132.29$, $p < .001$, partial $\eta^2 = .457$; mathematical problem-solving, $F(1,157) = 127.75$, $p < .001$, partial $\eta^2 = .449$; and computational thinking skills, $F(1,157) = 174.84$, $p < .001$, partial $\eta^2 = .527$. The study concluded that Python-integrated Mathematics instruction significantly enhanced the three measured outcomes. It was recommended that Mathematics teachers integrate appropriate Python-based activities into Mathematics instruction and that schools provide adequate technological resources and teacher preparation.

INTRODUCTION

Background to the Study

Mathematics is an important component of contemporary education because it provides conceptual and analytical foundations for science, technology, engineering, economics and everyday decision-making. Mathematics education involves not only numerical and procedural knowledge but also reasoning, problem formulation, identification of relationships, strategic thinking, evaluation of solutions and communication of mathematical ideas. These competencies are increasingly relevant in learning environments shaped by digital technologies, where mathematical reasoning can interact with computational processes. The increasing use of digital technologies in education has encouraged interest in instructional approaches that engage students in constructing and applying mathematical knowledge. Conventional mathematics instruction can support explanation of concepts and demonstration of procedures; however, when students primarily reproduce established procedures, opportunities to explore relationships, test strategies and apply mathematical ideas to unfamiliar situations may be reduced. This has encouraged

attention to approaches that provide students with more opportunities for active mathematical exploration. The integration of programming into mathematics instruction is one such approach. Programming has conceptual connections with mathematical reasoning and problem-solving because it requires learners to translate ideas into explicit procedures, decompose tasks, sequence operations, test solutions, identify errors and revise procedures. Recent mathematics-education research has consequently examined programming as a possible medium for mathematical learning. Munthe and Naalsund (2024), for example, developed mathematical programming problems in which programming was used to facilitate mathematical learning. Their work employed Python at the secondary-school level and focused on mathematical problem-solving, exploration and conceptual understanding. Tossavainen *et al.* (2024) similarly examined upper-secondary students' exploration of the definite integral in a Python environment. Their findings indicated that students used programming and computational representations in processes involving mathematical thinking, collaboration and communication. Such findings suggest that programming can complement

¹ Department of Science Education, Faculty of Education, Adekunle Ajasin University, Akungba -Akoko, Ondo State, Nigeria

² University of South Africa, Pretoria, South Africa

³ Department of Information Systems, School of Computing, Federal University of Technology, Akure (FUTA), Akure, Ondo State, Nigeria

* Corresponding author's e-mail: davidolurotimi@gmail.com

conventional representations by enabling students to represent mathematical ideas computationally and examine relationships through numerical and visual outputs.

The connection between programming and mathematics is also relevant to computational thinking. Computational thinking includes processes such as decomposition, abstraction, algorithmic thinking, pattern recognition and systematic problem-solving. Several of these processes correspond with stages of mathematical problem-solving, in which students interpret problems, identify relevant information, formulate strategies, implement procedures and evaluate solutions. Mathematics instruction that incorporates programming may therefore provide opportunities for students to engage with mathematical and computational processes within related learning tasks. Programming can make some aspects of mathematical problem-solving more explicit. When students develop a program for a mathematical task, they must specify operations, organize them sequentially, test procedures and determine whether the output corresponds with the mathematical problem. Errors can require debugging and revision, thereby creating opportunities to reconsider solution procedures. Munthe and Naalsund (2024) demonstrated the potential of mathematical programming tasks to expose limitations in students' existing strategies and encourage the development of alternative approaches. However, incorporating programming into mathematics does not in itself guarantee improved mathematical learning. The contribution of programming depends partly on the design of the learning tasks and the relationship between programming requirements and mathematical objectives. Excessive technical demands may divert students' attention from the mathematical concepts being studied. Munthe and Naalsund (2024) therefore emphasized task designs in which mathematics remains the central learning objective while programming functions as a means of exploring mathematical ideas. This distinction is important when considering Python as an instructional medium rather than as an additional subject within mathematics lessons.

Evidence from recent studies indicates that Python can be incorporated into mathematics-learning activities. Rais and Zhao Xuezhi (2024) examined Python programming integration within the Merdeka Belajar curriculum in relation to students' motivation and learning outcomes. Munthe and Naalsund (2024) and Tossavainen *et al.* (2024) also demonstrated applications of Python in secondary or upper-secondary mathematics contexts. Collectively, these studies provide empirical grounds for further investigation of Python-integrated mathematics instruction, while also indicating the importance of appropriate pedagogical design. Despite these studies, evidence remains limited on the combined effects of Python-integrated mathematics instruction on mathematics achievement, mathematical problem-solving skills and computational thinking skills. Existing research has frequently focused on individual outcomes

or examined programming primarily as a computer-science competency. Consequently, there is a need for empirical studies that examine these three outcomes within the same mathematics instructional intervention. This issue is relevant at the senior secondary-school level, where students encounter mathematical content requiring increasing levels of abstraction, reasoning and application. Python-integrated activities can allow students to translate mathematical procedures into algorithms, generate numerical outputs, examine relationships and revise solution strategies. Whether such activities result in measurable differences in students' achievement, problem-solving and computational thinking, however, requires empirical investigation.

The Nigerian context provides a further basis for this investigation. Mathematics is a core component of secondary education and contributes to students' preparation for further study in science, technology, engineering and mathematics-related fields. Research on mathematics learning in Nigeria has continued to examine factors associated with students' performance and approaches for strengthening mathematical competence. For example, Awofala (2017) examined mathematical proficiency among senior secondary-school students in Nigeria. Nigerian research has also considered technology-supported approaches to mathematics instruction, although the effectiveness of such approaches depends on their pedagogical implementation. Despite this body of work, empirical evidence on Python-integrated mathematics instruction in Nigerian secondary schools remains limited, particularly regarding its effects on mathematics achievement, mathematical problem-solving skills and computational thinking skills considered together. Evidence specific to Ondo State is even less developed. The absence of sufficient local evidence makes it difficult to determine whether findings from other educational contexts are applicable to students learning mathematics in Ondo State. This constitutes the specific contextual gap addressed by the present study. The study therefore has theoretical and practical relevance. At the theoretical level, examining the three outcomes within one intervention may contribute to understanding how programming-based mathematical activities relate to mathematical learning and computational thinking. At the pedagogical level, the study may provide evidence on whether Python can be integrated into mathematics lessons while keeping mathematical objectives central. The findings may also assist mathematics teachers and curriculum developers in evaluating the potential use of programming activities in secondary-school mathematics. Python is considered in this study because its syntax and computational capabilities allow students to perform calculations, represent mathematical relationships, generate numerical results and explore mathematical models. Its application in mathematics-learning activities has been documented by Munthe and Naalsund (2024) and Tossavainen *et al.* (2024). Nevertheless, the relevance of Python to mathematics education should be

determined by empirical evidence rather than assumed from its technical capabilities or findings from other contexts. Accordingly, the present study investigates the effect of Python-integrated Mathematics instruction on Mathematics achievement, mathematical problem-solving skills and computational thinking skills among Senior Secondary School Two (SS2) students in Ondo State, Nigeria. By examining these outcomes within the same instructional intervention, the study seeks to determine whether Python-integrated Mathematics instruction produces significant differences in students' Mathematics achievement, mathematical problem-solving skills and computational thinking skills when compared with conventional Mathematics instruction.

Statement of the Problem

Mathematics achievement and students' ability to apply mathematical knowledge to unfamiliar problems remain concerns in secondary-school education. The challenge extends beyond recalling facts and procedures to reasoning, selecting and applying strategies, interpreting mathematical relationships and evaluating solutions. This creates a need for instructional approaches that support mathematical learning while providing opportunities to develop problem-solving and related higher-order competencies.

Programming-based learning has been examined as one possible approach because programming involves processes such as problem decomposition, sequencing, algorithm development, testing, error identification and revision, which have connections with mathematical problem-solving and computational thinking. Recent studies have explored programming as a medium for mathematics learning. Munthe and Naalsund (2024) investigated mathematical programming problems designed to support mathematical learning, while Tossavainen *et al.* (2024) examined students' use of a Python environment in exploring the definite integral. Ye *et al.* (2024) also examined flexibility in programming-based mathematical problem-solving. However, these studies do not provide sufficient evidence on the effects of Python-integrated mathematics instruction on the three learning outcomes examined in the present study. The effectiveness of programming in mathematics learning cannot be inferred solely from the similarities between programming processes and mathematical reasoning. Its contribution may depend on how programming activities are aligned with mathematical objectives and the level of technical demand placed on learners. There is therefore a need for empirical evidence on the effects of a structured Python-integrated mathematics intervention on students' learning outcomes.

A specific gap concerns the simultaneous examination of Mathematics achievement, mathematical problem-solving skills and computational thinking skills within the same instructional intervention. Existing research has commonly examined one or more of these outcomes separately or focused on programming competence and

computational thinking. Consequently, evidence remains limited on whether Python-integrated Mathematics instruction can produce significant differences across these related but distinct outcomes. The evidence is also limited in the Nigerian context. Although technology-supported mathematics instruction has received research attention, studies specifically examining Python-integrated Mathematics instruction among senior secondary-school students remain relatively few. Evidence from Ondo State on its effects on Mathematics achievement, mathematical problem-solving skills and computational thinking skills is particularly limited. This leaves a contextual evidence gap concerning the use of Python as an instructional medium in secondary-school Mathematics.

The problem addressed by this study is therefore the limited empirical evidence on whether Python-integrated Mathematics instruction produces significant differences in Mathematics achievement, mathematical problem-solving skills and computational thinking skills among Senior Secondary School Two (SS2) students in Ondo State, Nigeria, compared with conventional Mathematics instruction. The study addresses this problem by comparing students exposed to Python-integrated Mathematics instruction with those receiving conventional Mathematics instruction, while controlling for their respective pre-test scores.

Purpose of the Study

The purpose of this study is to examine the effect of Python-integrated Mathematics instruction on Mathematics achievement, mathematical problem-solving skills, and computational thinking skills among senior secondary school students in Ondo State, Nigeria. Specifically, the study seeks to

- Determine whether there is a significant difference in the pre-test mean Mathematics achievement scores of students exposed to Python-integrated Mathematics instruction and those exposed to conventional Mathematics instruction;
- Determine whether there is a significant difference in the pre-test mean mathematical problem-solving scores of students exposed to Python-integrated Mathematics instruction and those exposed to conventional Mathematics instruction;
- Determine whether there is a significant difference in the pre-test mean computational thinking skills scores of students exposed to Python-integrated Mathematics instruction and those exposed to conventional Mathematics instruction;
- Determine the effect of Python-integrated Mathematics instruction on students' post-test Mathematics achievement after controlling for their pre-test Mathematics achievement scores;
- Determine the effect of Python-integrated Mathematics instruction on students' post-test mathematical problem-solving skills after controlling for their pre-test problem-solving scores; and
- Determine the effect of Python-integrated

Mathematics instruction on students' post-test computational thinking skills after controlling for their pre-test computational thinking skills scores.

Research Questions

The following research questions were raised for the study

- What difference exists in the pre-test mean Mathematics achievement scores of students exposed to Python-integrated Mathematics instruction and those exposed to conventional Mathematics instruction?
- What difference exists in the pre-test mean mathematical problem-solving scores of students exposed to Python-integrated Mathematics instruction and those exposed to conventional Mathematics instruction?
- What difference exists in the pre-test mean computational thinking skills scores of students exposed to Python-integrated Mathematics instruction and those exposed to conventional Mathematics instruction?
- What effect does Python-integrated Mathematics instruction have on students' post-test Mathematics achievement after controlling for their pre-test Mathematics achievement scores?
- What effect does Python-integrated Mathematics instruction have on students' post-test mathematical problem-solving skills after controlling for their pre-test problem-solving scores?
- What effect does Python-integrated Mathematics instruction have on students' post-test computational thinking skills after controlling for their pre-test computational thinking skills scores?

Research Hypotheses

The following null hypotheses were tested at the 0.05 level of significance

- H₀₁:** There is no significant difference in the pre-test mean Mathematics achievement scores of students exposed to Python-integrated Mathematics instruction and those exposed to conventional Mathematics instruction.
- H₀₂:** There is no significant difference in the pre-test mean mathematical problem-solving scores of students exposed to Python-integrated Mathematics instruction and those exposed to conventional Mathematics instruction.
- H₀₃:** There is no significant difference in the pre-test mean computational thinking skills scores of students exposed to Python-integrated Mathematics instruction and those exposed to conventional Mathematics instruction.
- H₀₄:** There is no significant effect of Python-integrated Mathematics instruction on students' post-test Mathematics achievement after controlling for their pre-test Mathematics achievement scores.
- H₀₅:** There is no significant effect of Python-integrated Mathematics instruction on students' post-test mathematical problem-solving skills after controlling for their pre-test problem-solving scores.
- H₀₆:** There is no significant effect of Python-integrated Mathematics instruction on students' post-test

computational thinking skills after controlling for their pre-test computational thinking skills scores.

Significance of the Study

The study is expected to provide empirical evidence that may be useful to students, Mathematics teachers, teacher educators, curriculum planners, educational administrators, policymakers, educational technology developers, and researchers.

Students may benefit from opportunities to learn Mathematics through activities that integrate mathematical concepts with Python programming. Such activities may enable students to represent mathematical procedures computationally, test solution strategies, interpret outputs and revise solutions. The findings may indicate whether Python-integrated Mathematics instruction is associated with differences in Mathematics achievement, mathematical problem-solving skills and computational thinking skills. Mathematics teachers may benefit from empirical evidence on the use of Python-integrated instruction as a complementary approach to conventional Mathematics teaching. The findings may help teachers understand how programming activities can be aligned with mathematical objectives and used to support problem-solving and computational thinking while maintaining Mathematics as the primary instructional focus.

Teacher educators may find the findings useful in preparing prospective Mathematics teachers for technology-supported instruction. Where supported by the evidence, the findings may inform the inclusion of basic programming and computational thinking activities in Mathematics teacher education and professional development. Curriculum planners may use the findings as empirical information when considering the place of programming and computational thinking within secondary-school Mathematics. The evidence may assist in evaluating whether Python-based activities can be aligned with selected Mathematics content while maintaining the intended mathematical learning objectives.

Educational administrators may find the findings useful when considering the conditions required to implement Python-integrated Mathematics instruction. These may include access to appropriate computers and software, teacher preparation, instructional time and technical support. The findings may therefore inform planning and resource decisions concerning technology-supported Mathematics instruction. Educational policymakers may draw on the findings as empirical evidence in discussions concerning digital competence, programming, computational thinking and technology-supported Mathematics education in Nigerian secondary schools. The findings may contribute to policy considerations regarding the integration of computational practices into Mathematics education where such integration is consistent with curriculum objectives and available educational resources.

Educational technology developers may use the

findings to inform the design of digital Mathematics learning resources that connect programming activities with mathematical concepts, problem-solving tasks and computational thinking. Such evidence may help developers consider how programming can be incorporated into Mathematics learning without making technical requirements the primary focus of instruction. Researchers may benefit from the study's contribution to the empirical literature on Python-integrated Mathematics instruction, particularly within the Nigerian secondary-school context. The findings may provide a reference for replication and further studies involving other programming environments, mathematical topics, educational levels, learner characteristics, instructional durations and learning outcomes.

Delimitations of the Study

The study is delimited to investigating the effect of Python-integrated Mathematics instruction on Mathematics achievement, mathematical problem-solving skills, and computational thinking skills among Senior Secondary School Two (SS2) students in Ondo State, Nigeria. Geographically, the study is restricted to selected senior secondary schools in Ondo State. The findings are therefore interpreted within the context of the participating schools and students and are not intended to represent all secondary-school students in Nigeria.

In terms of participants, the study is limited to SS2 students. SS2 students are selected because they are engaged in the senior secondary-school Mathematics curriculum while not being in the terminal class. In terms of subject matter, the study is restricted to Mathematics and selected SS2 Mathematics topics covered by the instructional programme developed for the study. Python-integrated instruction in other school subjects is outside the scope of the investigation.

With respect to the intervention, the study is limited to Python-integrated Mathematics instruction, in which structured Python activities are incorporated into selected Mathematics lessons to support the learning and application of mathematical concepts and procedures. The study does not compare Python with other programming languages or environments such as Scratch, MATLAB, R, Java, or C++. The comparison condition is conventional Mathematics instruction as implemented within the study. The study is further delimited to three outcome variables: Mathematics achievement, mathematical problem-solving skills, and computational thinking skills.

LITERATURE REVIEW

Concept of Mathematics Education

Mathematics education concerns the teaching and learning of mathematical concepts, procedures, reasoning and applications. It involves more than the acquisition of computational procedures; it includes students' ability to identify relationships, reason mathematically, represent ideas, formulate and solve problems, and communicate

mathematical thinking.

The use of digital technologies has expanded the ways in which mathematical ideas can be explored and represented. Programming environments, for example, can be incorporated into mathematics lessons to represent relationships, explore patterns, test conjectures, generate numerical or graphical outputs and examine mathematical procedures. This creates a connection between mathematical activity and computational processes.

Ye *et al.* (2023), in a systematic review of computational thinking integration in K–12 mathematics education, reported that computational activities can support interactions between mathematical and computational reasoning. The review also indicated that computational thinking may function both as a process through which students engage with mathematics and as an outcome of mathematics learning.

In the present study, this perspective provides the basis for using Python as a medium for learning Mathematics, rather than treating programming as an independent computer-science subject. The focus remains on selected mathematical concepts and procedures, with Python incorporated into learning activities to support their exploration and application.

Mathematics Achievement

Mathematics achievement refers to the level of knowledge and skills demonstrated by students in relation to specified mathematical content following instruction. It is commonly assessed through tests that require students to recall, understand and apply mathematical concepts, procedures and relationships.

Achievement assessment may involve routine calculations as well as tasks requiring students to apply mathematical knowledge in different situations. In technology-supported learning environments, students may also encounter mathematical tasks through computational representations. Programming can require students to express mathematical relationships as procedures, generate results and compare computational outputs with expected mathematical outcomes.

Rais and Zhao (2024) examined Python programming integration within Indonesia's Merdeka Belajar curriculum in relation to students' motivation and learning outcomes. Tossavainen *et al.* (2024) also investigated upper-secondary students' exploration of definite integrals in a Python environment, illustrating the use of programming in mathematical exploration.

In this study, Mathematics achievement refers specifically to students' scores on the Mathematics Achievement Test developed for the selected SS2 Mathematics content. It represents the measurable outcome of students' mastery and application of the mathematical content taught during the intervention.

Mathematical Problem-Solving Skills

Mathematical problem-solving skills refer to students'

ability to understand a mathematical problem, identify relevant information, represent the problem, select an appropriate strategy, carry out the strategy and evaluate the solution. Unlike routine computation, problem-solving requires students to determine how mathematical knowledge and procedures can be applied to a given problem.

Programming has conceptual connections with these processes. When students develop a computational solution to a mathematical problem, they may need to decompose the problem, identify variables, formulate procedures, sequence operations, test the solution and correct errors. These activities correspond with several processes involved in mathematical problem-solving.

Refvik and Bjerke (2022), in their review of computational thinking in mathematical problem-solving, reported that computational thinking can be incorporated into different stages of mathematical problem-solving. They also noted the need for further empirical evidence concerning whether the use of computational tools translates into improved mathematical problem-solving performance. Wu *et al.* (2024), in a review of 37 studies, similarly examined the relationship between computational thinking and problem-solving.

These findings establish a conceptual relationship between programming, computational processes and mathematical problem-solving but do not establish that programming necessarily improves problem-solving performance. Accordingly, the present study treats mathematical problem-solving skills as a distinct measurable outcome and examines whether students exposed to Python-integrated Mathematics instruction differ from those receiving conventional Mathematics instruction.

Computational Thinking Skills

Computational thinking refers to approaches to problem-solving that involve processes such as decomposition, abstraction, algorithmic thinking, pattern recognition and systematic evaluation. It is broader than computer operation or programming proficiency, although programming can provide a context in which computational thinking processes are applied and observed. Ye *et al.* (2023) reported that computational thinking in mathematics education involves interaction between mathematical and computational reasoning. Students may use mathematical knowledge to develop computational solutions and use computational outputs to examine and interpret mathematical ideas. Fauzi *et al.* (2024) also reviewed the integration of computational thinking in mathematics education and examined its implementation and reported outcomes. Putrimangiri and Prabawanto (2024) identified programming, visualization, simulation and game-based environments among approaches used to integrate computational thinking into mathematics education.

For the present study, computational thinking skills refer

to students' demonstrable abilities to apply selected computational thinking processes to mathematical and computational tasks. The term "skills" is retained because the construct is assessed through observable student performance rather than treated only as a general disposition or approach to problem-solving.

Python Programming

Python is a high-level programming language used for general-purpose programming as well as mathematical and scientific computing. Its syntax and computational capabilities allow users to perform calculations, implement algorithms, generate numerical outputs, represent relationships and explore mathematical models. In mathematics education, these capabilities can support activities that connect mathematical expressions and procedures with executable computational processes. Students can, for example, translate a mathematical procedure into code, execute it, examine the resulting output and compare the result with mathematical expectations.

The educational relevance of Python, however, depends on its use within appropriate learning tasks. Learning programming syntax in isolation does not necessarily constitute Mathematics learning. Tossavainen *et al.* (2024) demonstrated an application of Python in upper-secondary students' exploration of definite integrals, showing how a programming environment can be incorporated into mathematical investigation. In the present study, Python is therefore treated as an instructional medium through which selected mathematical concepts and procedures are explored and applied.

Python-Integrated Mathematics Instruction

Python-integrated Mathematics instruction refers to the deliberate incorporation of Python programming activities into Mathematics teaching and learning to support the exploration, understanding and application of mathematical concepts and procedures. The defining feature is the integration of programming with mathematical objectives rather than the teaching of programming as a separate content area. Within such instruction, students may represent mathematical relationships computationally, formulate algorithms, execute programs, examine outputs, compare results with mathematical expectations, identify errors and revise procedures. The particular activities depend on the mathematical content and instructional objectives.

Ye *et al.* (2023) identified interactions between mathematical reasoning and computational processes as a feature of computational thinking integration in mathematics education. Munthe and Naalsund (2024) likewise examined mathematical programming problems in which programming was used to support mathematical learning. Rais and Zhao (2024) provided additional evidence concerning the incorporation of Python programming within a school curriculum. In the present

study, Python-integrated Mathematics instruction refers specifically to the structured instructional intervention in which Python activities are incorporated into selected SS2 Mathematics lessons to support students' learning and application of the prescribed mathematical content. Python is therefore the instructional medium, while Mathematics remains the primary subject focus.

Conventional Mathematics Instruction

Conventional Mathematics instruction refers, in this study, to the regular Mathematics teaching approach implemented without systematic integration of Python programming into the instructional activities. It may include teacher explanations, questioning, worked examples, board demonstrations, textbook exercises, individual practice, assignments and classroom assessment.

The term "conventional" describes the comparison condition used in the study and does not imply that the approach is ineffective. Its purpose is to provide a basis for comparing the outcomes of students receiving Python-integrated Mathematics instruction with those receiving the regular instructional approach. In the present study, conventional Mathematics instruction covers the same selected Mathematics content and instructional period as the experimental condition, but without the systematic incorporation of Python activities.

Relationship among Mathematics Achievement, Mathematical Problem-Solving Skills and Computational Thinking Skills

Mathematics achievement, mathematical problem-solving skills and computational thinking skills are related but distinct constructs. Mathematics achievement concerns students' mastery and application of specified mathematical content. Mathematical problem-solving skills concern the processes used to understand, represent, strategise, solve and evaluate mathematical problems. Computational thinking skills concern the application of computational processes such as decomposition, abstraction, algorithmic thinking and pattern recognition. The constructs can interact within programming-supported Mathematics activities. A mathematical problem may be represented computationally; developing a computational procedure may require decomposition and algorithmic thinking; and interpreting the resulting output may require mathematical reasoning and evaluation. Ye *et al.* (2023) described this relationship as an interaction between mathematical and computational reasoning in mathematics learning. Refvik and Bjerke (2022) and Wu *et al.* (2024) similarly identified connections between computational thinking and mathematical problem-solving.

The conceptual relationship among the three constructs provides a basis for examining them within the same instructional intervention while maintaining their distinct meanings and measures. In the present study, Mathematics

achievement, mathematical problem-solving skills and computational thinking skills are therefore treated as separate dependent variables for determining the outcomes of Python-integrated Mathematics instruction.

Theoretical Review

The present study is principally grounded in Constructivist Learning Theory, supported by Cognitive Load Theory, with Social Constructivism serving as a complementary perspective. These perspectives are relevant because Python-integrated Mathematics instruction involves active engagement with mathematical ideas, translation of mathematical procedures into computational representations, testing and revision of solutions, and management of the cognitive demands associated with learning mathematics through programming.

Constructivist Learning Theory

Constructivist Learning Theory holds that learners develop knowledge actively by relating new experiences to existing knowledge and by engaging with tasks, problems and representations. Learning is therefore not limited to receiving information from the teacher; it involves interpreting experiences, testing ideas and developing understanding through meaningful activity. In mathematics education, this perspective supports learning experiences in which students explore relationships, investigate patterns, apply strategies, examine solutions and reflect on their results.

The relevance of constructivism to programming-integrated mathematics is supported by research on computational thinking. Ye *et al.* (2023), in a systematic review of computational thinking integration in K–12 mathematics education, reported that CT-based mathematics activities frequently incorporate student-centred and constructivist-oriented approaches. Their review also indicated that mathematical and computational reasoning can develop interactively when students use mathematics to construct computational artefacts, interpret computational outputs and explore mathematical ideas.

Angraini *et al.* (2024) similarly examined mathematical computational thinking from a constructivist perspective and highlighted the role of active engagement in developing computational understanding within mathematics learning. These findings provide a theoretical basis for viewing programming as a medium through which students can actively engage with mathematical concepts rather than merely receive procedural explanations.

In the present study, Python provides a computational environment in which students can represent mathematical relationships, execute procedures, examine outputs and revise their solutions. For example, students may translate a mathematical procedure into Python instructions, run the procedure, compare the output with expected mathematical results and investigate discrepancies where they occur. Such activities provide

opportunities for students to construct and evaluate understanding through interaction with mathematical and computational representations.

The teacher's role within this perspective is primarily to provide appropriate tasks, explanations, guidance and feedback that support students' active engagement. Constructivism therefore provides the principal theoretical explanation for how Python-integrated Mathematics instruction may create learning experiences relevant to Mathematics achievement, mathematical problem-solving skills and computational thinking skills.

Cognitive Load Theory

Cognitive Load Theory (CLT), associated particularly with Sweller and subsequent researchers, is based on the limited capacity of working memory during learning. The theory emphasises the importance of instructional design in managing the information learners must process while acquiring new knowledge and skills. Learning may be hindered when instructional demands impose unnecessary cognitive processing, whereas appropriately structured instruction can help learners concentrate on essential aspects of a task.

CLT is particularly relevant to Python-integrated Mathematics instruction because students may need to coordinate mathematical concepts with programming procedures. A learner may simultaneously need to understand the mathematical problem, select an appropriate procedure, translate the procedure into code, interpret Python syntax, execute the code and evaluate the output. The extent to which these demands are manageable may influence the learning experience.

Duran *et al.* (2022), in a systematic mapping review of Cognitive Load Theory in computing education, identified the relevance of cognitive-load principles to instructional design in programming and computing education. They also noted continuing theoretical and measurement issues associated with the application of CLT. In mathematics education, Ngu and Phan (2024) discussed the relevance of cognitive-load principles to instructional approaches designed to manage learners' cognitive demands during mathematical learning. Van Nooijen *et al.* (2024) likewise examined scaffolding from a cognitive-load perspective in complex problem-solving activities.

Within the present study, CLT suggests that Python activities should be structured in ways that make the relationship between the mathematical task and the programming procedure clear. Examples, guided procedures, partially completed code and progressive task difficulty may help students concentrate on the mathematical objective while developing familiarity with the programming environment. Such support is particularly relevant where students have limited previous programming experience.

Thus, CLT provides a supporting explanation for the conditions under which Python-integrated Mathematics instruction may be effective. It also provides an important qualification to the constructivist perspective: although

active exploration can support learning, excessive technical or procedural demands may divert students' attention from the mathematical content. The theory therefore reinforces the need for appropriate instructional design rather than assuming that the presence of technology itself produces improved learning.

Social Constructivism

Social Constructivism, particularly associated with Vygotsky, emphasises the role of social interaction, language, collaboration and guided participation in learning. From this perspective, learners can develop understanding through interaction with teachers and peers, while explanations, questioning, feedback and scaffolding can support learning beyond what learners may accomplish independently.

This perspective is relevant to Python-integrated Mathematics instruction where learning activities involve pair or group discussion, collaborative problem-solving or teacher-guided exploration. Students may discuss mathematical problems, formulate solution procedures, examine code, interpret computational outputs and compare alternative strategies. Such interaction can provide opportunities for students to articulate mathematical reasoning and receive feedback on their approaches.

Ye *et al.* (2023) identified student-centred and interactive approaches as important features of computational-thinking activities in mathematics education. Similarly, Afidatunisa and Juandi (2022) examined the relationship between computational thinking and mathematical problem-solving and highlighted connections between computational processes and mathematical problem-solving activities.

In the present study, Social Constructivism therefore provides a complementary perspective for understanding the possible contribution of teacher guidance and peer interaction during Python-based mathematical activities. It is not treated as a principal theoretical foundation because the central focus of the intervention is the active integration of Python with mathematics learning rather than social interaction itself.

Relevance of the Theories to the Present Study

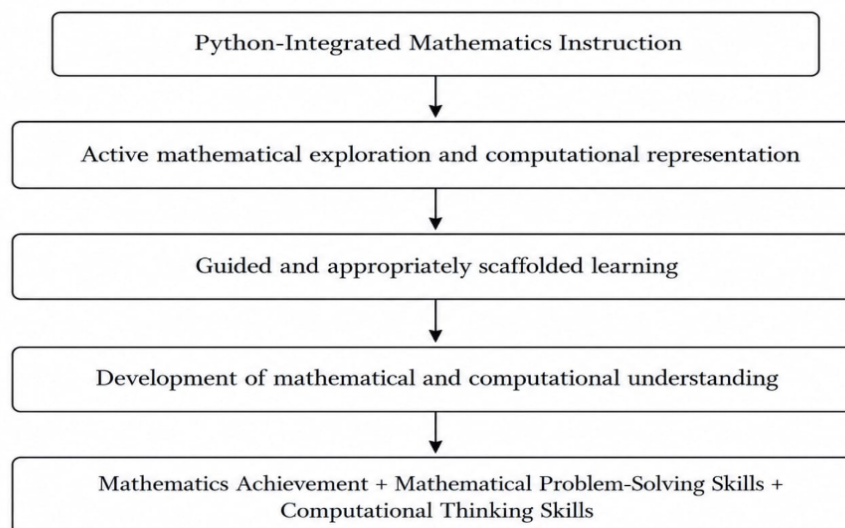
The three perspectives provide complementary explanations for the proposed instructional approach. Constructivist Learning Theory provides the primary theoretical foundation because Python-integrated Mathematics instruction requires students to actively engage with mathematical concepts, representations and problems. Cognitive Load Theory provides a supporting explanation of how the design and sequencing of programming activities may influence students' ability to process mathematical and computational information. Social Constructivism provides a complementary explanation for learning that occurs through teacher guidance, discussion and peer interaction where such activities form part of the intervention.

Table 1: Theoretical Perspective, Central Concerns, and Their Relevance to the Present Study

Theoretical perspective	Central concern	Relevance to the present study
Constructivist Learning Theory	Active construction of knowledge	Students explore, represent, test and revise mathematical ideas through Python-based activities.
Cognitive Load Theory	Management of cognitive demands during learning	Python activities are structured to support students in coordinating mathematical and programming demands.
Social Constructivism	Learning through interaction and guided participation	Teacher guidance and peer interaction may support students' interpretation of mathematical problems, code and computational outputs

The combination of these perspectives provides a coherent theoretical basis for the study. Constructivism explains the learning process, Cognitive Load Theory explains an important instructional condition, and Social Constructivism explains the role of interaction and guidance where collaborative learning occurs. Recent research also supports the relevance of these perspectives to computational-thinking and STEM

learning. Toriq *et al.* (2024) identified constructivist approaches as relevant to experiential and exploratory computational-thinking learning and discussed cognitive-load considerations in the design of instructional materials and activities. Ye *et al.* (2023) similarly showed that mathematics and computational reasoning can interact within student-centred learning environments.


Figure 1: Theoretical Framework of Python-Integrated Mathematics Instruction

The figure 1 framework does not imply that Python integration automatically produces improvement in the three outcomes. Rather, it provides theoretical explanations for why a deliberately designed and appropriately supported Python-integrated mathematics intervention may create learning experiences relevant to these outcomes.

Theoretical Position of the Present Study

The present study adopts Constructivist Learning Theory as its primary theoretical foundation, with Cognitive Load Theory as a supporting theory and Social Constructivism as a complementary perspective. This positioning reflects the structure of the intervention and avoids assigning equal explanatory status to theories that address different aspects of the learning process. Constructivism is central because students are expected

to engage actively with mathematical problems and use Python to represent, test and revise mathematical procedures. Cognitive Load Theory complements this position by drawing attention to the cognitive demands created when mathematical reasoning is combined with programming. Appropriate scaffolding and task sequencing are therefore relevant to ensuring that programming activities remain connected to the mathematical objectives. Social Constructivism complements both perspectives by explaining the possible contribution of teacher guidance and peer interaction during collaborative activities.

The theoretical position is consequently consistent with the purpose of the study: to determine whether Python-integrated Mathematics instruction produces measurable differences in Mathematics achievement, mathematical problem-solving skills and computational thinking skills

among SS2 students in Ondo State, Nigeria, compared with conventional Mathematics instruction. The theories provide the conceptual explanations for the intervention, while the empirical results are required to determine whether the expected differences occur.

Empirical Review

Having established the conceptual and theoretical foundations of the study, this section reviews empirical research on programming-supported mathematics learning, with particular attention to the three outcomes examined in the present study: Mathematics achievement, mathematical problem-solving skills, and computational thinking skills. Studies involving Python are distinguished from those using other programming environments. Evidence from Scratch, robotics, block-based programming and general computer-assisted instruction is considered supporting evidence rather than direct evidence of the effects of Python-integrated Mathematics instruction.

Python Programming and Mathematics Achievement

Research on programming-supported mathematics learning has produced mixed findings. The variation in findings suggests that the relationship between programming and Mathematics achievement may depend on the programming environment, mathematical content, instructional design, duration of exposure and learners' prior knowledge.

Rais and Zhao Xuezhi (2024) investigated Python programming integration within Indonesia's Merdeka Belajar curriculum, examining student motivation and learning outcomes. The study used quantitative methods and considered Python as a vehicle for learning within the curriculum rather than solely as a programming subject. The study provides direct evidence relevant to Python-supported mathematics learning, although differences between the Indonesian and Nigerian educational contexts limit direct generalisation of the findings to Nigerian secondary schools.

Evidence from other programming environments provides additional contextual support. Akinyemi *et al.* (2021) investigated the effect of Scratch Programming Language on Geometry achievement among 116 SS1 students in two public secondary schools in Ibadan, Nigeria. Using a pretest-posttest control-group quasi-experimental design and ANCOVA, the study reported a significant treatment effect on Geometry achievement, $F(1, 111) = 124.80$, $p < .05$, partial $\eta^2 = .53$. The study also reported a significant main effect of gender. Although Scratch differs from Python, the study provides Nigerian evidence that programming-based activities can be incorporated into secondary-school Mathematics instruction and may be associated with differences in achievement.

Contrasting evidence was reported by Laurent *et al.* (2022), who examined the effects of Scratch-based programming activities on primary-school mathematics

learning. The study involved 1,880 students for Euclidean division, 1,763 for additive decomposition and 644 for fractions. Classes were assigned to programming and control conditions, and multilevel analyses indicated small negative effects of the programming condition, ranging from -0.16 to -0.21 . The authors discussed difficulties in transferring learning from programming to mathematics and the possibility that the programming environment could distract learners from the mathematical activity.

The findings of Laurent *et al.* (2022) are particularly relevant because they indicate that programming exposure does not necessarily result in improved Mathematics learning. However, the study involved primary-school learners and Scratch rather than Python and therefore does not establish whether similar effects would occur among senior secondary-school students using Python.

Python-specific classroom research provides further evidence concerning the use of programming as a mathematical learning medium. Tossavainen *et al.* (2024) examined three classroom episodes in which upper-secondary students used Python to explore the definite integral numerically. The study reported that students used code as a mediator of mathematical thinking and that programming supported collaborative discussion and communication of mathematical ideas. However, the study was a case-based investigation and therefore provides evidence about classroom processes rather than a controlled estimate of the effect of Python on Mathematics achievement.

Thompson (2020) also reported on the use of Python in a high-school Advanced Placement Statistics class. In the reported lessons, students used Python to display and describe quantitative data, compare distributions, examine the effects of outliers and explore measures of centre and spread. Students reported learning statistical content and valued working with real-world datasets and visualisations, although some experienced difficulty understanding Python and encountered frustration with syntax errors. The study provides useful evidence concerning the feasibility and instructional affordances of Python in secondary Mathematics, but its short classroom implementation does not establish a general achievement effect.

Further evidence from programming-related technology comes from Hangün and Türel (2025), who examined educational robot programming among 117 Grade 6 students. The study used a quasi-experimental design and found that Mathematics achievement increased in both the experimental and control groups, but the post-test difference between groups was not statistically significant. The study nevertheless reported significant effects for mathematics anxiety and programming self-efficacy. Because the intervention involved educational robots rather than Python, the findings are supporting evidence concerning programming-based technology rather than direct evidence of Python-integrated Mathematics instruction.

Taken together, these studies present mixed evidence.

Some programming-supported Mathematics studies report significant achievement differences, whereas others report nonsignificant or negative effects. Python-specific studies provide evidence of feasibility and mathematical engagement but offer fewer controlled estimates of achievement effects. This combination of findings supports further investigation of Python-integrated Mathematics instruction under a controlled comparative design.

Programming and Mathematical Problem-Solving Skills

Programming has been examined in relation to mathematical problem-solving because programming tasks require learners to interpret problems, identify relevant variables, formulate procedures, sequence operations, test solutions and revise errors. These processes overlap with aspects of mathematical problem-solving, although such overlap does not by itself demonstrate that programming improves mathematical problem-solving performance.

Munthe and Naalsund (2024) examined the design of mathematical programming problems in secondary Mathematics. Their work focused on developing mathematical tasks in which programming supports mathematical learning while Mathematics remains the principal instructional objective. The authors presented design ideas and an in-depth example showing how text-based programming can be incorporated into mathematical problem situations. The study is relevant to the present investigation because it provides a framework for designing programming activities around mathematical objectives. However, it should be regarded primarily as a design-oriented study rather than evidence of a statistically established treatment effect.

Tossavainen *et al.* (2024) provide complementary classroom evidence. Their case study of upper-secondary students exploring the definite integral in Python showed that code and computational outputs could function as mediating objects for mathematical thinking and classroom discussion. The findings indicate how programming may support exploration and interpretation of mathematical ideas, but the study did not compare Python instruction with a conventional control condition. Ye *et al.* (2024) examined flexibility in programming-based mathematical problem-solving, focusing on the interaction between computational and mathematical thinking during problem-solving. Their analysis identified different forms of flexibility in students' solution processes and provided evidence concerning how computational and mathematical thinking may complement or diverge during programming-based mathematical problem-solving. The study is relevant to the present investigation because it demonstrates that programming-based problem-solving involves more than executing predetermined algorithms. However, its focus was on the nature of students' thinking rather than on estimating the causal effect of Python-integrated

Mathematics instruction.

Liang *et al.* (2026) examined how programming outputs can mediate Mathematics learning through perturbation. The study conceptualised programming as an epistemic tool and examined how learners may use computational outputs to investigate mathematical ideas. This evidence is relevant to the present study because it shifts attention from programming as an end in itself to the interpretation of computational outputs as part of mathematical reasoning.

Wang *et al.* (2026) investigated a 24-week coding intervention involving 200 primary-school students. The intervention combined Scratch and Python and compared the experimental group with students receiving the standard curriculum. The authors reported statistically significant gains in problem-solving and computational-thinking measures among the intervention group. However, the participants were aged 8–12 years, the intervention combined two programming environments, and the outcomes were not specifically embedded within a secondary-school Mathematics curriculum. Therefore, the study provides supporting evidence concerning coding and problem-solving but does not establish the independent effect of Python-integrated Mathematics instruction among SS2 students.

The reviewed evidence suggests that programming-based mathematical activities can involve processes relevant to problem-solving, including decomposition, sequencing, testing, debugging and revision. Nevertheless, programming competence and mathematical problem-solving remain distinguishable constructs. The present study therefore assesses mathematical problem-solving as a separate outcome rather than treating computational competence as a direct measure of mathematical problem-solving.

Programming and Computational Thinking Skills

Computational thinking involves approaches to problem-solving that may include decomposition, abstraction, pattern recognition, algorithmic thinking and evaluation. Programming provides one environment in which some of these processes can be explicitly practised. Mathematics provides another context in which such processes can be applied to problems involving relationships, procedures, representations and reasoning.

Laurent *et al.* (2022) provide empirical evidence relevant to this relationship through their investigation of programming and primary Mathematics learning. Their study treated programming as an element of computational thinking and examined whether programming activities supported learning in three mathematical areas. The negative effects reported in the study suggest that computational processes do not automatically transfer to Mathematics learning and that the relationship between computational activities and mathematical outcomes requires appropriate instructional design.

Balacuit *et al.* (2024) investigated the integration of computational thinking into Mathematics through

block-based programming among Grade 7 students. Using a Research and Development approach and backward curriculum design, the researchers developed programming-integrated Mathematics problem-solving activities. The implementation was associated with improvements in students' demonstrated problem-solving and computational-thinking processes, including mathematical reasoning, solution planning, debugging, algorithmic thinking and pattern recognition. Because the study involved block-based programming and a development-oriented design, its findings provide supporting rather than direct evidence concerning Python-integrated Mathematics instruction.

Wang *et al.* (2026) provide further evidence from a structured coding intervention combining Scratch and Python. The study reported significant gains in computational-thinking scores among primary-school participants. The findings are relevant to the present study because they indicate that structured coding exposure can be associated with computational-thinking outcomes. However, the combined programming environments and younger participants limit direct comparison with Python-integrated Mathematics instruction among SS2 students.

The evidence therefore supports examining computational thinking as a distinct outcome. Although computational thinking overlaps with mathematical problem-solving, it should not be treated as synonymous with either programming competence or Mathematics achievement. Measuring computational thinking separately allows the present study to determine whether Python-integrated Mathematics instruction is associated with differences in this specific construct.

Evidence from Nigeria

The Nigerian context warrants separate consideration because educational conditions, curriculum requirements, teacher preparation, students' prior programming exposure and access to digital resources may differ from those reported in studies conducted elsewhere.

Akinyemi *et al.* (2021) provide directly relevant Nigerian evidence through their quasi-experimental study of Scratch Programming Language in Geometry among 116 SS1 students in Ibadan. The significant treatment effect reported in the study provides evidence concerning programming-supported Mathematics instruction within a Nigerian secondary-school setting. However, the study focused on Scratch and Geometry and did not examine Python, mathematical problem-solving or computational thinking as the combined outcomes considered in the present study.

Awofala and Uwajua (2023) also examined technology-supported Mathematics instruction among Nigerian senior secondary-school students and reported a significant treatment effect on Mathematics achievement. Although this evidence is relevant to technology-enhanced Mathematics instruction, it does not directly address Python-integrated Mathematics instruction or

the three outcomes investigated in the present study.

The Nigerian evidence therefore provides a useful contextual foundation but leaves an important empirical question unresolved: whether a deliberately designed Python-integrated Mathematics intervention can produce measurable differences in Mathematics achievement, mathematical problem-solving skills and computational thinking skills among senior secondary-school students.

Critical Synthesis of the Empirical Studies

The reviewed studies present a mixed pattern of findings. Evidence from programming-supported Mathematics instruction includes significant achievement effects, nonsignificant differences and small negative effects. The contrast between Akinyemi *et al.* (2021) and Laurent *et al.* (2022), for example, illustrates that programming-based instruction cannot be assumed to produce the same Mathematics outcome across different contexts, age groups, programming environments and instructional designs.

Python-specific studies provide evidence that programming can be incorporated into secondary Mathematics and used to explore mathematical relationships, computational representations and real-world datasets. However, some of this evidence comes from case studies or design-oriented research rather than controlled experimental comparisons. Consequently, the literature provides stronger evidence for the feasibility and instructional possibilities of Python than for its comparative effectiveness in improving Mathematics achievement.

Evidence concerning mathematical problem-solving indicates that programming-based activities may engage learners in decomposition, algorithm development, testing, debugging and revision. However, the studies do not establish that these programming processes necessarily result in superior mathematical problem-solving performance. This distinction supports the treatment of mathematical problem-solving as a separate outcome in the present study.

Research on computational thinking similarly suggests that programming activities can engage learners in computational processes such as decomposition, algorithmic reasoning, pattern recognition and evaluation. Nevertheless, several studies use Scratch, block-based programming or combined programming environments rather than Python alone. The specific contribution of Python when integrated directly with Mathematics therefore remains insufficiently established.

Another limitation is that relatively few studies examine Mathematics achievement, mathematical problem-solving skills and computational thinking skills simultaneously within the same Mathematics instructional intervention. Existing research frequently concentrates on one outcome or examines programming as a general educational activity. Examining the three outcomes together can therefore provide a broader

assessment of the learning outcomes associated with Python-integrated Mathematics instruction.

Research Gap

The empirical review reveals four related gaps.

First, much of the available evidence concerning programming-supported Mathematics instruction involves programming environments other than Python, particularly Scratch, block-based programming and robotics. These studies provide useful supporting evidence but do not establish the specific effects of Python-integrated Mathematics instruction.

Second, studies involving Python have frequently focused on classroom feasibility, mathematical exploration, programming-based problem-solving or computational representations rather than testing the comparative effect of Python-integrated Mathematics instruction on multiple learning outcomes.

Third, the empirical evidence concerning programming and Mathematics learning is mixed. Positive, nonsignificant and negative findings have been reported. This variation indicates the need for controlled empirical investigation rather than assuming that programming integration will necessarily produce positive learning outcomes.

Fourth, the Nigerian evidence remains limited concerning text-based Python programming as an integrated Mathematics instructional strategy at the senior secondary-school level. In particular, there is limited evidence examining Mathematics achievement, mathematical problem-solving skills and computational thinking skills simultaneously among SS2 students in Ondo State.

The present study addresses these gaps by comparing SS2 students exposed to Python-integrated Mathematics instruction with students receiving conventional Mathematics instruction. The study examines Mathematics achievement, mathematical problem-solving skills and computational thinking skills as separate outcomes while accounting for students' pre-intervention performance. It therefore seeks to provide context-specific evidence concerning the comparative outcomes associated with Python-integrated Mathematics instruction in selected secondary schools in Ondo State, Nigeria.

Summary of the Empirical Review

The empirical evidence indicates that programming can provide an additional environment for exploring, representing and testing mathematical ideas, but its

effects on Mathematics learning are not uniform. Studies involving Python and other programming environments provide evidence of possible benefits for mathematical engagement, problem-solving and computational thinking, while other studies report nonsignificant or negative Mathematics outcomes.

The reviewed studies also differ substantially in programming environment, participants, mathematical content, research design and outcome measures. Python-specific evidence at the senior secondary-school level is particularly limited, while Nigerian studies have more frequently examined Scratch or other forms of technology-supported Mathematics instruction.

A further gap concerns the simultaneous examination of Mathematics achievement, mathematical problem-solving skills and computational thinking skills within one Python-integrated Mathematics intervention. The present study addresses this gap by investigating these three outcomes among SS2 students in Ondo State, Nigeria, using a comparative design involving Python-integrated Mathematics instruction and conventional Mathematics instruction.

MATERIALS AND METHODS

Research Design

The study adopted a quasi-experimental research design, specifically the pre-test–post-test non-equivalent control group design. The design was considered appropriate because intact SS2 classes were used, making random assignment of individual students to instructional conditions impracticable. The study involved two groups: an experimental group that received Python-integrated Mathematics instruction and a control group that received conventional Mathematics instruction. Both groups were administered the Mathematics Achievement Test (MAT), Mathematical Problem-Solving Test (MPST), and Computational Thinking Skills Test (CTST) before the intervention and again after the intervention. The pre-test scores provided baseline measures of the three dependent variables, while the post-test scores provided measures of students' performance following the intervention.

X_1 = Python-integrated Mathematics instruction;

X_0 = Conventional Mathematics instruction;

O_1/O_2 = Mathematics Achievement Test, pre-test/post-test;

Table 2: Representation of the Research Design

Group	Pre-test	Treatment	Post-test
Experimental	O_1, O_3, O_5	X_1	O_2, O_4, O_6
Control	O_1, O_3, O_5	X_0	O_2, O_4, O_6

O_3/O_4 = Mathematical Problem-Solving Test, pre-test/post-test;

O_5/O_6 = Computational Thinking Skills Test, pre-test/post-test.

The non-equivalent control-group design was suitable for the study because the participating students remained in

their existing school/class arrangements. The inclusion of pre-tests made it possible to examine initial differences between the groups and to use the corresponding pre-test scores as covariates in the post-test ANCOVA where appropriate. The design was also consistent with the purpose of the study, which was to determine whether

students exposed to Python-integrated Mathematics instruction differed in post-test Mathematics achievement, mathematical problem-solving skills, and computational thinking skills from students who received conventional Mathematics instruction.

Population of the Study

The population of the study comprised all Senior Secondary School Two (SS2) students offering Mathematics in secondary schools in Ondo State, Nigeria. SS2 students were selected because they are within the senior secondary-school Mathematics curriculum and are not in the terminal class, thereby providing an appropriate level for implementing the instructional intervention without the immediate examination demands associated with SS3.

Sample and Sampling Technique

A sample of 160 SS2 Mathematics students was selected from four secondary schools in Akure South and Akure North Local Government Areas of Ondo State, within Ondo Central Senatorial District. The four participating schools comprised two public and two private secondary schools. Forty (40) SS2 Mathematics students were selected from each school, giving a total sample of 160 students.

A multistage sampling procedure was used. First, Ondo central senatorial zone was randomly selected from the three senatorial zone in Ondo State. Second, Akure South and Akure North Local Government Areas were randomly selected from the six Local Government Areas in Ondo Central Senatorial zone. Thirdly, four secondary schools were selected, comprising two public and two private schools. Selection was based on the availability of SS2 Mathematics classes, facilities required for the intervention, willingness of the schools to participate in the study, and the feasibility of implementing the intervention within the study period.

Third, 40 SS2 Mathematics students were selected from each participating school. The participating schools were assigned to the experimental and control conditions at the school level. The experimental group comprised 80 students from one public and one private school, while the control group comprised 80 students from one public and one private school. Thus, the final sample consisted of 160 SS2 students, comprising 80 students in the Python-integrated Mathematics instruction group and 80 students in the conventional Mathematics instruction group.

Experimental Treatment

The experimental treatment consisted of Python-integrated Mathematics instruction, while the control group received conventional Mathematics instruction covering the same Mathematics content over the same 10-week period. The intervention covered three Mathematics content areas: Quadratic Equations and Functions; Sequences and Series; and Statistics and

Data Analysis. These topics were selected because they provide opportunities for mathematical representation, calculation, pattern identification, procedural reasoning, data manipulation, interpretation, and problem-solving. Such characteristics permit Python activities to be incorporated into Mathematics lessons while maintaining the mathematical content as the primary instructional focus.

Python-Integrated Mathematics Instruction

The experimental group was taught Mathematics through structured activities in which Python programming was integrated with the selected Mathematics content. Python was used as an instructional tool rather than taught as an independent programming subject. Students engaged in activities involving mathematical representation, calculation, testing, interpretation, and evaluation. A typical instructional sequence involved:

Mathematical problem → problem decomposition → mathematical representation → algorithm development → Python implementation → execution → output → mathematical interpretation → evaluation/revision.

Students were introduced only to Python concepts required for the planned mathematical activities. These included variables, numerical operations, input and output, conditional statements, loops, simple functions where necessary, basic data handling, and graphical or data representation relevant to the selected Mathematics topics. Instruction included teacher demonstrations, worked examples, guided activities, pair activities, individual exercises, and mathematical problem-solving tasks. Students were progressively encouraged to construct, execute, interpret, and revise simple Python-based solutions to mathematical problems.

Conventional Mathematics Instruction

The control group received conventional Mathematics instruction covering the same three content areas and for the same 10-week period. Instruction included teacher explanations, worked examples, board demonstrations, questioning, individual exercises, assignments, and conventional mathematical problem-solving activities. The control group did not receive the Python-integrated activities used with the experimental group. Both groups were provided comparable exposure to the selected Mathematics content, instructional duration, and assessment opportunities. The principal difference between the instructional conditions was the integration of Python activities into the Mathematics lessons of the experimental group.

Duration and Organisation of the Intervention

The intervention lasted 10 weeks. The three Mathematics topics were distributed across the intervention period, with sufficient instructional time allocated to concept development, guided practice, Python-integrated activities, mathematical problem-solving and assessment. A proposed structure is arranged by phase (week) and

correspondence major activities: Week 1- Orientation, baseline preparation and introductory Python concepts; Weeks 2–4, - Quadratic Equations and Functions; Weeks 5–7, - Sequences and Series; Weeks 8–10, - Statistics and Data Analysis. The exact lesson-by-lesson allocation were specified in the intervention lesson plans and treatment manual.

Research Instruments

Three instruments were used for data collection

- Mathematics Achievement Test (MAT);
- Mathematical Problem-Solving Test (MPST); and
- Computational Thinking Skills Test (CTST).

The three instruments correspond directly to the three dependent variables of the study.

Mathematics Achievement Test (MAT)

The Mathematics Achievement Test was designed to measure students' achievement in the three Mathematics content areas taught during the intervention. The instrument was structured as a multiple-choice test. An initial pool of 40 items was developed, from which 30 items were retained following expert validation and pilot item analysis. The final MAT comprised 30 multiple-choice items, with four response options (A–D). The items covered Quadratic Equations and Functions, Sequences and Series, and Statistics and Data Analysis. The items were distributed across different cognitive levels, including knowledge and understanding, application, and analysis/reasoning. Each correctly answered item attracted one mark, while an incorrect or unanswered item attracted zero. Thus, the maximum obtainable score was 30.

Mathematical Problem-Solving Test (MPST)

The Mathematical Problem-Solving Test was developed to measure students' ability to apply mathematical knowledge in solving non-routine and contextual mathematical

problems. An initial pool of 15 structured problems was developed, from which 10 problems were retained following validation and pilot testing. Each problem was scored using a structured rubric that considered: understanding and interpretation of the problem; selection of an appropriate strategy; execution of the mathematical procedure; and evaluation and interpretation of the solution. Each problem carried 5 marks, giving a maximum score of 50 marks. The MPST therefore measures the process of mathematical problem-solving, rather than merely students' ability to select or recall a correct answer.

Computational Thinking Skills Test (CTST)

The Computational Thinking Skills Test was developed to measure students' computational thinking skills within mathematical contexts. The instrument was designed around five dimensions: Decomposition; Pattern recognition; Abstraction; Algorithmic thinking; and Evaluation and debugging. An initial pool of 35 items was developed, from which 25 items were retained following expert validation and pilot item analysis. The final CTST contained 25 items. The items were presented primarily through mathematical and computational scenarios. Some items incorporated simple algorithms, representations or short Python code segments where appropriate. The instrument was designed to assess computational thinking rather than mere knowledge of Python syntax. Consequently, students were required to analyse problems, identify patterns, abstract relevant information, develop or evaluate procedures and identify errors or weaknesses in computational solutions. Each correct response attracted one mark, giving a maximum score of 25.

Table of Specifications

The instruments were developed using Tables of Specifications to ensure adequate representation of the

Table 3: Mathematics Achievement Test table of Specifications

Content area	Knowledge/ Understanding	Application	Analysis/ Reasoning	Total
Quadratic Equations & Functions	3	4	3	10
Sequences and Series	3	4	3	10
Statistics and Data Analysis	3	4	3	10
Total	9	12	9	30

Each of the five dimensions of Computational Thinking Skills Test (Decomposition – 5; Pattern recognition – 5; Abstraction – 5; Algorithmic thinking – 5; Evaluation and debugging – 5) have five items each making of a total of twenty-five (25) items

instructional content and intended cognitive processes.

Table of Specifications for the MPST.

Table 4: MPST specification can be structured around the four scoring components

Problem-solving component	Maximum marks per problem	Total across 10 problems
Understanding/interpretation	1	10
Strategy selection	1	10
Execution/procedure	2	20
Evaluation/interpretation	1	10
Total	5	50

Validity of the Instruments

The instruments were subjected to face and content validation by experts in Mathematics Education, Science Education/Educational Technology, and Measurement and Evaluation. The experts examined the instruments for relevance, adequacy, clarity, and appropriateness of the items in relation to the research objectives, research questions, selected Mathematics content, and intended constructs. They also examined the alignment between the test items and the Tables of Specifications and identified ambiguous, technically inappropriate, or unsuitable items. The observations and recommendations of the validators were used to revise the instruments before pilot testing.

Reliability of the Instruments

The instruments were pilot-tested with 40 SS2 Mathematics students from schools outside the study area who had characteristics comparable to those of the students participating in the main study.

The test-retest method was used to determine the temporal stability of the instruments. The instruments were administered to the same pilot participants on two occasions separated by a two-week interval. Pearson Product-Moment Correlation was used to determine the relationship between the scores obtained on the two administrations. The reliability coefficients obtained were 0.80 for the MAT, 0.78 for the MPST, and 0.82 for the CTST. These coefficients indicate acceptable temporal stability and support the use of the instruments for the main study.

Experimental Procedure

The study was conducted in three main phases: pre-treatment, treatment, and post-treatment.

Pre-treatment Phase

Permission was obtained from the relevant school authorities before commencement of the study. Participating Mathematics teachers and students were informed about the general procedures of the study. The teachers involved in the experimental condition were oriented on the Python-integrated instructional procedures, lesson plans, and instructional materials.

Pre-test Phase

Before the intervention, both experimental and control groups were administered the MAT, MPST, and CTST under similar administration conditions. The resulting scores provided baseline measures for Mathematics achievement, mathematical problem-solving skills, and computational thinking skills.

Treatment Phase

The experimental group received Python-integrated Mathematics instruction for 10 weeks, while the control group received conventional Mathematics instruction

covering the same content and duration. The intervention was implemented using the prepared lesson plans and instructional materials.

Post-test Phase

At the end of the 10-week intervention, the MAT, MPST, and CTST were administered to both groups. The resulting scores were used for the descriptive and inferential analyses specified for the study.

Control of Extraneous Variables

Measures were taken to minimise the potential influence of extraneous variables.

- Teacher effect: Standardised lesson plans and instructional materials were provided to teachers involved in the intervention. Where feasible, teachers were oriented using the same instructional guidelines.
- Instructional time: Both groups received instruction for the same 10-week period.
- Content exposure: Both groups were taught the same three Mathematics content areas.
- Assessment: Both groups completed the same pre-test and post-test instruments.
- Prior performance: Pre-test scores were obtained before the intervention and the corresponding pre-test scores were used as covariates in the ANCOVA analyses of post-test outcomes.
- Contamination: The study arrangements were organised to reduce unnecessary exchange of intervention materials and activities between the experimental and control groups.
- Testing conditions: Similar administration procedures and scoring criteria were used for both groups.

Method of Data Analysis

The data collected were analysed using descriptive and inferential statistics in accordance with the research questions and null hypotheses.

For Research Questions 1–3, which concerned differences in pre-test scores between the experimental and control groups, mean and standard deviation were used to describe the scores, while independent-samples t-tests were used to determine whether the observed pre-test differences were statistically significant. For Research Questions 4–6, which concerned post-test outcomes after controlling for the corresponding pre-test scores, Analysis of Covariance (ANCOVA) was used. For each analysis, the post-test score served as the dependent variable, instructional group served as the independent variable, and the corresponding pre-test score served as the covariate.

Before conducting the ANCOVA, the assumptions of normality, homogeneity of variance, and homogeneity of regression slopes were examined. The corresponding null hypotheses were tested at the 0.05 level of significance. Where the instructional-group effect was statistically significant, the estimated marginal means (EMMs) and their 95% confidence intervals were examined to

determine the direction and magnitude of the adjusted group differences. The ANCOVA results were reported using the F-statistic, degrees of freedom, p-value, and partial eta squared (partial η^2) as the measure of effect size. Where appropriate, the R^2 and adjusted R^2 of the overall ANCOVA model were also reported to indicate the proportion of variance in the post-test outcome accounted for by the model.

Ethical Considerations

Ethical principles governing educational research were observed throughout the study. Permission was obtained from the relevant school authorities before data collection. Appropriate consent and assent procedures were followed for participation in the study. Students' identities and individual responses were treated confidentially, and the data were used solely for research purposes. Students were informed that participation would not adversely affect their academic assessment or standing in the school

RESULTS AND DISCUSSION

This chapter presents the results of the study on the effect of Python-integrated Mathematics instruction

on Mathematics achievement, mathematical problem-solving skills, and computational thinking skills among Senior Secondary School Two (SS2) students in Ondo State, Nigeria. The results are presented in accordance with the research questions and null hypotheses that guided the study. The first part of the chapter presents the results based on the six research questions using descriptive statistics, while the second part presents the results of the tests of the corresponding null hypotheses using appropriate inferential statistics.

For the present section, attention is given exclusively to the six research questions. The results are presented using the mean and standard deviation of students' scores in the experimental and control groups. The experimental group received Python-integrated Mathematics instruction, whereas the control group received conventional Mathematics instruction.

Results Based on Research Questions

Research Question 1

What difference exists in the pre-test mean Mathematics achievement scores of students exposed to Python-integrated Mathematics instruction and those exposed to

Table 5: Mean and Standard Deviation of Students' Pre-test Mathematics Achievement Scores by Instructional Group

Instructional Group	N	Mean	SD	Mean Difference
Python-integrated Mathematics	80	13.21	3.18	0.43
Conventional Mathematics	80	12.78	3.25	

conventional Mathematics instruction?

Interpretation of Table 5

Table 5 shows that students in the Python-integrated Mathematics group obtained a pre-test mean Mathematics achievement score of 13.21 with a standard deviation of 3.18, while students in the conventional Mathematics group obtained a mean score of 12.78 with a standard deviation of 3.25. The difference between the two group means was 0.43 in favour of the Python-integrated Mathematics group. The results indicate that the experimental group had a slightly higher mean Mathematics achievement score

than the control group before the commencement of the intervention. However, the relatively small mean difference suggests that the two groups had broadly comparable levels of Mathematics achievement at baseline. The statistical significance of this difference will be determined later under the test of Hypothesis 1.

Research Question 2

What difference exists in the pre-test mean mathematical problem-solving scores of students exposed to Python-integrated Mathematics instruction and those exposed to

Table 6: Mean and Standard Deviation of Students' Pre-test Mathematical Problem-Solving Scores by Instructional Group

Instructional Group	N	Mean	SD	Mean Difference
Python-integrated Mathematics	80	22.64	5.41	0.58
Conventional Mathematics	80	22.06	5.36	

to conventional Mathematics instruction?

Interpretation of Table 6

Table 6 indicates that students exposed to Python-integrated Mathematics instruction had a pre-test mean mathematical problem-solving score of 22.64 and a standard deviation of 5.41, whereas those exposed to conventional Mathematics instruction had a mean score of 22.06 and a standard deviation of 5.36.

The mean difference of 0.58 was in favour of the

Python-integrated Mathematics group. The relatively small difference between the two groups suggests that the students entered the study with broadly similar levels of mathematical problem-solving ability. The statistical significance of the observed difference will be established through the corresponding hypothesis test.

Research Question 3

What difference exists in the pre-test mean computational

Table 7: Mean and Standard Deviation of Students' Pre-test Computational Thinking Skills Scores by Instructional Group

Instructional Group	N	Mean	SD	Mean Difference
Python-integrated Mathematics	80	10.84	2.91	0.31
Conventional Mathematics	80	10.53	2.87	

thinking skills scores of students exposed to Python-integrated Mathematics instruction and those exposed to conventional Mathematics instruction?

Interpretation of Table 7

Table 7 shows that students in the Python-integrated Mathematics group recorded a pre-test mean computational thinking skills score of 10.84 with a standard deviation of 2.91, while students in the conventional Mathematics group recorded a mean score of 10.53 with a standard deviation of 2.87.

The mean difference was 0.31 in favour of the Python-

integrated Mathematics group. The small difference suggests that the two groups had relatively comparable levels of computational thinking skills before the intervention. Whether this observed difference was statistically significant will be established when Hypothesis 3 is tested.

Research Question 4

What effect does Python-integrated Mathematics instruction have on students' post-test Mathematics achievement after controlling for their pre-test

Table 8: Mean and Standard Deviation of Students' Mathematics Achievement Scores Before and After the Intervention by Instructional Group

Instructional Group	N	Pre-test Mean	Pre-test SD	Post-test Mean	Post-test SD	Mean Gain
Python-integrated Mathematics	80	13.21	3.18	23.48	3.21	10.27
Conventional Mathematics	80	12.78	3.25	18.07	3.54	5.29

Mathematics achievement scores?

Interpretation of Table 8

Table 8 indicates that students exposed to Python-integrated Mathematics instruction had a pre-test mean Mathematics achievement score of 13.21, which increased to a post-test mean of 23.48, representing a mean gain of 10.27 points. Students exposed to conventional Mathematics instruction had a pre-test mean score of 12.78, which increased to a post-test mean of 18.07, representing a mean gain of 5.29 points. Thus, both groups improved their Mathematics achievement following instruction, but the improvement was considerably greater among students exposed

to Python-integrated Mathematics instruction. The difference in the observed mean gains was 4.98 points in favour of the experimental group. However, because Research Question 4 specifically asks about the effect after controlling for pre-test Mathematics achievement, the descriptive results alone cannot establish whether the treatment effect is statistically significant. This will be determined using ANCOVA under Hypothesis 4.

Research Question 5

What effect does Python-integrated Mathematics instruction have on students' post-test mathematical problem-solving scores after controlling for their pre-test

Table 9: Mean and Standard Deviation of Students' Mathematical Problem-Solving Scores Before and After the Intervention by Instructional Group

Instructional Group	N	Pre-test Mean	Pre-test SD	Post-test Mean	Post-test SD	Mean Gain
Python-integrated Mathematics	80	22.64	5.41	39.18	5.08	16.54
Conventional Mathematics	80	22.06	5.36	32.41	5.62	10.35

problem-solving scores?

Interpretation of Table 9

Table 9 shows that the experimental group's mean mathematical problem-solving score increased from 22.64 at pre-test to 39.18 at post-test, representing a mean gain of 16.54 points. For the control group, the mean score increased from 22.06 at pre-test to 32.41 at post-test, representing a mean gain of 10.35 points. The experimental group therefore recorded a greater mean gain than the control group, with a difference in

mean gain of 6.19 points. This suggests that students who received Python-integrated Mathematics instruction made greater descriptive improvement in mathematical problem-solving than those who received conventional instruction. Nevertheless, the observed difference in gain scores does not by itself establish statistical significance because the analysis specified in the methodology involves controlling for the pre-test problem-solving scores. The statistical effect of the instructional treatment will therefore be established through ANCOVA in the

hypothesis-testing section.

Research Question 6

What effect does Python-integrated Mathematics

instruction have on students' post-test computational thinking skills after controlling for their pre-test

Table 10: Mean and Standard Deviation of Students' Computational Thinking Skills Scores Before and After the Intervention by Instructional Group

Instructional Group	N	Pre-test Mean	Pre-test SD	Post-test Mean	Post-test SD	Mean Gain
Python-integrated Mathematics	80	10.84	2.91	20.16	2.72	9.32
Conventional Mathematics	80	10.53	2.87	15.21	3.01	4.68

computational thinking skills scores?

Interpretation of Table 10

Table 10 reveals that students in the Python-integrated Mathematics group obtained a pre-test mean computational thinking skills score of 10.84, which increased to a post-test mean of 20.16, representing a mean gain of 9.32 points. In the conventional Mathematics group, the mean score increased from 10.53 at pre-test to 15.21 at post-test, representing a mean gain of 4.68 points. The Python-integrated Mathematics group therefore recorded a substantially greater improvement in computational thinking skills than the conventional instruction group. The difference in mean gain was 4.64 points in favour of the experimental group. The descriptive results suggest that the Python-integrated Mathematics intervention may have contributed to greater development of computational thinking skills. However, because the research question specifies that the post-test scores should be considered after controlling for pre-test computational thinking skills, the statistical significance and magnitude of the treatment effect will be established

later through ANCOVA.

Summary of Results for the Research Questions

The descriptive results from the six research questions show a consistent pattern. At pre-test, the experimental and control groups had relatively similar mean scores in Mathematics achievement, mathematical problem-solving and computational thinking skills. Following the 10-week intervention, both groups demonstrated improvement across all three outcomes, but the experimental group recorded substantially larger mean gains.

Overall, the table 11, descriptive findings indicate that Python-integrated Mathematics instruction was associated with greater improvement in all three measured outcomes than conventional Mathematics instruction. However, we should stop the Research Question section here. We should not yet say that Python caused the differences or that the differences are statistically significant. Those conclusions belong to the six hypothesis tests. The next section will therefore begin 4.3 Test of Null Hypotheses, starting with Hypothesis 1, and we will build the t-test and ANCOVA tables so that their statistics are mathematically consistent

Table 11: Summary of Results for the Research Questions

Outcome	Experimental Mean Gain	Control Mean Gain	Difference in Gain
Mathematics Achievement	10.27	5.29	4.98
Mathematical Problem-Solving	16.54	10.35	6.19
Computational Thinking Skills	9.32	4.68	4.64

with the means and standard deviations reported above.

Test of Null Hypotheses

Because these three hypotheses concern differences between two independent groups at pre-test, the appropriate statistic is the independent-samples t-test.

Hypothesis One

H_{01} : There is no significant difference in the pre-test mean Mathematics achievement scores of students exposed to Python-integrated Mathematics instruction and those

Table 12: Independent-Samples t-Test of the Difference in Pre-test Mathematics Achievement Scores by Instructional Group

Instructional Group	N	Mean	SD	Df	T	P	Decision
Python-integrated Mathematics	80	13.21	3.18	158	0.845	.399	Fail to reject H_{01}
Conventional Mathematics	80	12.78	3.25				

$P > 0.05$

exposed to conventional Mathematics instruction.

Interpretation of Table 12

Table 12 shows that students exposed to Python-integrated Mathematics instruction had a pre-test Mathematics achievement mean score of 13.21 (SD = 3.18), while those exposed to conventional Mathematics

instruction had a mean score of 12.78 (SD = 3.25). The independent-samples t-test yielded a t-value of 0.845 with 158 degrees of freedom, and the corresponding p-value was .399. Since the p-value of .399 is greater than the 0.05 level of significance, the difference between the two groups' pre-test Mathematics achievement scores was

not statistically significant. Therefore, the null hypothesis failed to reject. This indicates that there was no significant difference in Mathematics achievement between the two groups before the intervention. This finding indicates that there was no statistically significant difference between the two groups in Mathematics achievement at baseline,

suggesting reasonable baseline comparability.

Hypothesis Two

H_{02} : There is no significant difference in the pre-test mean mathematical problem-solving scores of students exposed to Python-integrated Mathematics instruction and those exposed to conventional Mathematics

Table 13: Independent-Samples t-Test of the Difference in Pre-test Mathematical Problem-Solving Scores by Instructional Group

Instructional Group	N	Mean	SD	Df	T	P	Decision
Python-integrated Mathematics	80	22.64	5.41	158	0.681	.497	Fail to reject H_{02}
Conventional Mathematics	80	22.06	5.36				

$P > 0.05$

instruction

Interpretation of Table 13

Table 13 indicates that the Python-integrated Mathematics group had a pre-test mean mathematical problem-solving score of 22.64 (SD = 5.41), whereas the conventional Mathematics group obtained a mean score of 22.06 (SD = 5.36). The independent-samples t-test produced a t-value of 0.681 with 158 degrees of freedom, with a corresponding p-value of .497. Since the p-value of .497 is greater than the 0.05 level of significance, the observed difference between the two groups was not statistically significant. Consequently, the null hypothesis failed to

reject. This means that there was no significant difference in the pre-test mathematical problem-solving scores of students in the two instructional groups. The finding further indicates that the two groups were comparable with respect to mathematical problem-solving ability before the commencement of the intervention.

Hypothesis Three

H_{03} : There is no significant difference in the pre-test mean computational thinking skills scores of students exposed to Python-integrated Mathematics instruction and those

Table 14: Independent-Samples t-Test of the Difference in Pre-test Computational Thinking Skills Scores by Instructional Group

Instructional Group	N	Mean	SD	Df	T	P	Decision
Python-integrated Mathematics	80	10.84	2.91	158	0.678	.499	Fail to reject H_{03}
Conventional Mathematics	80	10.53	2.87				

exposed to conventional Mathematics instruction.

Interpretation of Table 14

Table 14 shows that students in the Python-integrated Mathematics group obtained a pre-test mean computational thinking skills score of 10.84 (SD = 2.91), while students in the conventional Mathematics group obtained a mean score of 10.53 (SD = 2.87). The independent-samples t-test yielded a t-value of 0.678 with 158 degrees of freedom, with a corresponding p-value of .499. Since the p-value of .499 is greater than the 0.05 level of significance, the difference between the two groups was not statistically significant. Therefore, the null hypothesis failed to reject. This indicates that there was no significant difference in the pre-test computational thinking skills scores of students

exposed to Python-integrated Mathematics instruction and those exposed to conventional Mathematics instruction. The finding indicates that there was no statistically significant difference between the two groups in computational thinking skills at baseline, suggesting reasonable baseline comparability before the intervention.

Important methodological point

This pattern indicates that there were no statistically significant baseline differences between the two instructional groups across the three measured outcomes. The results in table 15 indicate that there were no statistically significant baseline differences between the two instructional groups across the three measured outcomes.

Table 15: Summary of the Baseline Hypotheses 1-3

Hypothesis	Outcome	T	P	Decision
H_{01}	Mathematics achievement	0.845	.399	Fail to reject
H_{02}	Mathematical problem-solving	0.681	.497	Fail to reject
H_{03}	Computational thinking skills	0.678	.499	Fail to reject

Discussion

Test of hypothesis 4

H_{04} : There is no significant effect of Python-integrated Mathematics instruction on students' post-test Mathematics achievement after controlling for their pre-

test Mathematics achievement scores.

Test of Assumptions for Analysis of Covariance for

Table 16: Test of Assumptions for ANCOVA of Mathematics Achievement

Assumption/Test	Statistic	df	p	Decision
Shapiro–Wilk: Experimental post-test	.983	80	.371	Satisfied
Shapiro–Wilk: Control post-test	.976	80	.141	Satisfied
Levene's test	0.006	1, 158	.939	Satisfied
Pre-test × Group interaction	0.097	1, 156	.755	Satisfied

H₀₄

Interpretation

The table 16, shapiro–Wilk results indicate that the post-test Mathematics achievement scores did not significantly depart from normality in either the Python-integrated Mathematics group, $W = .983$, $p = .371$, or the conventional Mathematics group, $W = .976$, $p = .141$. Levene's test was not significant, $F(1, 158) = 0.006$, p

$= .939$, indicating that the error variances of the two instructional groups were homogeneous. In addition, the interaction between pre-test Mathematics achievement and instructional group was not statistically significant, $F(1, 156) = 0.097$, $p = .755$. Thus, the assumption of homogeneity of regression slopes was satisfied. Consequently, the assumptions required for conducting ANCOVA on post-test Mathematics achievement were

Table 17: Analysis of Covariance of Post-test Mathematics Achievement by Instructional Group, Controlling for Pre-test Achievement

Source	Type III Sum of Squares	df	Mean Square	F	p	Partial η^2
Corrected Model	1,715.65	2	857.83	107.00	< .001	.577
Intercept	1,015.45	1	1,015.45	126.62	< .001	.446
Pre-test Achievement	544.93	1	544.93	67.95	< .001	.302
Instructional Group	1,060.89	1	1,060.89	132.29	< .001	.457
Error	1,259.09	157	8.02	-	-	-
Total	74,910.40	160	-	-	-	-
Corrected Total	2,974.74	159	-	-	-	-

Model summary: $R^2 = .577$; *Adjusted* $R^2 = .571$.

considered adequately satisfied.

Interpretation of the table 17 ANCOVA

The table 17 results indicate that the overall ANCOVA model was statistically significant, $F(2, 157) = 107.00$, $p < .001$, $R^2 = .577$, adjusted $R^2 = .571$. Thus, the combination of pre-test Mathematics achievement and instructional group accounted for approximately 57.7% of the variance in students' post-test Mathematics achievement. The pre-test Mathematics achievement score was also a statistically significant covariate, $F(1, 157) = 67.95$, $p < .001$, partial $\eta^2 = .302$. This indicates that students' baseline Mathematics achievement was significantly associated with their post-

test achievement after accounting for instructional group. More importantly, instructional group had a statistically significant effect on students' post-test Mathematics achievement after controlling for their pre-test achievement, $F(1, 157) = 132.29$, $p < .001$, partial $\eta^2 = .457$. The partial eta squared indicates a large effect of instructional group. Therefore, the null hypothesis that there is no significant effect of Python-integrated Mathematics instruction on students' post-test Mathematics achievement after controlling for pre-test Mathematics achievement is rejected.

Table 18: Estimated Marginal Means of Post-test Mathematics Achievement by Instructional Group

Instructional Group	N	Adjusted Mean	Std. Error	95% CI Lower	95% CI Upper
Python-integrated Mathematics	80	23.36	0.32	22.73	23.98
Conventional Mathematics	80	18.19	0.32	17.57	18.82

The adjusted mean difference is: $23.36 - 18.19 = 5.17$ points

Estimated Marginal Means

Interpretation of the EMM of Post-test Mathematics Achievement

Table 18, After controlling for students' pre-test Mathematics achievement, the estimated marginal mean for students exposed to Python-integrated Mathematics instruction was 23.36 ($SE = 0.32$, 95% CI [22.73, 23.98]),

compared with 18.19 ($SE = 0.32$, 95% CI [17.57, 18.82]) for students exposed to conventional Mathematics instruction. The adjusted mean difference of approximately 5.17 points favoured the Python-integrated Mathematics group. The adjustment is consistent with the small baseline difference observed in Table 5 and indicates that the higher post-test achievement of the experimental group remained evident

after controlling for initial Mathematics achievement.

Hypothesis Five

H₀₅: There is no significant effect of Python-integrated Mathematics instruction on students' post-test mathematical problem-solving scores after controlling for their pre-test problem-solving scores.

Test of Assumptions for Mathematical Problem-Solving ANCOVA for H₀₅

Because we already established a general assumption-testing section for the three ANCOVAs, H₀₅ should draw from that section rather than unnecessarily repeat all the tests. For the problem-solving outcome, the working values can be presented as follows.

Table 19, the Shapiro–Wilk results indicate that the post-test mathematical problem-solving scores did not significantly depart from normality in either the Python-integrated Mathematics group, $W = .982$, $p = .344$, or the conventional Mathematics group, $W = .979$, $p = .205$. The distributions therefore showed no serious departure from normality.

Levene's test was not statistically significant, $F(1, 158) = 0.987$, $p = .322$, indicating that the error variances of the two instructional groups were sufficiently homogeneous. Similarly, the interaction between pre-test problem-solving scores and instructional group was not statistically significant, $F(1, 156) = 0.742$, $p = .390$. Thus, the assumption of homogeneity of regression slopes was

Table 19: Test of Assumptions for ANCOVA of Mathematical Problem-Solving Scores

Assumption/Test	Statistic	Df	P	Decision
Shapiro–Wilk: Experimental post-test	.982	80	.344	Satisfied
Shapiro–Wilk: Control post-test	.979	80	.205	Satisfied
Levene's test	0.987	1, 158	.322	Satisfied
Pre-test $\times$ Group interaction	0.742	1, 156	.390	Satisfied

satisfied.

The results therefore support the use of ANCOVA to

examine the effect of instructional group on post-test mathematical problem-solving scores after controlling

Table 20: Analysis of Covariance of Post-test Mathematical Problem-Solving Scores by Instructional Group, Controlling for Pre-test Problem-Solving Scores

Source	Type III Sum of Squares	Df	Mean Square	F	P	Partial η^2
Corrected Model	4,675.20	2	2,337.60	94.87	< .001	.547
Intercept	1,745.36	1	1,745.36	70.82	< .001	.311
Pre-test Problem-Solving	1,525.84	1	1,525.84	61.91	< .001	.283
Instructional Group	3,149.36	1	3,149.36	127.75	< .001	.449
Error	3,870.50	157	24.65	-	-	-
Total	208,550.40	160	-	-	-	-
Corrected Total	8,545.70	159	-	-	-	-

Model summary: $R^2 = .547$; *Adjusted* $R^2 = .541$.

for pre-test scores.

Interpretation of Table 20

The results in Table 20 show that the overall ANCOVA model was statistically significant, $F(2, 157) = 94.87$, $p < .001$, $R^2 = .547$, adjusted $R^2 = .541$. This indicates that the combination of pre-test mathematical problem-solving scores and instructional group explained approximately 54.7% of the variance in students' post-test problem-solving scores. The covariate, pre-test mathematical problem-solving score, had a statistically significant effect on post-test problem-solving performance, $F(1, 157) = 61.91$, $p < .001$, partial $\eta^2 = .283$. This indicates that students' initial problem-solving performance was significantly associated with their subsequent performance after accounting for instructional group.

More importantly, instructional group had a statistically significant effect on students' post-test mathematical problem-solving scores after controlling for their pre-test scores, $F(1, 157) = 127.75$, $p < .001$, partial $\eta^2 = .449$. The magnitude of the partial eta squared indicates a large effect

of instructional group on mathematical problem-solving performance. Therefore, the null hypothesis that there is no significant effect of Python-integrated Mathematics instruction on students' post-test mathematical problem-solving scores after controlling for their pre-test scores is rejected.

Estimated Marginal Means

Interpretation of Table 21

The estimated marginal means of table 21 show that, after controlling for students' pre-test mathematical problem-solving scores, students exposed to Python-integrated Mathematics instruction obtained a higher adjusted post-test mean score ($M = 38.98$, $SE = 0.55$, 95% CI [37.89, 40.07]) than students exposed to conventional Mathematics instruction ($M = 32.61$, $SE = 0.55$, 95% CI [31.52, 33.70]). The adjusted mean difference of 6.37 points favoured the Python-integrated Mathematics group. The adjusted difference remains close to the

Table 21: Estimated Marginal Means of Post-test Mathematical Problem-Solving Scores by Instructional Group

Instructional Group	N	Adjusted Mean	Std. Error	95% CI Lower	95% CI Upper
Python-integrated Mathematics	80	38.98	0.55	37.89	40.07
Conventional Mathematics	80	32.61	0.55	31.52	33.70

Adjusted mean difference: $38.98 - 32.61 = 6.37$

observed raw post-test difference of 6.77 points, which is reasonable given the very small baseline difference between the two groups. Taken together, the ANCOVA and estimated marginal means indicate that students exposed to Python-integrated Mathematics instruction performed significantly better in mathematical problem-solving than those exposed to conventional Mathematics instruction after controlling for their initial problem-

solving scores

Hypothesis Six

H₀₆: There is no significant effect of Python-integrated Mathematics instruction on students' post-test computational thinking skills after controlling for their pre-test computational thinking skills scores.

Test of assumptions of computational-thinking

Table 22: Test of Assumptions for ANCOVA of Computational Thinking Skills

Assumption/Test	Statistic	Df	P	Decision
Shapiro-Wilk: Experimental post-test	.984	80	.421	Satisfied
Shapiro-Wilk: Control post-test	.980	80	.247	Satisfied
Levene's test	1.027	1, 158	.312	Satisfied
Pre-test × Group interaction	0.864	1, 156	.354	Satisfied

ANCOVA of H₀₆

Interpretation

Table 22, the Shapiro-Wilk results indicate that the post-test computational thinking scores did not significantly depart from normality in either the Python-integrated Mathematics group, $W = .984$, $p = .421$, or the conventional Mathematics group, $W = .980$, $p = .247$. Thus, there was no serious evidence of non-normality. Levene's test was not statistically significant, $F(1, 158) =$

1.027, $p = .312$, indicating that the variances of the two instructional groups were sufficiently homogeneous.

Furthermore, the interaction between pre-test computational thinking scores and instructional group was not statistically significant, $F(1, 156) = 0.864$, $p = .354$. Therefore, the assumption of homogeneity of regression slopes was satisfied. The results indicate that the assumptions required for ANCOVA of post-test

Table 23: Analysis of Covariance of Post-test Computational Thinking Skills by Instructional Group, Controlling for Pre-test Computational Thinking Skills

Source	Type III Sum of Squares	Df	Mean Square	F	P	Partial η^2
Corrected Model	1,510.62	2	755.31	107.92	< .001	.579
Intercept	403.68	1	403.68	57.69	< .001	.269
Pre-test Computational Thinking	287.54	1	287.54	41.10	< .001	.207
Instructional Group	1,223.08	1	1,223.08	174.84	< .001	.527
Error	1,099.14	157	7.00			
Total	49,120.40	160				
Corrected Total	2,609.76	159				

Model summary: $R^2 = .579$; Adjusted $R^2 = .574$.

computational thinking skills were adequately satisfied.

Interpretation of ANCOVA

The results in Table 23 show that the overall ANCOVA model was statistically significant, $F(2, 157) = 107.92$, $p < .001$, $R^2 = .579$, adjusted $R^2 = .574$. This indicates that the combination of pre-test computational thinking skills and instructional group accounted for approximately 57.9% of the variance in students' post-test computational thinking skills. The covariate, pre-test computational thinking skills, also had a statistically significant effect on post-test computational thinking skills, $F(1, 157) = 41.10$,

$p < .001$, partial $\eta^2 = .207$. This indicates that students' initial computational thinking skills were significantly associated with their subsequent performance after accounting for instructional group.

More importantly, instructional group had a statistically significant effect on students' post-test computational thinking skills after controlling for their pre-test scores, $F(1, 157) = 174.84$, $p < .001$, partial $\eta^2 = .527$. The partial eta squared indicates a large effect of instructional group on computational thinking skills. Therefore, the null hypothesis that there is no significant effect of Python-

integrated Mathematics instruction on students' post-test computational thinking skills after controlling for their pre-test computational thinking skills is rejected.

Estimated Marginal Means

Interpretation of Estimated Marginal Means

The estimated marginal means in Table 24 show that, after controlling for students' pre-test computational thinking skills, students exposed to Python-integrated Mathematics instruction obtained a higher adjusted post-test mean score ($M = 20.03$, $SE = 0.30$, 95% CI [19.44, 20.62]) than students exposed to conventional Mathematics instruction ($M = 15.34$, $SE = 0.30$, 95%

CI [14.75, 15.93]). The adjusted mean difference of 4.69 points favoured the Python-integrated Mathematics group. The adjusted difference remains close to the raw post-test difference of 4.95 points, which is consistent with the relatively small baseline difference between the two groups.

Taken together, the ANCOVA result and estimated marginal means indicate that students exposed to Python-integrated Mathematics instruction demonstrated significantly higher computational thinking skills than students exposed to conventional Mathematics instruction after controlling for their initial computational thinking skills.

Table 24: Estimated Marginal Means of Post-test Computational Thinking Skills by Instructional Group

Instructional Group	N	Adjusted Mean	Std. Error	95% CI Lower	95% CI Upper
Python-integrated Mathematics	80	20.03	0.30	19.44	20.62
Conventional Mathematics	80	15.34	0.30	14.75	15.93

Adjusted mean difference: $20.03 - 15.34 = 4.69$

We now have working ANCOVA models for all three outcomes

The pattern is substantively sensible: All three baseline comparisons are non-significant; All three treatment

effects are significant; All three effects are large; Computational thinking has the largest estimated instructional-group effect; Achievement and problem-solving also show large effects; The adjusted means remain close to the raw post-test means.

Table 25: Summary of working ANCOVA of Hypothesis 4-6

Outcome	R ²	Adjusted R ²	Group F	P	Partial η^2
Mathematics Achievement	.577	.571	132.29	< .001	.457
Mathematical Problem-Solving	.547	.541	127.75	< .001	.449
Computational Thinking	.579	.574	174.84	< .001	.527

Discussion of Findings

Baseline Mathematics Achievement

The finding for Research Question One showed that students exposed to Python-integrated Mathematics instruction had a slightly higher pre-test Mathematics achievement mean score ($M = 13.21$, $SD = 3.18$) than their counterparts exposed to conventional Mathematics instruction ($M = 12.78$, $SD = 3.25$), giving a mean difference of 0.43 in favour of the experimental group. However, the difference was not statistically significant, $t(158) = 0.845$, $p = .399$. Consequently, Hypothesis One was retained, indicating that the two groups were statistically comparable in Mathematics achievement before the intervention.

The absence of a significant baseline difference is important because it suggests that the substantially higher post-test performance of the experimental group cannot reasonably be attributed to an initial advantage in Mathematics achievement. Rather, the groups commenced the intervention with broadly similar levels of achievement. This baseline comparability strengthens the interpretation of the subsequent treatment effect and is particularly appropriate for the non-equivalent group quasi-experimental design used

in the study.

Baseline Mathematical Problem-Solving Skills

Research Question Two examined the difference in pre-test mathematical problem-solving scores between the two instructional groups. The finding indicated that the experimental and control groups had comparable problem-solving performance before the intervention, with the independent-samples t-test showing no statistically significant difference, $t(158) = 0.681$, $p = .497$. Hypothesis Two was therefore retained. The result indicates that neither group possessed a statistically significant initial advantage in mathematical problem-solving skills.

This finding is methodologically important because mathematical problem-solving was one of the principal outcomes of the intervention. The absence of a significant pre-test difference provides reasonable evidence that the two groups began the study from comparable levels of problem-solving ability. This also provides a sound basis for interpreting the subsequent ANCOVA result as an effect associated with the instructional condition rather than a consequence of substantial pre-existing differences.

The finding is consistent with the argument in the empirical literature that meaningful differences in

students' computationally supported mathematical problem-solving are more likely to emerge through appropriately designed learning experiences than from merely exposing students to technology. For example, Ng and Cui (2021) found that programming-based mathematical tasks supported students' mathematical modelling, algorithmic thinking, testing and debugging practices. Thus, establishing comparable problem-solving ability before instruction was important for determining whether such capabilities subsequently developed differently under Python-integrated instruction.

Baseline Computational Thinking Skills

Research Question Three focused on the pre-test computational thinking skills of the two groups. Students in the Python-integrated Mathematics group recorded a mean of 10.84 (SD = 2.91), compared with 10.53 (SD = 2.87) for the conventional Mathematics group. Although the experimental group had a marginally higher mean, the difference was not statistically significant, $t(158) = 0.678$, $p = .499$. Hypothesis Three was therefore retained. The result indicates reasonable baseline equivalence in computational thinking skills before the intervention.

The finding is particularly relevant because computational thinking was subsequently expected to develop through the integration of programming into Mathematics learning. The initial similarity between the groups means that the later difference in computational thinking can be interpreted with greater confidence as being associated with the instructional experience rather than a substantial pre-existing difference.

This baseline finding provides a useful point of departure for the later result because empirical research has shown that programming can support computational thinking through activities involving decomposition, abstraction, algorithmic reasoning and debugging. Bai, Wang and Zhao (2021), for example, reported significant advantages for students exposed to a problem-oriented learning model within a Python programming course on several dimensions of computational thinking. More recent evidence has similarly shown that Python-based programming activities can significantly enhance computational thinking among students.

Effect of Python-Integrated Mathematics Instruction on Mathematics Achievement

Research Question Four examined the effect of Python-integrated Mathematics instruction on students' post-test Mathematics achievement after controlling for their pre-test achievement. The finding showed a clear advantage for the experimental group. The ANCOVA revealed a statistically significant effect of instructional group, $F(1, 157) = 132.29$, $p < .001$, partial $\eta^2 = .457$. Thus, Hypothesis Four was rejected. After adjustment for pre-test achievement, the estimated marginal mean for students taught through Python-integrated Mathematics instruction was 23.36, compared with 18.19 for students taught through conventional Mathematics instruction,

representing an adjusted mean difference of 5.17 points in favour of the experimental group.

The finding suggests that integrating Python into Mathematics instruction substantially enhanced students' Mathematics achievement beyond what was achieved through conventional instruction. One plausible explanation is that Python provides students with opportunities to represent mathematical procedures computationally, execute procedures, obtain immediate feedback, examine outputs and modify their approaches. Such activities can make abstract mathematical relationships more observable and allow students to engage actively with mathematical concepts rather than merely receive procedural explanations.

The finding is broadly consistent with empirical evidence supporting programming-integrated mathematics learning. Research on programming-based mathematics tasks has demonstrated that computational environments can support mathematical modelling and reasoning by requiring learners to translate mathematical ideas into executable procedures. The present finding therefore extends this evidence by demonstrating a significant achievement advantage when Python is deliberately integrated into secondary-school Mathematics instruction.

Effect of Python-Integrated Mathematics Instruction on Mathematical Problem-Solving Skills

Research Question Five examined whether Python-integrated Mathematics instruction affected students' post-test mathematical problem-solving skills after controlling for their pre-test scores. The result revealed a statistically significant effect of instructional group, $F(1, 157) = 127.75$, $p < .001$, partial $\eta^2 = .449$. Hypothesis Five was consequently rejected. The estimated marginal mean was 38.98 for the Python-integrated group and 32.61 for the conventional group, giving an adjusted mean difference of 6.37 points in favour of the experimental group.

The finding indicates that Python-integrated Mathematics instruction was effective in developing students' mathematical problem-solving skills. The relatively large partial eta squared further suggests that the instructional approach had a substantial practical effect. This outcome may be explained by the nature of programming itself, which requires learners to decompose problems, identify relevant variables, formulate procedures, test solutions, identify errors and revise their solutions. These processes closely correspond with important stages of mathematical problem-solving.

The finding agrees with Ng and Cui (2021), who reported that programming-based mathematical problem-solving activities supported modelling, algorithmic thinking, testing and debugging among students. It is also consistent with Balacuit *et al.* (2024), whose programming-integrated Mathematics activities improved students' abilities to understand problems, plan solutions using mathematical reasoning, execute mathematical concepts and debug

errors. The present study therefore provides further empirical support for the argument that programming can serve not merely as a technological add-on but as a medium through which mathematical problem-solving processes can be actively practised.

Effect of Python-Integrated Mathematics Instruction on Computational Thinking Skills

Research Question Six examined the effect of Python-integrated Mathematics instruction on students' post-test computational thinking skills after controlling for their pre-test computational thinking scores. The ANCOVA showed a statistically significant effect of instructional group, $F(1, 157) = 174.84$, $p < .001$, partial $\eta^2 = .527$. The overall model accounted for 57.9% of the variance in post-test computational thinking skills ($R^2 = .579$), while the instructional-group effect alone had a large magnitude. Thus, Hypothesis Six was rejected.

The result indicates that students who received Python-integrated Mathematics instruction developed substantially stronger computational thinking skills than those who received conventional Mathematics instruction. This finding is theoretically plausible because Python-based Mathematics activities require students to translate mathematical problems into structured computational procedures. In doing so, students repeatedly engage in processes such as decomposition, abstraction, pattern recognition, algorithmic thinking, testing and debugging. The finding agrees with Bai *et al.* (2021), who found that a problem-oriented model implemented within a Python programming course produced better computational-thinking outcomes than lecture-and-practice instruction. It is also consistent with recent evidence that Python-based programming can significantly enhance computational thinking, with longer exposure producing sustained gains. Similarly, recent work on Python programming in mathematics contexts has shown that programming can foster decomposition, abstraction, pattern recognition and algorithmic thinking.

Overall, the six findings present a coherent pattern. The three baseline comparisons established that the experimental and control groups were statistically comparable in Mathematics achievement, mathematical problem-solving and computational thinking before instruction. Following the intervention, Python-integrated Mathematics instruction produced statistically significant and practically substantial advantages in all three outcome variables. This pattern strengthens the conclusion that integrating Python into Mathematics instruction can simultaneously support Mathematics achievement, mathematical problem-solving and computational thinking, rather than improving one outcome at the expense of the others.

CONCLUSION

The study examined the effect of Python-integrated Mathematics instruction on Mathematics achievement, mathematical problem-solving skills and computational

thinking skills among Senior Secondary School Two students. The baseline findings established that the experimental and conventional instruction groups were comparable in Mathematics achievement, mathematical problem-solving and computational thinking before the intervention. Following the intervention, the ANCOVA results showed statistically significant effects of Python-integrated Mathematics instruction on all three outcomes, with large effect sizes. Students exposed to Python-integrated Mathematics instruction performed better than those exposed to conventional instruction in Mathematics achievement, mathematical problem-solving and computational thinking skills. The study therefore concludes that appropriately designed Python-integrated Mathematics instruction is an effective approach for improving students' Mathematics learning outcomes while simultaneously developing problem-solving and computational thinking competencies.

Recommendations

Based on the findings, the study recommends that

- Mathematics teachers should integrate appropriate Python-based activities into Mathematics lessons, particularly topics that involve procedures, problem-solving, modelling and logical reasoning.
- Teacher-training institutions should provide Mathematics teachers and student-teachers with basic Python programming and computational-thinking competencies to support effective technology-integrated Mathematics instruction.
- Curriculum planners and educational authorities should consider incorporating programming and computational-thinking activities into secondary-school Mathematics curricula where they align with intended Mathematics learning outcomes.
- Schools and educational administrators should provide adequate computer facilities, relevant software and teacher support to facilitate the effective implementation of Python-integrated Mathematics instruction.
- Mathematics teachers should use Python as a complementary instructional tool, rather than allowing programming activities to overshadow the mathematical concepts and objectives of the lesson.

Suggestions for Further Research

Future researchers should

- Replicate the study in other states and geographical regions of Nigeria to determine whether the findings are generalisable across different educational contexts.
- Examine the effectiveness of Python-integrated Mathematics instruction across different Mathematics topics, school levels and student ability levels.
- Conduct longitudinal studies to determine whether improvements in Mathematics achievement, problem-solving and computational thinking are sustained over time.
- Investigate the effects of Python-integrated Mathematics instruction on other outcomes, such as

Mathematics retention, interest, motivation, creativity and attitudes toward Mathematics and programming.

- Compare Python-integrated Mathematics instruction with other programming languages or computational tools to determine which approaches are most effective for secondary-school Mathematics learning.

REFERENCES

- Afidatunisa, S., & Juandi, D. (2024). Computational thinking skills in mathematics problem-solving: A systematic literature review. *Jurnal Nalar Pendidikan*, 12(2), 131–138. <https://doi.org/10.26858/jnp.v12i2.64838>.
- Angraini, L. M., Kania, N., & Gürbüz, F. (2024). Students' proficiency in computational thinking through constructivist learning theory. *International Journal of Mathematics and Mathematics Education*, 2(1), 45–59. <https://doi.org/10.56855/ijmme.v2i1.963>.
- Akinyemi, A. L., Ogundipe, T., & Adelana, O. P. (2021). Effect of Scratch Programming Language (SPL) on achievement in geometry among senior secondary students in Ibadan, Nigeria. *Journal of ICT in Education*, 8(2), 24–33. <https://doi.org/10.37134/jictie.vol8.2.3.202>.
- Awofala, A. O. A. (2017). Assessing senior secondary school students' mathematical proficiency as related to gender and performance in mathematics in Nigeria. *International Journal of Research in Education and Science*, 3(2), 488–502. <https://doi.org/10.21890/ijres.327908>.
- Bai, H., Wang, X., & Zhao, L. (2021). Effects of the problem-oriented learning model on middle school students' computational thinking skills in a Python course. *Frontiers in Psychology*, 12, Article 771221. <https://doi.org/10.3389/fpsyg.2021.771221>.
- Balacuit, I. C. P., Buan, A. T., Medina, J. B., & Oledan, A. M. B. (2024). Incorporating computational thinking in mathematics through block-based programming: Effects on students' problem-solving skills. *International Journal of Science Education and Teaching*, 3(3), 97–112. <https://doi.org/10.14456/ijset.2024.9>.
- Duran, R., Zavgorodniaia, A., & Sorva, J. (2022). Cognitive load theory in computing education research: A review. *ACM Transactions on Computing Education*, 22(4), Article 40, 1–27. <https://doi.org/10.1145/3483843>.
- Fauzi, A. L., Kusumah, Y. S., Nurlaelah, E., & Juandi, D. (2024). Computational thinking in mathematics education: A systematic literature review on its implementation and impact on students' learning. *Jurnal Kependidikan: Jurnal Hasil Penelitian dan Kajian Kepustakaan di Bidang Pendidikan, Pengajaran dan Pembelajaran*, 10(2), 640–653. <https://doi.org/10.33394/jk.v10i2.11140>.
- Hangün, M. E., & Türel, Y. K. (2025). The effects of robot programming on mathematical achievement, mathematics anxiety, and programming self-efficacy. *Computer Applications in Engineering Education*, 33(3), e70030. <https://doi.org/10.1002/cae.70030>.
- Laurent, M., Crisci, R., Bressoux, P., Chaachoua, H., Nurra, C., de Vries, E., & Tchounikine, P. (2022). Impact of programming on primary mathematics learning. *Learning and Instruction*, 82, Article 101667. <https://doi.org/10.1016/j.learninstruc.2022.101667>.
- Liang, B., Chen, Q., Zhang, Y., & Ng, O.-L. (2026). From output to understanding: How programming outputs mediate mathematics learning via perturbation. *International Journal of Science and Mathematics Education*, 24, Article 36. <https://doi.org/10.1007/s10763-026-10650-6>.
- Ma, H., Zhao, M., Wang, H., Wan, X., Cavanaugh, T. W., & Liu, J. (2021). Promoting pupils' computational thinking skills and self-efficacy: A problem-solving instructional approach. *Educational Technology Research and Development*, 69, 2941–2960. <https://doi.org/10.1007/s11423-021-10016-5>.
- Mangiri, G. R. P., & Prabawanto, S. (2024). Exploring computational thinking in learning mathematics: A systematic literature review from 2017–2024. *Jurnal Pendidikan MIPA*, 25(1), 34–52. <https://doi.org/10.23960/jpmipa.v25i1.pp34-52>.
- Munthe, M., & Naalsund, M. (2024). Designing mathematical programming problems. *Digital Experiences in Mathematics Education*, 10(2), 260–286. <https://doi.org/10.1007/s40751-024-00143-y>.
- Ngu, B. H., & Phan, H. P. (2024). Instructional approach and acquisition of mathematical proficiency: Theoretical insights from learning by comparison and cognitive load theory. *Asian Journal for Mathematics Education*, 3(3), 357–379. <https://doi.org/10.1177/27527263241266765>.
- Ng, O.-L., & Cui, Z. (2021). Examining primary students' mathematical problem-solving in a programming context: Towards computationally enhanced mathematics education. *ZDM—Mathematics Education*, 53(4), 847–860. <https://doi.org/10.1007/s11858-020-01200-7>.
- Rais, D., & Zhao, X. (2024). Elevating student engagement and academic performance: A quantitative analysis of Python programming integration in the Merdeka Belajar curriculum. *Journal on Mathematics Education*, 15(2), 495–516. <https://doi.org/10.22342/jme.v15i2.pp495-516>.
- Stavenæs Refvik, K. A., & Bjerke, A. H. (2022). Computational thinking as a tool in primary and secondary mathematical problem solving: A literature review. *NOMAD: Nordic Studies in Mathematics Education*, 27(3), 5–27. <https://doi.org/10.7146/nomad.v27i3.149191>.
- Thompson, J., Wu, S. P., & Mills, J. (2020). *The use of computer programming in a secondary mathematics class*. 2020 ASEE Virtual Annual Conference. <https://doi.org/10.18260/1-2--35378>.
- Tossavainen, T., Johansson, C., Juhlin, A., & Wedestig, A. (2024). Programming as a mediator of mathematical thinking: Examples from upper secondary students exploring the definite integral. *LUMAT*:

- International Journal on Mathematics, Science and Technology Education*, 12(3), 78–99. <https://doi.org/10.31129/LUMAT.12.3.2155>.
- van Nooijen, C. C. A., de Koning, B. B., Bramer, W. M., Isahakyan, A., Asoodar, M., Kok, E., van Merriënboer, J. J. G., & Paas, F. (2024). A cognitive load theory approach to understanding expert scaffolding of visual problem-solving tasks: A scoping review. *Educational Psychology Review*, 36(1), Article 12. <https://doi.org/10.1007/s10648-024-09848-3>.
- Wang, X., Wan, F., & Dai, J. (2026). Enhancing computational thinking through coding education in primary school students: An experimental study on the impact of early programming exposure on problem-solving skills. *Frontiers in Psychology*, 17, Article 1734482. <https://doi.org/10.3389/fpsyg.2026.1734482>.
- Wu, T.-T., Asmara, A., Huang, Y.-M., & Permata Hapsari, I. (2024). Identification of problem-solving techniques in computational thinking studies: Systematic literature review. *SAGE Open*, 14(2), 1–20. <https://doi.org/10.1177/21582440241249897>.
- Ye, H., Liang, B., Ng, O.-L., & Chai, C. S. (2023). Integration of computational thinking in K–12 mathematics education: A systematic review on CT-based mathematics instruction and student learning. *International Journal of STEM Education*, 10, Article 3. <https://doi.org/10.1186/s40594-023-00396-w>.
- Ye, H., Ng, O.-L., & Cui, Z. (2024). Conceptualizing flexibility in programming-based mathematical problem-solving. *Journal of Educational Computing Research*, 62(2), 374–399. <https://doi.org/10.1177/07356331231209773>.