THIRD TERM
WEEKLY LESSON NOTES – B8
WEEK 7

Week Ending: 11-08-2023	**DAY:**	**Subject:** Computing	
Duration: 60mins		**Strand:** Computational Thinking	
Class: B8	**Class Size:**	**Sub Strand:** Introduction to Programming	
Content Standard: B8.4.1.1. Show an understanding of the concept of programming	**Indicator:** B8.4.1.1.1 Describe the basic concepts in programming		**Lesson:** 1 of 2
Performance Indicator: Learners can describe the basic concepts in programming		**Core Competencies:** CC8.2: CP6.1	
Reference: Computing Curriculum Pg. 36			

Activities For Learning & Assessment	**Resources**	**Progression**
Starter (5mins) Revise with learners to review their understanding in the previous lesson. Share performance indicators and introduce the lesson. ***Main (35mins)*** Introduce the concept of programming and its importance in the world of technology. Explain that programming involves giving instructions to a computer to perform specific tasks. Discuss key concepts such as algorithms, variables, and control structures. Provide a simple problem or task to solve (e.g., making a peanut butter and jelly sandwich). In groups, ask learners to design a step-by-step algorithm to complete the task. Introduce the concepts of variables and control structures (e.g., loops, conditionals). Demonstrate how variables can store and manipulate data, and how control structures help control the flow of a program. Provide examples and encourage learners to identify variables and control structures in familiar scenarios. Task learners to design a simple algorithm for a problem of their choice.	Pictures and videos	Describing the basic concepts in programming

Allow learners to share their algorithms and discuss their thinking process with the class if time permits <u>Assessment</u> 1. What is programming, and why is it important in the world of technology? 2. What are some key concepts in programming? Explain algorithms, variables, and control structures. 3. In groups, design an algorithm for a simple task of your choice. Share your algorithm with the class. 4. How do variables help in programming, and why are they important? 5. What are control structures, and how do they control the flow of a program? ***Reflection (10mins)*** Use peer discussion and effective questioning to find out from learners what they have learnt during the lesson. Take feedback from learners and summarize the lesson.		
Homework/Project Work/Community Engagement Suggestions		
Task learners to design a simple algorithm for a problem of their choice		
Potential Misconceptions/Student Learning Difficulties		
None		

Week Ending: 11-08-2023	**DAY:**	**Subject:** Computing	
Duration: 60mins		**Strand:** Computational Thinking	
Class: B8	**Class Size:**	**Sub Strand:** Introduction to Programming	
Content Standard: B8.4.1.1. Show an understanding of the concept of programming	**Indicator:** B8.4.1.1.1 Describe the basic concepts in programming		**Lesson:** 2 of 2
Performance Indicator: Learners can create a table to compare how the same arithmetic notations are represented in coding and in classroom mathematics		**Core Competencies:** CC8.2: CP6.1	
Reference: Computing Curriculum Pg. 36			

Activities For Learning & Assessment	**Resources**	**Progression**
Starter (5mins) Revise with learners to review their understanding in the previous lesson. Share performance indicators and introduce the lesson. ***Main (35mins)*** Explain the importance of arithmetic operations in programming for performing calculations and manipulating data. Discuss common arithmetic operations such as addition, subtraction, multiplication, and division. Provide learners with a table template with columns for arithmetic notation, mathematical representation, and coding representation.	Pictures and videos	Describing the basic concepts in programming

Arithmetic Notation	Coding Representation	Classroom Mathematics
Addition	+	+
Subtraction	-	-
Multiplication	*	×
Division	/	÷
Exponentiation	** or ^	^ or Exponentiation
Parentheses	( )	( )
Square Root	sqrt() or **0.5	√
Absolute Value	abs()	| | or
Floor Division	//	÷ (with quotient)
Modulo	%	% (Remainder)
Order of Operations	Follows PEMDAS/BODMAS	Follows PEMDAS/BODMAS

Guide learners to fill in the table by comparing arithmetic notations commonly used in mathematics and their equivalent representations in coding languages (e.g., "+" for addition, "-" for subtraction). Review the completed comparison table as a class, discussing any differences or similarities between the two representations. Provide additional examples and ask learners to identify the corresponding coding representation for given arithmetic expressions. <u>Assessment</u> 1. What are some common arithmetic operations used in programming? 2. Create a comparison table with arithmetic notations, mathematical representation, and coding representation. 3. Give an example of an arithmetic expression in mathematics, and identify its coding representation. 4. How does the coding representation of arithmetic notations differ from the mathematical representation? 5. Why is it important for programmers to understand and translate mathematical concepts into coding representations? ***Reflection (10mins)*** Use peer discussion and effective questioning to find out from learners what they have learnt during the lesson. Take feedback from learners and summarize the lesson.		
Homework/Project Work/Community Engagement Suggestions		
Task learners to design a simple algorithm for a problem of their choice		
Potential Misconceptions/Student Learning Difficulties		
None		