Creating Software

Teacher Profile			
Name	Danise Pierce	**Date**	04-07-2024
Email Address	dpierce@toltecsd.org	**School Site**	TES

Lesson Information			
Title of Lesson	Computer Science		
Content Area	Computer Science	**Lesson Plan**	K- Lesson 5: Programming with Harvester 1st - Lesson 6: Loops with Scrat 2nd - Lesson 10: Ocean Scene with Loops 3rd - Lesson 12: Mini-Project: A Royal Battle with Events 4th - Lesson 10: Mini-Project: Sticker Art 5th - Lesson 9: Loops in Ice Age 6th - Lesson 7: Fancy Shapes using Nested Loops 7th and 8th Accelerated Intro to CS Course
Standard(s)	K.AP.A.1 With teacher assistance, model daily processes by following algorithms (sets of step-by-step instructions) to complete tasks. Routines, such as morning meeting, clean-up time, and dismissal, are examples of algorithms that are common in many early elementary classrooms. Just as people use algorithms to complete daily routines, they can program computers to use algorithms to complete different tasks. Algorithms are commonly implemented using a precise language that computers can interpret. For example, students begin to recognize daily step-by-step processes, such as brushing teeth or following a morning procedure, as "algorithms" that lead to an end result. Practice(s): Developing and Using Abstractions: 4.4		

****

1.AP.A.1

Model daily processes by following algorithms (sets of step-by-step instructions) to complete tasks.

Routines, such as morning meeting, clean-up time, and dismissal, are examples of algorithms that are common in many early elementary classrooms. For example, students begin to understand and model daily step-by-step processes, such as brushing teeth, implementing a morning procedure, or following a simple recipe as "algorithms" that lead to an end result.

Practice(s): Developing and Using Abstractions: 4.4

****

2.AP.C.1

Develop programs with sequences and simple loops, to express ideas or address a problem.

Programming is used as a tool to create products that reflect a wide range of interests. Control structures specify the order in which instructions are executed within a program. Computers follow instructions literally. Sequences are the order of instructions in a program. For example, sequences of instructions include steps for drawing a shape or moving a character across the screen. If the commands to program a robot are not in the correct order, the robot will not complete the task desired. Loops allow for the repetition of a sequence of code multiple times. For example, in a program to show the life cycle of a butterfly, a loop could be combined with move commands to allow continual but controlled movement of the character. For example: students independently identify loops and sequences in songs, rhymes, and games, such as the song Head, Shoulders, Knees and Toes

Practice(s): Creating Computational Artifacts: 5.2

****

3.AP.V.1

Create programs that use variables to store and modify data.

Variables are used to store and modify data. At this level, understanding how to use variables is sufficient. Students may use mathematical operations to add to the score of a game or subtract from the number of lives in a game. Programs can imply either digital or paper-based designs.

Practice(s): Creating Computational Artifacts: 5.2

****

4.AP.C.1

Create programs that include sequences, events, loops, and/or conditionals.

Control structures specify the order (sequence) in which instructions are executed within a program and can be combined to support the creation of more complex programs. If dialogue is not sequenced correctly when programming a simple animated story, the story will not make sense. Events allow portions of a program to run based on a specific action. Students could write a program to explain the water cycle and when a specific component is clicked (event), the program would show information about that part of the water cycle. Loops allow for the repetition of a sequence of code multiple times. In a program that produces an animation about a famous historical character, students could use a loop to have the character walk across the screen as they introduce themselves. Conditionals allow for the execution of a portion of code in a program when a certain condition is true. Students could write a math game that asks multiplication fact questions and then uses a conditional to check whether or not the answer that was entered is correct.

Practice(s): Creating Computational Artifacts: 5.2

****

5.AP.C.1

Create programs that include sequences, events, loops, and conditionals.

Control structures specify the order (sequence) in which instructions are executed within a program and can be combined to support the creation of more complex programs. For example, if dialogue is not sequenced correctly when programming a simple animated story, the story will not make sense. Events allow portions of a program to run based on a specific action. For example, students could write a program to explain the water cycle and when a specific component is clicked (event), the program would show information about that part of the water cycle. Loops allow for the repetition of a sequence of code multiple times. For example, in a program that produces an animation about a famous historical character, students could use a loop to have the character walk across the screen as they introduce themselves. Conditionals allow for the execution of a portion of code in a program when a certain condition is true. For example, students could write a math game that asks multiplication fact questions and then uses a conditional to check whether or not the answer that was entered is correct.

Practice(s): Creating Computational Artifacts: 5.1

****

6.AP.C.1

Design programs that combine control structures, including nested loops and compound conditionals.

Control structures can be combined in many ways. Nested loops are loops placed within loops. Compound conditionals combine two or more conditions in a logical relationship (e.g., using AND, OR, and NOT), and nesting conditionals within one another allows the result of one conditional to lead to another. For example, when programming an interactive story, students could use a compound conditional within a loop to unlock a door only if a character has a key AND is touching the door.

Practice(s): Creating Computational Artifacts: 5.1, 5.2

****

7.AP.V.1

Compare and contrast variables that represent different data types and perform operations on their values.

A variable is like a container with a name, in which the contents may change, but the name (identifier) does not. When planning and developing programs, students should decide when and how to declare and name new variables. Students should use naming conventions to improve program readability. For example, possible operations include adding points to the score, combining user input with words to make a sentence, changing the size of a picture, or adding a name to a list of people.

Practice(s): Creating Computational Artifacts: 5.1, 5.2

	** 8.AP.V.1 Create named variables that represent different data types and perform operations on their values. A variable is like a container with a name, in which the contents may change, but the name (identifier) does not. When planning and developing programs, students should decide when and how to declare and name new variables. Students should use naming conventions to improve program readability. Examples of operations include adding points to the score, combining user input with words to make a sentence, changing the size of a picture, or adding a name to a list of people. Practice(s): Creating Computational Artifacts: 5.1, 5.2
Essential Question	What is software?
Learning Target(s) KNOW	Objectives Students will be able to understand the following: • Input • Processor • Memory • Output
Success Criteria SHOW	The students will be able to compare and understand how a computer is designed very much like our own personal computer. .
Vocabulary	Vocabulary • Input - what is put in, taken in, or operated on by any process or system."perceptions and sensory input" • Processor (CPU) - is the logic circuitry that responds to and processes the basic instructions that drive a computer. The CPU is seen as the main and most crucial integrated circuitry (IC) chip in a computer, as it is responsible for interpreting most of computers commands. • Memory - the faculty by which the mind stores and remembers information.

	• Output - a place where power or information leaves a system.
Notes	**Parts of a Computer** In this skill-building lesson, students and I will be discussing each standard mentioned above. Then they will be taking a quiz on Typing.com. They will be permitted to take this quiz as many times as they want. Only the highest score will be recorded. Then they will spend the rest of class time exercising their typing skills in a lesson of they choose. **Purpose** This lesson gives them practice in understanding that a computer relies on many parts just as our body relies on many parts that work together.