


## Key Computational Concepts

This document serves as a guide for understanding and implementing the concepts and skills within the junior/intermediate Math and Coding curriculum with concrete examples of sample activities from Scratch and Code.org.

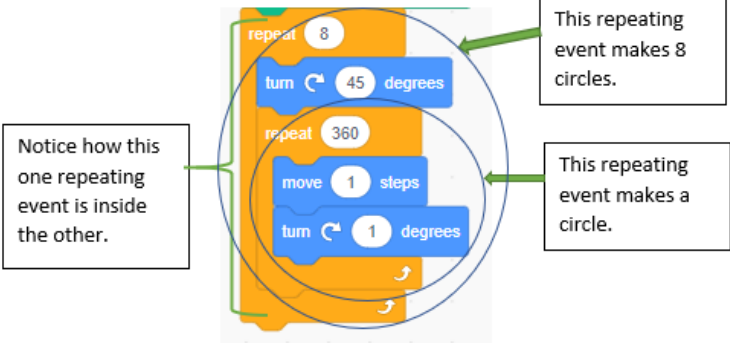
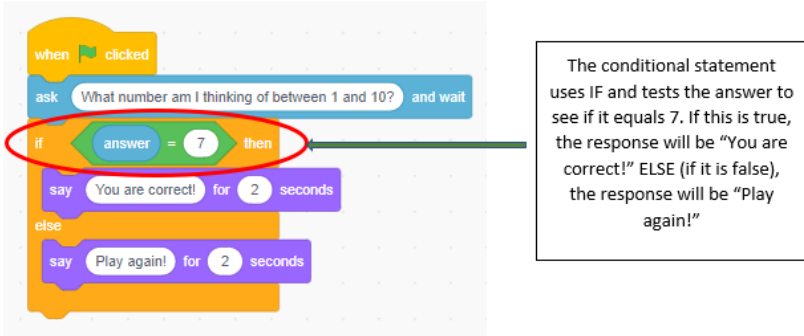


Compiled by Joy Benjamin & Diane Tepylo

Grade OF Intro	Concept /Skill	Definitions	Curriculum (C3 - Ontario 2020)	Code.org Activities
C3.1 Solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves _____ (blanks represent coding vocabulary that changes by grade; found in column 2) C3.2 Read and alter existing code that involves _____, and describe how changes affect the outcomes				
1	Sequences	<b>Sequential Events:</b> actions that occur one after the other. The order of the events is important. As students create and read code, they begin to understand that the instructions are given in order.	Sequential events.	<a href="https://studio.code.org/g/s/courseb-2020/stage/3/puzzle/1">https://studio.code.org/g/s/courseb-2020/stage/3/puzzle/1</a>
2	Concurrent Events	<b>Concurrent Events:</b> actions that happen at the same time or simultaneously. 1) A 2nd (3rd, 4th...) sprite acts at the same time as the 1st sprite. 2) Two threads that run at the same time for simultaneous actions	Sequential and concurrent events.	<a href="https://studio.code.org/g/s/coursee-2020/stage/6/puzzle/1">https://studio.code.org/g/s/coursee-2020/stage/6/puzzle/1</a>
3 <sup>1</sup>	Loops	<b>Repeating Events:</b> actions that occur over and over (also known as loops). Loops can be definite (repeat an exact number of times), indefinite (repeat forever), or conditional (stop when a certain condition is met - see Grade 5).   <div data-bbox="871 1161 1249 1307"> <p>In this section of the code, the sprite moves 100 steps and turns 90 degrees. These two actions are repeated 4 times to make a square.</p> </div> <div data-bbox="808 1323 1186 1437"> <p>Notice the orange bracket around the instructions and the loop symbol at the bottom.</p> </div>	Sequential, concurrent, and repeating events.	<a href="https://studio.code.org/g/s/coursec-2020/stage/10/puzzle/1">https://studio.code.org/g/s/coursec-2020/stage/10/puzzle/1</a>

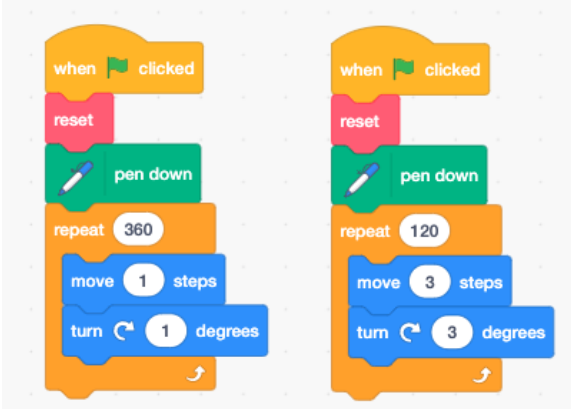
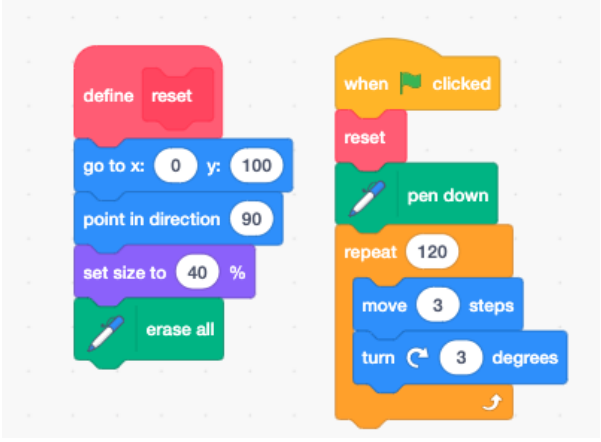
C3.1 Solve problems and create computational representations of mathematical situations by writing and executing **efficient** code, including code that involves \_\_\_\_\_

C3.2 Read and alter existing code that involves \_\_\_\_\_, and describe how changes affect the outcomes and the efficiency of the code

4 <sup>1</sup>	Nested Event	<p><b>Nested Events:</b> events (or loops) within loops. To make a spiral for example, you will need to make a circle (which is a repeating event), turn a few degrees, make another circle and continue making circles.</p> 	Sequential, concurrent, repeating event and nested events.	<a href="https://studio.code.org/s/express-2020/stage/8/puzzle/1">https://studio.code.org/s/express-2020/stage/8/puzzle/1</a>
5 <sup>1</sup>	Conditional statements (if, else, while)	<p><b>Conditional Events:</b> test something to determine if it is true or false and then do an action depending on the result. Conditional statements might be IF/ELSE, WHILE, REPEAT UNTIL</p>  <p><b>Other Control Structures:</b> control structures are anything that changes the flow of the program. Thus they include repeating events, conditional events, time blocks and even events that start running code.</p>	Conditional statements and other control structures.	<a href="https://studio.code.org/s/express-2020/stage/10/puzzle/1">https://studio.code.org/s/express-2020/stage/10/puzzle/1</a>  <a href="https://studio.code.org/s/express-2020/stage/11/puzzle/1">https://studio.code.org/s/express-2020/stage/11/puzzle/1</a>  <a href="https://www.tynker.com/hour-of-code/candy-quest">https://www.tynker.com/hour-of-code/candy-quest</a>

C3.1 Solve problems and create computational representations of mathematical situations by writing and executing **efficient** code, including code that involves:

C3.2 Read and alter existing code that involves \_\_\_\_\_, and describe how changes affect the outcomes and the efficiency of the code.

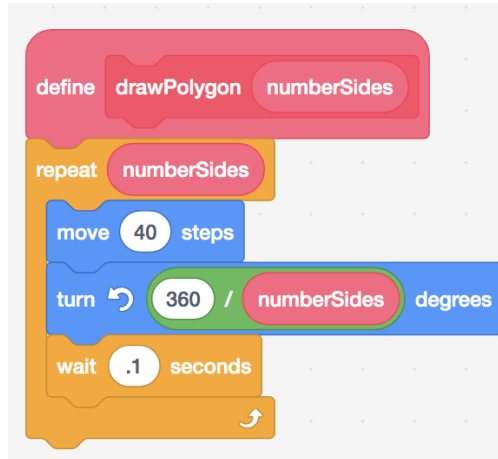
6 <sup>1</sup>	Efficient Code	<p><b>Efficient Code:</b> code that uses the lowest number of instructions to accomplish a task, thereby minimizing storage space and execution time.</p>  <p>The code on the right requires <math>\frac{1}{3}</math> of the moves of the code on the left. It is more <b>efficient</b>. Can you see the difference in the output?</p>	Conditional statements and other control structures.	<a href="https://studio.code.org/g/s/express-2020/stage/13/puzzle/1">https://studio.code.org/g/s/express-2020/stage/13/puzzle/1</a>
7 <sup>1</sup>	Functions (SubPrograms)	<p><b>Subprogram:</b> A small set of instructions for completing one small task. Subprograms can be combined in a main program to accomplish a large task using small steps.</p> 	Events influenced by a defined count and/or subprogram and other control structures.	<p>Variables: <a href="https://studio.code.org/g/s/coursef-2020/stage/10/puzzle/1">https://studio.code.org/g/s/coursef-2020/stage/10/puzzle/1</a></p> <p>Functions: <a href="https://studio.code.org/g/s/coursef-2020/stage/1/puzzle/1">https://studio.code.org/g/s/coursef-2020/stage/1/puzzle/1</a></p> <p>Counters: <a href="https://studio.code.org/g/s/express-2020/stage/22/puzzle/1">https://studio.code.org/g/s/express-2020/stage/22/puzzle/1</a></p>

Subprograms can:

- 1) Make the code easier to read by moving the details out of the main thread. The name of subprogram should clearly indicate what it does
- 2) Simplify debugging - test one subprogram at a time
- 3) Increase the efficiency of a program as the CPU bundles the instructions for reuse

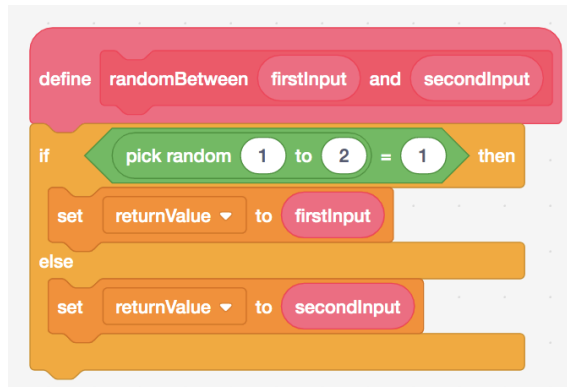
### A subprogram with a parameter

(a value that is passed to the subprogram)



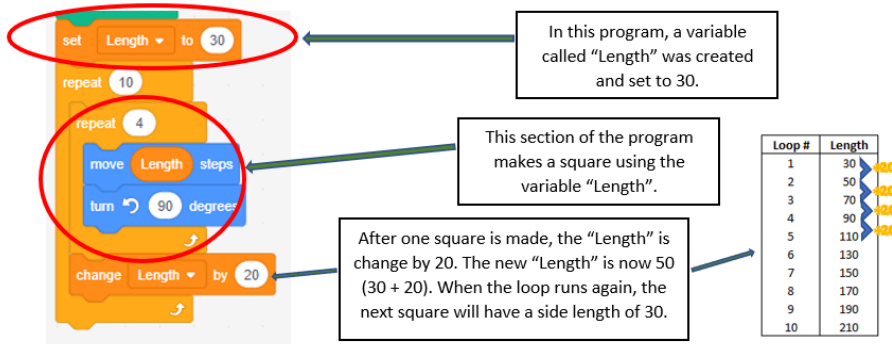
Not within the curriculum, but sometimes helpful

Functions are often used to mean subprograms that return a value. In Scratch, here is a workaround to return a value using a variable



C3.1 Solve problems and create computational representations of mathematical situations by writing and executing **efficient** code, including code that involves:

C3.2 Read and alter existing code that involves \_\_\_\_\_, and describe how changes affect the outcomes and the efficiency of the code.

8 <sup>1</sup>	Variables, Lists (arrays)	<p>Data usually involves conditionals and variables-</p> <p>A variable is simply a placeholder to store a value, and/or to make changes in values often inside a loop or in a conditional event.</p> <p>In mathematics, we often use 'x' as the variable. In coding, we often name the variable more clearly to describe what the variable does.</p> <div><p>The diagram shows a Scratch script with the following blocks: a 'set Length to 30' block, a 'repeat 10' loop containing a 'repeat 4' loop. Inside the inner loop are 'move Length steps', 'turn 90 degrees', and 'change Length by 20' blocks. Annotations explain that 'Length' is set to 30, used in the move block, and then increased by 20. A table shows the values of 'Length' after each loop iteration, increasing from 30 to 210.</p><table><thead><tr><th>Loop #</th><th>Length</th></tr></thead><tbody><tr><td>1</td><td>30</td></tr><tr><td>2</td><td>50</td></tr><tr><td>3</td><td>70</td></tr><tr><td>4</td><td>90</td></tr><tr><td>5</td><td>110</td></tr><tr><td>6</td><td>130</td></tr><tr><td>7</td><td>150</td></tr><tr><td>8</td><td>170</td></tr><tr><td>9</td><td>190</td></tr><tr><td>10</td><td>210</td></tr></tbody></table></div>	Loop #	Length	1	30	2	50	3	70	4	90	5	110	6	130	7	150	8	170	9	190	10	210	Analysis of data in order to inform communication decisions.	
Loop #	Length																									
1	30																									
2	50																									
3	70																									
4	90																									
5	110																									
6	130																									
7	150																									
8	170																									
9	190																									
10	210																									

<sup>1</sup> **Variables:** Variables are not specifically mentioned in relation to coding (expectation C3.1 & 3.2), Variables are, however, stressed throughout expectation C.2 - demonstrate of variables [...] and apply this understanding in various contexts.

In grade 2 (C2.1), students are expected to “identify when symbols are being used as variables and describe how they are being used.” This could connect to C3.2 - read and alter.

In grade 3 (C2.1), students are expected to “describe how variables are being used and use them in various contexts as appropriate.” One of these contexts could be coding.

### Teaching Hints for Variables

- 1) First have students **read** variables in many contexts,
- 1) Then have students **alter** code with variables to meet their own needs
- 2) Last have students **create** their own variables



Compiled by Joy Benjamin & Diane Tepylo