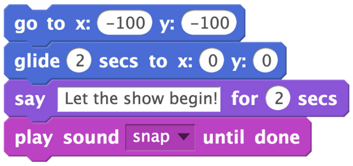
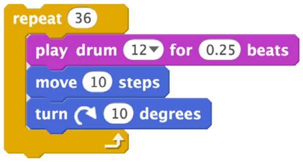
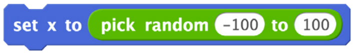
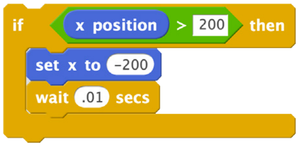
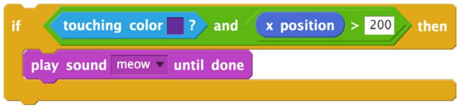
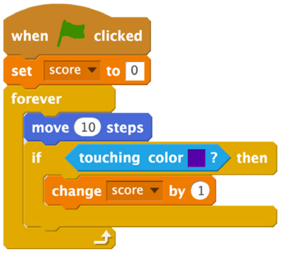
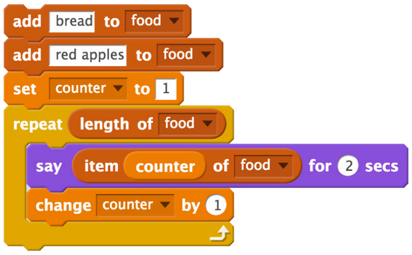
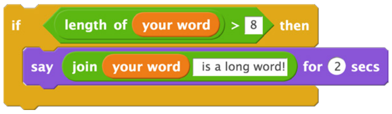
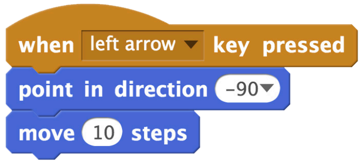
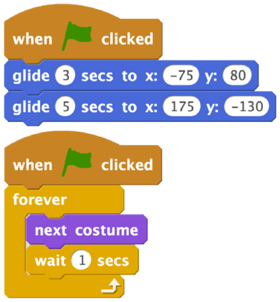
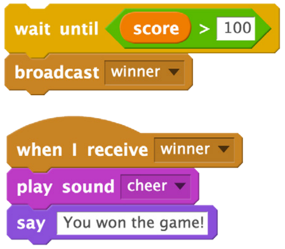
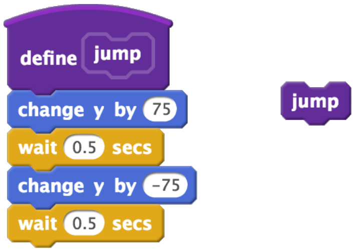
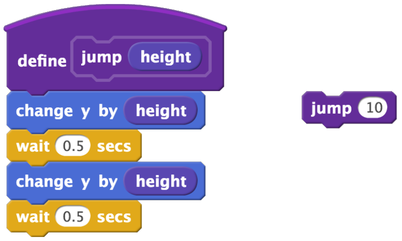
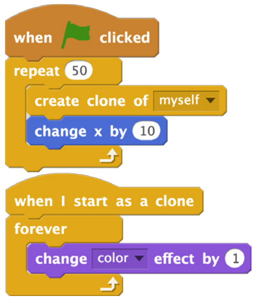
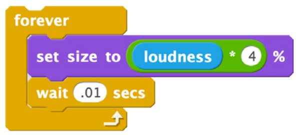
COMPUTATIONAL CONCEPTS SUPPORTED IN SCRATCH

In the process of creating projects with Scratch, young people develop as computational thinkers. They learn concepts, engage in practices, and develop perspectives they can use to express their ideas with code. This list features fundamental computational concepts that are supported in Scratch.

Concept	Explanation	Example
sequence	To create a program in Scratch, you need to think systematically about the order of steps.	 go to x: -100 y: -100 glide 2 secs to x: 0 y: 0 say Let the show begin! for 2 secs play sound snap until done
iteration (looping)	*forever* and *repeat* can be used for iteration (repeating a series of instructions)	 repeat 36 play drum 12 for 0.25 beats move 10 steps turn ↻ 10 degrees
random	*pick random* selects random integers within a given range.	 set x to pick random -100 to 100
conditional statements	*if* and *if else* check for a condition.	 if x position > 200 then set x to -200 wait .01 secs
boolean logic	*and*, *or*, *not* are examples of boolean logic	 if touching color ? and x position > 200 then play sound meow until done
variables	The variable blocks allow you to create variables and use them in a program. Variables can store numbers or strings. Scratch supports both global and object-specific variables.	 when clicked set score to 0 forever move 10 steps if touching color ? then change score by 1
lists (arrays)	The list blocks allow for storing and accessing a list of numbers and strings. This kind of data structure can be considered a "dynamic array."	 add bread to food add red apples to food set counter to 1 repeat length of food say item counter of food for 2 secs change counter by 1
string manipulation	You can change or get information about strings of letters using *length of*, *letter of*, and *join*.	 if length of your word > 8 then say join your word is a long word! for 2 secs

To learn more computational thinking, see scratched.gse.harvard.edu/ct/

Concept	Explanation	Example
event handling	*when key pressed* and *when sprite clicked* are examples of event handling – responding to events triggered by the user or another part of the program.	 when left arrow key pressed point in direction -90 move 10 steps
threads (parallel execution)	Launching two stacks at the same time creates two independent threads that execute in parallel.	 when ⚑ clicked glide 3 secs to x: -75 y: 80 glide 5 secs to x: 175 y: -130 when ⚑ clicked forever next costume wait 1 secs
coordination and synchronization	*broadcast* and *when I receive* can coordinate the actions of multiple sprites. Using broadcast and wait allows synchronization.	 wait until score > 100 broadcast winner when I receive winner play sound cheer say You won the game!
procedures	The *Make a Block* feature lets you define a new block that you use in your scripts. Defining a block can also be called naming a procedure. The Make a Block feature allows reusing code within a sprite, and can support modularity and abstraction.	 define jump change y by 75 wait 0.5 secs change y by -75 wait 0.5 secs jump
parameters	When you *Make a Block*, you can add number, string, and boolean parameters. This allows your new block to have inputs. You can use blocks with parameters to make recursive procedures.	 define jump height change y by height wait 0.5 secs change y by height wait 0.5 secs jump 10
cloning	*create clone* makes a copy of a sprite that exists until the project stops running. You can use it to dynamically create many copies of the same sprite with the same code.	 when ⚑ clicked repeat 50 create clone of myself change x by 10 when I start as a clone forever change color effect by 1
physical sensing	Blocks such as *loudness* allow interactions with microphones and other physical interfaces.	 forever set size to loudness * 4 % wait .01 secs

To learn more about Scratch see scratch.mit.edu/about