



Scratch Blocks: An Overview

In this section, you'll learn about the different blocks available in Scratch, their names, and their intended usage. The goal is to define some of the terms that you'll read in the next chapters. You can come back to this section as you progress if you need to refresh your memory.

As shown in Figure 1-25, Scratch has four kinds of blocks: command blocks, function blocks, trigger blocks, and control blocks. *Command blocks* and *control blocks* (also called *stack blocks*) have bumps on the bottom and/or notches on the top. You can snap these blocks together into stacks. *Trigger blocks*, also called *hats*, have rounded tops because they are placed at the top of a stack. Trigger blocks connect events to scripts. They wait for an event—such as a key press or mouse click—and run the blocks underneath them when that event happens. For example, all scripts that start with the **when green flag clicked** block will run when the user clicks the green flag icon.

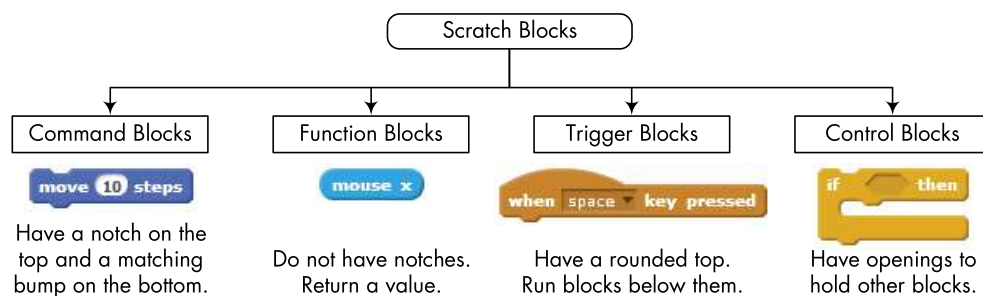


Figure 1-25: The four types of blocks available in Scratch

Function blocks (also called *reporters*) don't have notches or bumps. They can't form a layer of a script alone; instead, they're used as inputs to other blocks. The shapes of these blocks indicate the type of data they return. For example, blocks with rounded ends report numbers or strings, whereas blocks with pointed ends report whether something is true or false. This is illustrated in Figure 1-26.

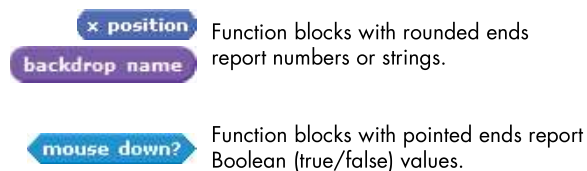


Figure 1-26: The shape of a function block indicates the type of data it returns.

Some function blocks have a checkbox next to them. If you check the box, a *monitor* appears on the Stage to display the current value of the reporter. Select a sprite and check the box on the **x position** block (in the *Motion* palette). Then drag the sprite around the Stage and watch that monitor. It should change as you move the sprite back and forth.

Operators and Functions

The different types of operators in scratch are.

1.Arithmetic operator

2.Relational Operator

3.Logical Operator

Now, let's take a quick look at the arithmetic operators and functions supported in Scratch. If you've lost your calculator, then your worries are over! You could make your own calculator in Scratch with the blocks from the *Operators* palette, which you'll explore in this section.

Arithmetic Operators

Scratch supports the four basic arithmetic operations of addition (+), subtraction (-), multiplication (*), and division (/). The blocks used to perform these operations, called *operators*, are shown in Figure 1-27. Since these blocks produce a number, you can use them as inputs to any block that accepts numbers, as demonstrated in this figure.

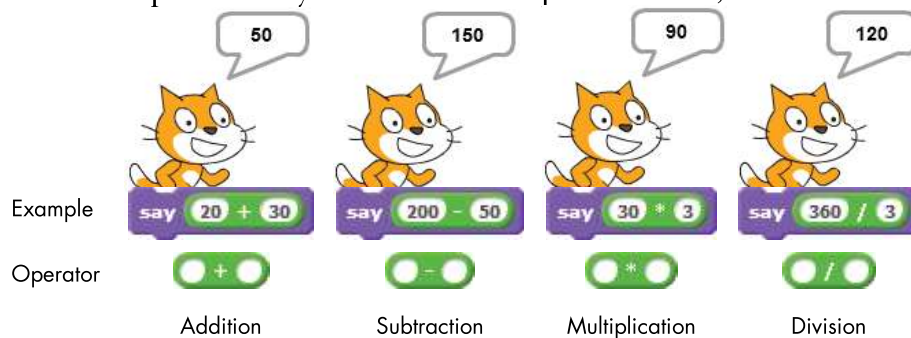


Figure 1-27: Arithmetic operators in Scratch

Scratch also supports the modulus (**mod**) operator, which returns the remainder of the division of two numbers. For example, **10 mod 3** returns 1 because the remainder of dividing 10 by 3 is 1. A common use of the modulus operator is to test the divisibility of one *integer* (whole number) by another (smaller) integer. A modulus of 0 indicates that the larger number is divisible by the smaller number. Does this give you an idea for checking whether a number is even or odd?

Another useful operator supported by Scratch is the round operator, which rounds decimal numbers to the nearest whole number.







For example, $\text{round}(3.1) = 3$, $\text{round}(3.5) = 4$, and $\text{round}(3.6) = 4$.

Random Numbers

As you program more often, you'll probably need to generate random numbers at some point, especially if you create games and simulations. Scratch provides the pick random block specifically for this purpose.

This block outputs a random number each time you use it. Its two editable white boxes allow you to enter a range for that number, and Scratch will only choose values between the two limits (inclusive). Table 1-1 shows some examples of using this block.

Table 1-1: Examples of Using the Pick Random Block

Example	Possible Outcome
	{0, 1}
	{0, 1, 2, 3, ..., 10}
	{-2, -1, 0, 1, 2}
	{0, 10, 20, 30, ..., 100}
	{0, 0.1, 0.15, 0.267, 0.3894, ..., 1.0}
	{0.01, 0.12, 0.34, 0.58, ..., 1.0}







*The outputs of **pick random 0 to 1** and **pick random 0 to 1.0** are different. The first case will give you either a 1 or a 0, but the second gives a decimal value between 0 and 1. If any input to the **pick random** block contains a decimal point, the output will also be a decimal value.*

Relational Operators

Scratch has 3 operators that allow you to compare the relationship between two values or variables. It will compare the two and decide whether it is true or false. If the comparison is true, the program will go down to the next line of code and do what is there. If the comparison is false the code will not execute.

The three relational operators are:

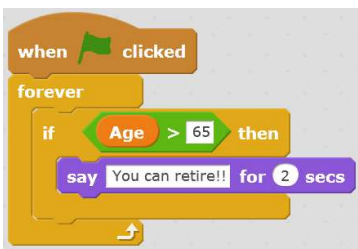
- greater than
- less than
- equal to

Operator	Meaning	Example
	greater than	 Is price greater than 2,000?
	less than	 Is price less than 2,000?
	equal to	 Is price equal to 2,000?

Greater Than

The block checks if the first value is greater than the other value. If it is less, the block returns true; if not, it returns false. This block works with letters too, not just numbers. In Scratch, letters at the top of the alphabet (e.g. a, b, c) are worth less than letters at the end (e.g. x, y, z).

Eg:



When the green flag is clicked, this code will forever check if the value in the age variable is greater than the value of 65. If the value is greater than 65, the sprite will say "You can retire" for 2 seconds. Otherwise nothing will happen.

Less than

The block checks if the first value is less than the second value. If it is less, the block returns true; if not, it returns false. This block works with letters too, as well as numbers. In Scratch, letters at the top of the alphabet (e.g. a, b, c) are worth less than letters at the end (e.g. x, y, z). Example:

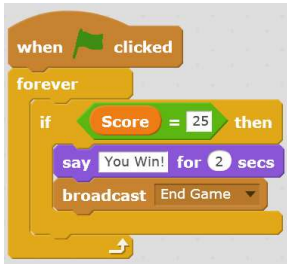


When the green flag is clicked, this code will forever check if the value in the age variable is less than the value of 15. If the value of age is less than 25, the sprite will say "Not Old enough to drive" for 2 seconds. Otherwise nothing will happen.

Equals to

The block checks if the first value is equal to the other value. If the values are equal, the block returns true; if not, false.

Example:






When the green flag is clicked, this code will forever check if the value inside the score variable is equal to the value of 25. If the value of score is equal to 25, the sprite will say "You Win!" for 2 seconds and then broadcast the the End Game message.

LOGIC OPERATORS

Using logical operators, you can combine two or more relational operators to produce a single true/false result. Using these blocks allow you to further refine your comparison of values.

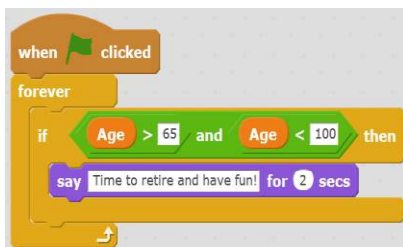
There are 3 logic operators:

- And
- Or
- Not

Operator	Meaning
	The result is true only if the two expressions are true.
	The result is true if either of the two expressions is true.
	The result is true if the expression is false.

And Operator

The and operator takes two expressions as parameters. If both expressions are true, the and operator returns true; otherwise, it returns false.



When the green flag is clicked, this code will forever check if the value inside the age variable is greater than 65 **and** less than 100. If both of those comparisons are true then the sprite will say "Time to retire and have fun!" for 2 seconds. If that comparison is not true then the code will not execute.

Or Operator

The or operator also takes two expressions as parameters. If either expression is true, the or operator returns true. It returns false only when the two expressions are both false.

Example:



When the green flag is clicked, this code will forever check if the value inside the age variable is greater than 0 **or** less than 125. If only one of those comparisons are true then the sprite will say, "Congratulations! You are alive." for 2 seconds. If both of those comparisons are false then the code will not execute.

It is important to realize that only one side of the OR operator has to be true if the code is going to execute.

Not Operator

The not operator takes only one expression as input. The result of the operator is true if the expression is false and false if the expression is true.

Example:

















When the green flag is clicked, this code will forever check if the value inside the score variable is not greater than 100. The comparison will be true if the score is not greater than 100. If it is true the sprite will say, "Sorry, you can't proceed to the next level." for 2 seconds.

It is important to realize that this code will only execute when the comparison is false. Don't try to think about it too much. just remember that with this block, false is true and true is false.

Mathematical Functions

Scratch also supports a large number of mathematical functions. The sqrt of block from the Operator palette contain 14 math functions that can be selected from the drop-down menu, including square root, trigonometric, logarithmic, and exponential functions. Table 1 briefly describes these functions.

Table 1: Scratch's Mathematical Functions

Function	Description
	Returns the absolute value of x . For example, $abs(5) = 5$, $abs(0) = 0$, and $abs(-4) = 4$. Geometrically, $abs(x)$ is the distance between x and 0 on the number line. Similarly, $abs(x - y)$ is the distance between x and y on the number line.
	Returns the largest integer that is less than or equal to x . For example, $floor(2.1) = 2$, $floor(2.9) = 2$, $floor(-2.1) = -3$.
	Returns the smallest integer that is greater than or equal to x . For example, $ceiling(2.1) = 3$, $ceiling(2.9) = 3$, $ceiling(-2.1) = -2$.
	Returns the square root of x . This is another number y such that $y^2 = x$. For example, $sqrt(16) = 4$, $sqrt(2) = 1.4142$, and $sqrt(0) = 0$. Passing a negative value for x returns NaN (short for "not a number").
	Returns the sine of x , where x is an angle expressed in degrees. For example, $sin(0) = 0$, $sin(30) = 0.5$, and $sin(90) = 1$.
	Returns the cosine of x , where x is an angle expressed in degrees. For example, $cos(0) = 1$, $cos(60) = 0.5$, and $cos(90) = 0$.
	Returns the tangent of x , where x is an angle expressed in degrees. For example, $tan(0) = 0$, and $tan(45) = 1$.
	Returns the inverse sine, or arcsine, of x . The arcsine of x is the angle whose sine is x . For example, $asin(0.5) = 30$.
	Returns the inverse cosine, or arccosine, of x . The arccosine of x is the angle whose cosine is x . For example, $acos(0.5) = 60$.
	Returns the inverse tangent, or arctan, of x . The arctan of x is the angle whose tangent is x . For example, $atan(1) = 45$.
	Returns the natural logarithm of x . For example, $ln(2.718) \approx 1$.
	Returns the base-10 logarithm of x . For example, $log(1000) = 3$.
	Returns the exponential function of x . For example, $e^1 \approx 2.718$.
	Returns 10 to the power of x . For example, $10^2 = 100$.