Arithmetic

Standard arithmetic, such as addition and multiplying, can be used in expression with either variables or constants.

For numbers, addition, subtraction and multiplication all use the same operator symbol for integers and singles. Division uses a different operator for getting an integer result than a single.

Operator symbol	Function	Type(s)	Example expression
+	Addition	Integer, single	Number + 1
-	Subtraction	Integer, single	Number – 26
*	Multiplication	Integer, single	Number * 5
/	division	Single	7 / 3 (= 2.333)
\	Division	Integer	7 \ 3 (= 2)
Mod	Remainder	Integer	7 Mod 3 (= 1)
^	Exponent	Integer, single	Area = 3.14159 * radius ^ 2

If there is more than one operator in an expression, then they are either evaluated in the order indicated by parentheses, but if there are no parentheses, exponents are evaluated first, then multiplication and division, and finally addition and subtraction.

For strings, there is an operator to catenate two strings together. For example, in the following:

String1 = “The Cat”
String2 = “in the Hat”
String3 = String1 & “ “ & String2 (Note that the middle string is a single space.)

The value of String3 is “The Cat in the Hat”.

Note that strings can have numbers in them, for example:

String1 = “Result = “
String2 = “35”
String3 = String1 & String 2

The value of String3 is “Result = 35”

Counters and Accumulators

Counters and accumulators are specific types of assignment statements that are used to perform a couple of very useful computations that involve adding numbers into a running total. A classic use of counters and accumulators is for finding an average. In order to get an average, you must get a total of all the items, as well as the number of items.

Counters

Basically, a counter variable counts by repeatedly adding 1. It keeps track of how many times an event occurred. Generally, you will set the counter to 0 as the program opens, and then every time a particular event occurs (e.g., every time a button is pressed), you add 1 to the counter. Later, we will use counters in loops as well.

The formats are very simple:

First, define the variable:	*dim num as integer*	
To set it to zero:	*num = 0*	
To increment it:	*num = num + 1*	(usually done repeatedly)

Accumulators

An accumulator variable accumulates by keeping a running total. This means that you can repeatedly add a number to the variable. This is the same pattern as a counter except that you can add any number and not just 1. Generally, you will set the accumulator to 0 as the program opens, and then every time a particular event occurs (e.g., every time a button is pressed), you add some number to it. Later, we will also use accumulators in loops.

Again, the formats are very simple:

First, define the variable:	*dim total as integer*	
To set it to zero:	*total = 0*	
To add to it:	*total = total + cost*	(usually done repeatedly)