


Chapter 3 – Variables

This chapter is a new chapter in this edition that will introduce you to the concept and basic use of a variable.

What is a Variable

In computer program a variable is "a quantity or function that may assume any given value or set of values."¹ To describe it another way, we can think of a variable as name for a reserved location in the computer's temporary memory. We may store, change, and retrieve values from this location as our program runs by using the variable name.

In BASIC-256 a variable may be used to store integers (whole numbers), decimal numbers, and strings.

 <p>New Concept</p>	<p>Variable</p>
	<p>A variable allows you to assign a name to a block of storage in the computer's short-term memory. You may store, change and retrieve values from these variables in your program.</p> <p>A variable's name must begin with a letter and may contain letters, numbers, and dollar signs. Variable names are case-sensitive and you may not use words reserved by the BASIC-256 language when naming your variables (see Appendix I).</p> <p>Examples of valid variable names include: a, b6, reader, x, f\$4, and zoo.</p>

1 <http://dictionary.reference.com/browse/variable>



Warning

Variable names are case-sensitive. This means that an upper case variable and a lowercase variable with the same letters do not represent the same location in the computer's memory.

Assigning Values to Variables

In this introduction we will use the optional **let** statement to assign values to variables. **Let** calculates the expression on the right sign of the equals sign and then assign that new value to the variable on the left-hand side.

```
1 # letsimple.kbs - use variables to store numbers
2
3 let numerator = 30
4 let denominator = 4
5 let result = numerator / denominator
6
7 print numerator + " / " + denominator + " is " +
  result
8
9 let result = result * 2
10
11 print "result doubled is " + result
```


Program 17: Use Variables to Store Numbers

```
30 / 4 is 7.5
result doubled is 15.0
```

Sample Output: 17: Use Variables to Store Numbers

The program above uses three variables. On line three it stores the value 30

into the location named "numerator". Line four stores the value 4 in the variable "denominator". Line five takes the value from "numerator" divides it by the value in the "denominator" variable and stores the value in the variable named "result". Another thing to watch is on line nine, you can see the statement **let result = result * 2** takes the value in result multiplies it by two and then save the value back into the variable result.




let variable = expression
variable = expression

The let statement will calculate an expression (if necessary) on the right hand side of the equals sign and saves the value into the variable on the left. We call this process assignment or assigning a variable.

variable

expression



The actual **let** statement is optional. You can just assign a variable using the equal sign.

In the first example you saw whole numbers and floating-point numbers stored into variables. In the next example you will see that a variable may contain a string value, just as easily.

```

1 # letstring.kbs = assign a variable a string
2
3 let word = "hello"
4 let rhyme = "yellow"
5
6 stuff = word + " and " + rhyme + " are words that
  rhyme."
```

```

7 print stuff
8 say stuff

```

Program 18: Use Variables to Store Strings

```
hello and yellow are words that rhyme.
```

Sample Output 18: Use Variables to Store Strings

Variable Assignment Shortcuts

Another thing you will learn about computer programming is that there are often more than one way to do a task. BASIC-256 and most computer programming languages allow for a shortcut form of addition and subtraction when working with a variable. In the programs of future chapters you will see these shortcuts.

Shortcut Assignment	Description
<code>variable += expression</code>	Add expression to a variable
<code>variable -= expression</code>	Subtract expression from a variable
<code>variable++</code>	Add one to a variable using old value
<code>variable--</code>	Subtract one from a variable using old value
<code>++variable</code>	Add one to a variable using new value
<code>--variable</code>	Subtract one from a variable using new value

Table 2: Shortcut Variable Assignment

```

1 a = 9
2 a += 10
3 print a ## print 19 - 9 + 10

```

```
4
5     a++ ## a = a + 1 (20)
6     print a
7
8     print a++ ## print a then add 1 (20)
9
10    print a ## 21
11
12    print ++a ## add 1 then print a
```

Program 19: Variable Shortcuts

```
19
20
20
21
22
```

Sample Output 19: Variable Shortcuts

Variable and Data Types

It has been mentioned in prior sections that BASIC-256 understands numbers and strings. Actually, it has four standard types of values: 1) unassigned, 2) integers, 3) floating-point numbers, and 4) strings. We call these data types. In most programming languages there are many more types, but just about all languages have these.

Unassigned

If you attempt to use a variable before it has been assigned a value, it will cause an error or warning to be displayed and the value of "" will be returned to your program.

```
1     print '/' + x + '/'
```

Program 20: Unassigned Variable

```
WARNING on line 1: Variable has not been  
assigned a value.  
//
```

Sample Output 20: Unassigned Variable

In the Preferences settings screen you may choose the "Runtime handling of unassigned variables" option. It has three settings: 1) Ignore – return a value of "" and do not print a warning, 2) Warn – return the value of "" and display a warning message, or 3) Error – Display an error and stop the program.

Integers

An integer is "a whole number (not a fractional number) that can be positive, negative, or zero"². We use integers to count things or to hold exact values. On most computers integers have a range from -2,147,483,648 to 2,147,483,647. The range is limited because the number is stored in 32 bits (4 bytes) of the computer's memory.

Floating-Point Numbers

Numbers with decimal points are also allowed in BASIC-256 but they are stored in the computer's memory as floating-point numbers. Floating-point is "a mathematical notation in which a number is represented by an integer or a decimal fraction multiplied by a power of the number base indicated by an exponent"³. Using this method of storage we can typically represent any number from 1.7×10^{-308} to 1.7×10^{308} . The computer actually stores an approximation of a decimal number by only keeping track of the 15 most significant digits.

Floating-point numbers may be entered as decimal numbers (1.45, -0.998, 12345.678) or entered in scientific notation using an "E" to mark the base 10 exponent ($3.24e-1 = .324$, $1.456e10 = 14560000000.0$). You must not use a

2 <http://whatis.techtarget.com/definition/integer>

3 <http://www.merriam-webster.com/dictionary/floating%E2%80%93point>

thousand's separator when putting the numbers in your program.

When floating-point numbers are printed on the screen or the printed page, they will be shown with a thousand's separator, a decimal point, and a trailing zero if needed. This way, when you see one displayed, you will know it is a float and not an integer.

Strings

A string is "finite sequence of characters (i.e., letters, numerals, symbols and punctuation marks)"⁴ In BASIC-256 a string is a bunch of letters, numbers, and other things surrounded by quotation marks. A string may be surrounded by single quotes (') or double quotes(""). Be careful to always close your string with the same type of quote that you started with.

Examples include: "candy bar", "Say 'hi' to her for me.", and 'Why not?'.

Determining the Type of a Value or Variable

The **typeof** function in BASIC-256 that will tell you the type of the data stored in a variable or the type returned by an expression. **typeof** returns an integer:

typeof Value	Constant	Description
0	TYPE_UNASSIGNED	unassigned variable
1	TYPE_INT	integer
2	TYPE_FLOAT	floating-point
3	TYPE_STRING	string

Table 7: The typeof Function

4 <http://www.linfo.org/string.html>

**New
Concept**

`typeof(expression or variable)`

This function will return the type of an expression's result or the contents of a variable. If a variable had not been assigned a value the type will be 0. Expressions will return 1 for integers, 2 for floating-point numbers, and 3 for strings.

```
1 # types.kbs
2 print "integer 67 is type " + typeof(67)
3 print "floating-point 2.718 is type " + typeof(2.718)
4 print "string 'abcd' is type " + typeof('abcd')
5
6 print "variable a unassigned is type " + typeof(a)
7
8 a = 9
9 print "variable a containing " + a + " is type " +
  typeof(a)
10
11 a = 74.98
12 print "variable a containing " + a + " is type " +
  typeof(a)
13
14 a = "nine"
15 print "variable a containing " + a + " is type " +
  typeof(a)
```


Program 21: Data Types


```
integer 67 is type 1
floating-point 2.718 is type 2
string 'abcd' is type 3
variable a unassigned is type 0
variable a containing 9 is type 1
variable a containing 74.98 is type 2
variable a containing nine is type 3
```



Sample Output 21: Data Types

Converting Values from One Type to Another

BASIC-256 includes three functions that will convert values from one type to another. They are: **int()**, **float()**, and **string()**.

 New Concept	int (expression)
	<p>Return an integer value.</p> <p>If the expression is floating-point number the decimal portion will be removed and just the whole part will be returned. No rounding will occur.</p> <p>If the expression is a string, BASIC-256 will attempt to convert it to an integer (whole number). If the string does not contain a number then an error or warning will be displayed and zero will be returned.</p>

 New Concept	float (expression)
	<p>Return a floating-point value.</p> <p>If the expression is an integer, a floating-point number with the same value will be returned.</p> <p>If the expression is a string, BASIC-256 will attempt to convert it to a floating-point number. If the string does not contain a number then an error or warning will be displayed and zero will be returned.</p>

 New Concept	<code>string(expression)</code>
	Return a string value. If the expression is a numeric type (integer or float) then this function will return a string containing that number.


```
1 # intandstring.kbs
2
3 a = 9/2
4
5 # convert a to a string and concatenate
6 print "a is " + string(a)
7
8 # convert a to an integer
9 print "int(a) is " + int(a)
10
11 # round a to an integer
12 print "a rounded is " + int(a + .5)
```


Program 22: Converting Data Types

```
a is 4.5
int(a) is 4
a rounded is 5
```

Sample Output 22: Converting Data Types

Exercises:

 <p>Word Search</p>	<pre> d s u h l s b h k s f m a s s i g n m e n t f s u n a s s i g n e d n t s f m x y i l s v m m r w h l o f n n y a f g i i m o o b h u t r b t n n x c r a z s y i t y g t l v i t t n e a p p u e e j v m c f v b p e r g z f q w a u u l x o j e c a o d b j t e z f d r c r j z s n j n p d a </pre> <p>assignment, float, int, integer, shortcut, string, typeof, unassigned, variable</p>
---	---

 <p>Problems</p>	<ol style="list-style-type: none"> 1. Create a program with two variables 'a' and 'b' that you will assign to two numbers. Print the sum of a and b, the difference of a and b, the difference of b and a, the product of a and b, the quotient of a divided by b, and the quotient of b divided by a. Run the program with several values of a and b. What happens when a or b are set to the value of zero?
--	--