

Unit 2: Assigning Values to Variables

Skill Builder 1: Global Variables

In this first lesson for Unit 2, you will use variables in a program, assign values to variables, and discover the effect in the document.

Objectives:

- Use a variable in a program
- Learn the impact of creating a variable in a program to the rest of the document
- Explore the 'scope' of the arguments to a program

Storing a Value in a Variable Using :=

Often you need to assign a value to a variable within your program. Our example will use Heron's formula for calculating the area of a triangle from the lengths of its three sides, a, b, and c. This is a two-step formula and the computations will be assigned to variables:

Determine half of the perimeter, and assign the value to variable **s**:

$$s := 1/2(a+b+c)$$



Calculate the area, and assign the value to variable **area**:



$$area := (s(s-a)(s-b)(s-c))^{0.5}$$

The program is in the image to the right. Notice how the symbol $[:=]$ is used to assign the value of a computation to a variable; it is not just an equals sign.

$[:=]$ means "gets the value of..." or, more simply, "gets", so the statement

$s := 1/2 * (a + b + c)$ is read "s gets 1/2 the perimeter of the triangle."

On the TI-Nspire™ CX, this symbol, $[:=]$, is accessed by pressing  .

On a computer or the handheld, you can also type the two characters (colon and equals sign) separately. On the handheld, press  to access the colon. For the equals sign, press . On a handheld, it is simpler to just select the symbol $[:=]$.

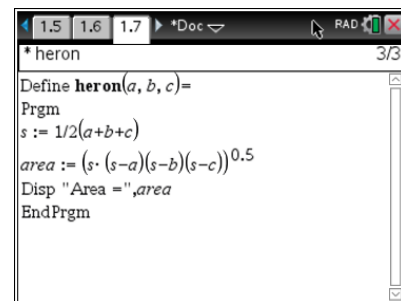
Storing a Value in a Variable Using →

You can use the **store** operator (\rightarrow) in place of the **gets** operator ($[:=]$) but the order is a bit different. See the program shown at the right. Notice the reverse order of the statements in which the computation comes first, then the store operator \rightarrow (**ctrl+var**), and then the variable. This is the order in which the statement is executed (left-to-right).

$\langle \text{expression} \rangle \rightarrow \langle \text{variable} \rangle$

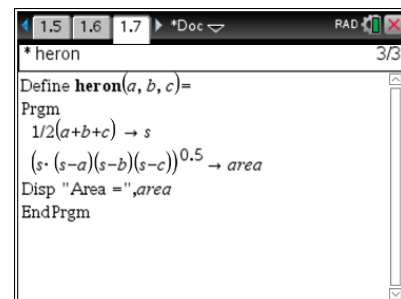
Either the **gets** operator ($[:=]$) or the **store** operator (\rightarrow) can be used in any assignment statement. Just remember the order of the statement for each method.

Both methods 'assign' a value (usually the result of a computation) to a variable so these are called assignment statements.



```

* heron
Define heron(a, b, c)=
Prgm
s := 1/2(a+b+c)
area := (s*(s-a)*(s-b)*(s-c))0.5
Disp "Area =",area
EndPrgm
  
```



```

* heron
Define heron(a, b, c)=
Prgm
1/2(a+b+c) → s
(s*(s-a)*(s-b)*(s-c))0.5 → area
Disp "Area =",area
EndPrgm
  
```

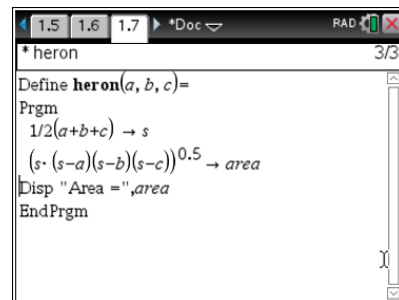
1. Enter the complete *heron*() program as shown in the image to the right.
2. When the code has been entered, 'Check Syntax & Store' the program by selecting **menu > Check Syntax & Store > Check Syntax & Store** (or use the shortcut **ctrl+B**).

Be sure to use a multiplication sign after the first *s* in the area formula. Without that multiplication sign, the TI-Nspire treats the expression *s*(as a function, and the error message 'Invalid implied multiply' appears when you 'Check Syntax & Store' (**ctrl+B** or **ctrl+R**).

3. Test the program with the known values of 3, 4, 5 to make sure you get a result of 6 for the area. (Why is the area 6?)
4. Next, in the Calculator app, select the **var** key and notice the list of variables. The variables **area**, **heron** and **s** appear in the list.
heron is the program name.
The variables **area** and **s** were created by the program.

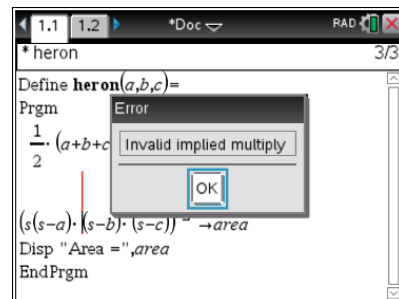
a, *b*, and *c* were also used in the program. Why are they not listed? The answer lies in the fact that *a*, *b*, and *c* are *arguments* to the program and, as *arguments*, they only exist *within* the program and are not created in the document. In fact, we really do not want the variables **area** and **s** in the document either. We'll learn how to fix this 'side effect' in the next lesson.

5. Remember to save the document (**ctrl+S**) so that you save the program. We will use this program in the next lesson.



```

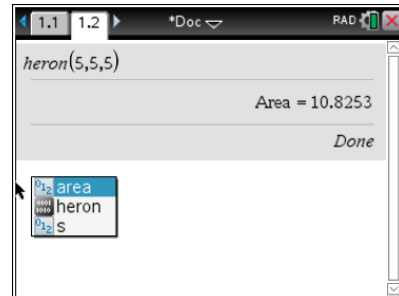
* heron
Define heron(a,b,c)=
Prgm
  1/2(a+b+c) → s
  (s·(s-a)(s-b)(s-c))0.5 → area
  Disp "Area =",area
EndPrgm
  
```



```

* heron
Define heron(a,b,c)=
Prgm
  1/2 (a+b+c
  (s(s-a)(s-b)(s-c))0.5 → area
  Disp "Area =",area
EndPrgm
  
```

Error
Invalid implied multiply
OK



```

* heron
Define heron(a,b,c)=
Prgm
  1/2(a+b+c) → s
  (s·(s-a)(s-b)(s-c))0.5 → area
  Disp "Area =",area
EndPrgm
  
```

heron(5,5,5)

Area = 10.8253

Done

area
heron
s