

## Programming the TI-89™

The TI-89 is programmable, with its own built-in programming language. A program is different from a script. A script is simply a series of commands that you would normally enter at the command line.

A program is a series of commands in the TI-89 language that it will execute one at a time, in order. It has more capabilities than a script. It may call for user interaction as it executes and, as a result, solve different problems each time it is run. A script, on the other hand, must be modified before being executed in order to solve different problem.

Here is a program to allow the user (you) to enter a value for  $x$ , and calculate and display the value of  $2x^2 - 3x + 7$ :

Program steps	What the steps do:
<b>Prog1()</b>	Title of program. the “( )” are necessary
<b>Prgm</b>	Indication that the program commands begin
<b>CLRIO</b>	Clear the screen
<b>Disp “Enter an x-value”</b>	Display (Print) the message on the screen
<b>Input x</b>	Allow user to enter a value for x
<b>x^2-3x+7→y</b>	Calculate the value of the expression and stor the answer in y.
<b>Disp y</b>	Display the value of y
<b>EndPrgm</b>	Indicates the conclusion of the program

Let’s enter this program in the calculator:

press **[APPS]**

then choose

**7: Program Editor**



NOTE: On the TI-89 Titanium ed™, Press **[APPS]**, use the cursor to navigate to the Program Editor, and press **[ENTER]**

- The calculator displays 3 options:  
**1:Open**, to work on a previous program (not the most recent one you have worked on)  
**2:Current**, to continue work on the most recent program that you have created or edited, and interrupted to do something else.  
**3:New**, to create a new program.
- Choose **3:NEW**. This screen is displayed:



- Use the down arrow key to move to the Variable box, and type **prog1**. The calculator will be in **alpha** mode when you start typing in the variable box. You will have to press **[alpha]** to leave **alpha** mode in order to type the 1. **prog1** will be the name of our program. Press **[ENTER]** twice. The following display will appear:



The first line is the program name. The ( ) are supplied by the calculator. You will see more on the parentheses later.

**Prgm** and **EndPrgm** are tags that the calculator inserts. They denote the beginning and end of the commands that make up the program. You will type the program lines (commands) between them.

- To type the program lines:  
Move to the blank line after **Prgm**, and type the following lines  
Press **[ENTER]** after each line to start a new line. The calculator displays new colon (:) at the beginning of each new line.

<i>Instruction (Command that you type)</i>	<i>Comments / aids / where to find it</i>
<b>ClrIO</b>	Type the instruction or find it in the <b>catalog</b>
<b>Disp "enter a value for x:"</b>	<b>Disp</b> is <b>F3:I/O   choice 2</b> (quote (") is <b>[2<sup>nd</sup>]   1</b> )
<b>Input x</b>	<b>Input</b> is <b>F3:I/O   choice 3</b>
<b>x^2-3x+7→y</b>	
<b>Disp y</b>	

- The display should look like this when you are done:



- Press **[2<sup>nd</sup>] – [QUIT]** to exit the program editor. The program will not be lost; it is saved in memory as you go along.

7. To execute the program, type **PROG1()** – include the **()** – on the command line and press **[ENTER]**.

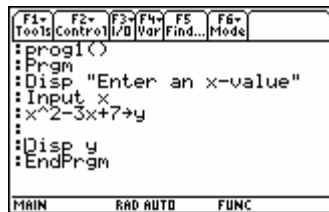
The screen should now look like:



Type in a value, and press **[ENTER]**. The calculator will calculate the value of **y**, and display it:



8. Let's fix up the display. Press **[APPS] - 7** to enter the program editor, and choose **1: Current**. The program is displayed on the screen. Put the cursor before the **D** in the **Display y** line, and press **[ENTER]** to insert a blank line.

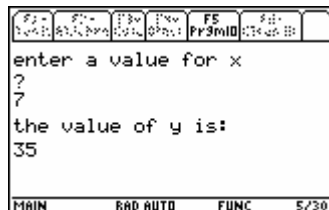


9. On the blank line, type the instruction: **Disp "the value of y is:"**. The Last several lines of the program are now:



Press **[2<sup>nd</sup>] - [QUIT]** to exit the program editor.

Run this program as before (Type **Prog1()** and press **[ENTER]**). Choose any input value for **x**. The screen output now resembles:



Press **[F5]: PrgmIO**, which returns the calculator to the **Home** screen. (**[F5]** toggles between **Home** screen and the program output screen.) You may also return to the **Home** screen by pressing **[HOME]**.

Any program is entered in this manner. Or course, there are many more commands that the calculator understands in its language.

Read about the following instructions in the TI-89 manual:

<u>Command</u>	<u>Page(s)</u>
<b>IF</b>	433, 283-284
<b>FOR</b>	427

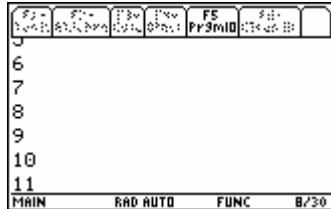
(If necessary, you may download the complete TI-89 user manual from the TI web site. This is a free download. Follow the link at the top of the Math 163 web page. ([www.mathbykoehler.com](http://www.mathbykoehler.com)) The manual is in Adobe PDF™ format, so you will also need the Adobe Reader™ software. This is also a free download. Follow the Adobe™ link at the left on the Lab 163 web page to obtain it.)

Enter the program editor, and create a new program: **Prog2**:

```
Prog2()
Prgm
ClrIO
5→a
11→b
For i, a, b
Disp i
EndFor
EndPrgm
```

Notice that the **EndFor** is automatically inserted to close the loop when you insert the “**For i, a, b**” command

Exit the editor ( **[2<sup>nd</sup>] | [QUIT]** ). Then type **prog2()** on the command line and press **[ENTER]** to execute the program. The display should be:



Notice that the **For - EndFor** sequence of instructions cycles through several times, using the values 5 through 11 in turn. Each successive time through the loop, the next value of  $x$  is used. When the last value (11) is processed, the loop does not repeat. Instead, the program continues on with the next instruction.

Notice that the first line of output partially scrolls off the screen. Space on the screen is somewhat limited.

Now create this new program: **prog3**.



Press **[2<sup>nd</sup>] | [QUIT]** to exit the Editor, and type **prog3()** on the command line, then press **[ENTER]** to execute the program. Execute it several times, using both positive and negative input values. Notice that  $x$  is negated only if it is a negative value. This is what produces the absolute value of your input value.

## Tracing a program

In some programs, the values of variables can become rather obscure. For example, consider the program segment:

- (1) **5→A**
- (2) **6→B**
- (3) **B-A→A**
- (4) **A+B→B**
- (5) **Disp B**

The following table is called a **TRACE** of the above program. It is intended to aid in keeping track of values.

There are always columns for the line # executed, each variable, and output when it is produced. Entries in the columns are made when there is a change in the value, or output is produced.

Line #	A	B	Output
1	5		
2		6	
3	1		
4		7	
5			7

TRACES for different programs may have different numbers of lines.. If loops are involved, loops, line numbers will repeat in the first column.

Here is a trace for a program segment containing a loop:

- (1) **For I, 3, 5**
- (2) **5\*I →A**
- (3) **Disp A**
- (4) **EndFor**

Line #	I	A	Output
1	3		
2		15	
3			15
1	4		
2		20	
3			20
1	5		
2		25	
3			25
1	6	<i>(I is now &gt; 5, so exit loop)</i>	

**NOTE:** In a **For** loop, the index variable (**i**, in this case) initially takes on the value of the number following the **i** in the **For** statement. A **For** loop *actually* terminates when it returns to the **For** statement and upon adding 1 to the index variable, it becomes larger than the second value in the **For** statement (5, in this case). When this occurs, execution transfers to the command following the **EndFor** statement.

Exercises: Do a TRACE (if you need to) and give the output for each of the following segments of programming code:

1.     **5→A**  
        **7→B**  
        **A+B→V**  
        **A+V→B**  
        **A-B→A**  
        **Disp "A=",A**

3.     **4→B**  
        **7→E**  
        **For I,B,E**  
            **Disp 4\*I**  
        **EndFor**

2.     **4→A**  
        **5→B**  
        **A^2-B→y**  
        **y^2-2y→z**  
        **Disp z**

4.     **0→S**  
        **For I, 2, 5**  
            **S+I→S**  
        **EndFor**  
        **Disp S**

5. If  $p$  dollars is invested for  $y$  years at  $r$  % interest compounded daily, the value of the investment,  $a$ , at the end of that time is given by the formula:

$$a = p * \left( 1 + \frac{.01r}{365} \right)^{365y}$$

Write a program called **value** that will prompt the user to enter values for **p**, **r**, **y**, and then calculate and display the final value of the investment, **a**.

Write the program below, and give the output for input of : **p = 10000, r = 4.55, y = 15**

6. The command **Prompt a** will cause the letter **a** to be displayed on the screen, followed by a “?”. The program then waits for you, the user, **to type a value and press [ENTER]**. The value you type is stored in **a**, and execution continues on to the next command in the program. Thus, the **Prompt** command can be more convenient to use in a program.

Write a program named **quad()** that will produce the real roots of a quadratic equation. It is acceptable for your program to produce an error message “**Non-real result.**” if the roots are not real. Here is the quadratic formula for your reference:

$$\text{Given the equation } ax^2 + bx + c = 0, \text{ the roots are given by the formula: } x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Your program must allow the user to input values for **a**, **b**, and **c**. Use the **Disp** and **Input** commands, or else the **Prompt** command to allow the user to enter values for **a**, **b**, and **c**.

It should then display the message “**roots are**”, and finally calculate and display the roots of the equation in exact (fraction and/or radical) form.

A sample run of the program might look like this (user input is underlined):

```
a? 2
b? 5
c? 3
roots are:
-1
-3/2
```

*You will receive extra credit* if your program uses an **If** or **If - Then** statement to avoid displaying the error window containing the message “**Non-real result**” when the errors occurs. Instead, it should display the message “**Roots are not real**”, and do no further calculation. Before adding this to your program, be sure your program works correctly for real roots, and produces the error window when the roots are imaginary. You recall that the roots will be imaginary only when the discriminant ( $b^2 - 4ac$ ) is less than zero. *i.e.*  $b^2 - 4ac < 0$ . The “**If**” statement is explained on page 433, and on pages 283 – 284 of the complete TI-89 manual.