Unit 4: **Skill Builder 1: Loops and the For(…) loop**

In this first lesson for Unit 4 you will learn about the loop concept and the structure and use of the **For(…)** loop.

Objectives:

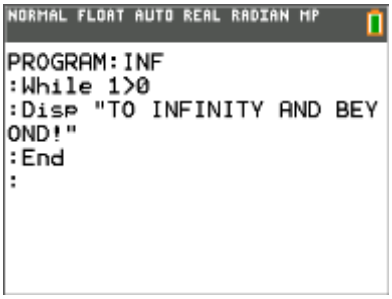
- Understand loops.
- Use the **For(…)** loop to generate a list of values.

Teacher Tip: There are three fundamental loops in TI-Basic: **For**, **While**, and **Repeat**. A loop structure gives a program the ability to process a set of statements over and over, either iterating over a sequence of values (as in the **For** loop) or until a specific condition is met (or not) as in **While** and **Repeat**. In this unit each lesson deals with just one of these structures. It is also possible to use the archaic **Lbl** and **Goto** statements to make a loop but that leads to bad habits and could result in program errors if not done properly, so we avoid referencing these statements at all. However, the **Lbl** statement is used and, in fact, required in conjunction with the **Menu** statement to design custom menus within a program. For conditional statements and loops, though, the **Goto** statement is not needed at all.

Programs can become complicated because it becomes necessary to mix in all the control structures, (**If** statements and loops) in a program to satisfy more complex algorithms. This is what makes programming FUN!

Loops

A loop is a method of repeating a set of statements. All programming languages have at least one looping structure. The loop structure has a way of going backwards in a program to a previous spot. TI-Basic has three different types of loops. An infinite loop never ends.



An *infinite loop*. Why?

To 'break' (stop) a running program, press [ON]. You'll see the options 'Quit' and 'Goto'. 'Quit' returns you to the HOME screen and 'Goto' takes you into the Program Editor to the spot where the program stopped.

The Three TI-Basic Loops are:

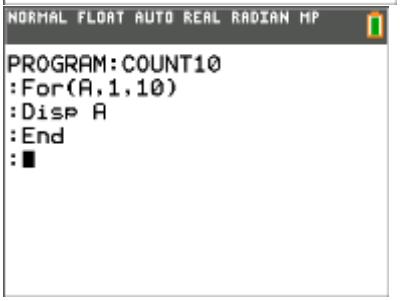
For( ) … **End**
While <condition is true> … **End**
Repeat <*until* condition is true> … **End**

This rest of this lesson deals only with the **For**( ) loop.

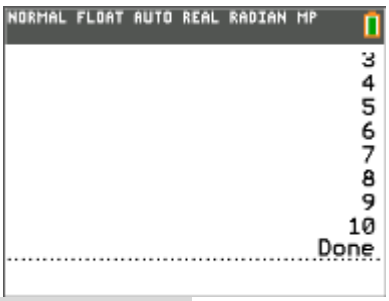
The For(…) Loop

Structure: **For**(*variable*, *starting value*, *ending value*)
loop body
End

Example: **For**(A,1,10)
Disp A ← *loop body*
End



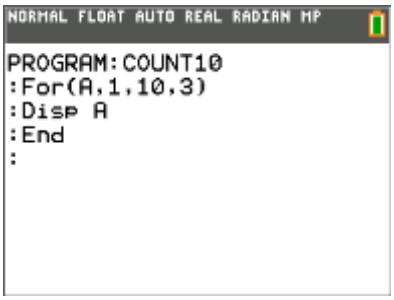
Note:
*The **For**( ) statement requires a variable (the loop control variable), a starting value and an ending value, separated by commas. The starting and ending values can be variables. The loop body can be as many statements as needed but should not change the loop control variable. The loop will run from your starting value to your ending value with a standard increment of 1.*



Teacher Tip: The loop ends when the variable *exceeds* it's ending value. If you **Disp A** after the loop above ends you will see that its value is 11, not 10.

For Loop With Increments Other Than 1

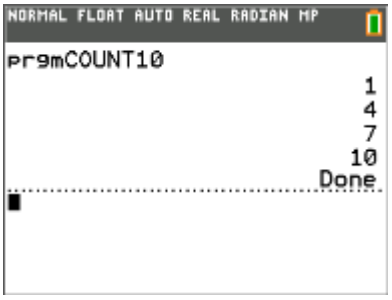
There is an optional fourth argument for the **For( )** statement: the *increment*. The increment is the value by which the loop control variable increases with each iteration of the loop. The default value is 1.



Teacher Tip: When the loop increment is negative the loop ends when the loop variable is *less than* the ending value.

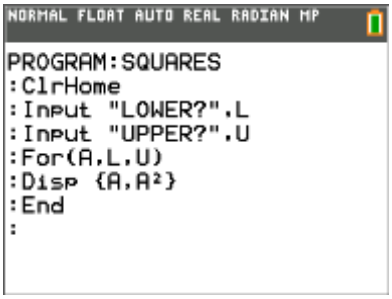
For(A,1,10,3) starts with **A**=1, then adds 3 to **A** each time the loop repeats. The loop stops when **A** is greater than 10. The increment can be a negative number.

For(B,10,0,-1) counts down from 10 to 0.



Programming with For(…)

Let's write a program that displays a table of numbers and their squares. The user can enter the lower and upper bounds of the range of numbers. The tricky part is to **Disp**lay the pairs of numbers on the same line! We can do this using **lists**.



Note:
L *and* **U** *are used to represent* **L***ower and* **U***pper. The* **For( )** *statement uses the values of* **L** *and* **U**. *The list brackets are on the parentheses keys. Press* 2nd (*and* 2nd) *for the brackets.*

Teacher Tip: Displaying a list is a convenient method to display more than one value on a line on the HOME screen.

Run the program and enter a lower and an upper bound for the table.

If the lists go by too quickly then consider adding a **Pause** statement after the **Disp** statement and before the **End** statement.

Challenge:
Use an **If … Then… End** structure to **Pause** every 5 pairs of numbers. Recall the divisibility technique from the previous unit.

Sample Answer:

```
:If (A-L+1)/5=int((A-L+1)/5)
:Pause "Press Enter"
```