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.

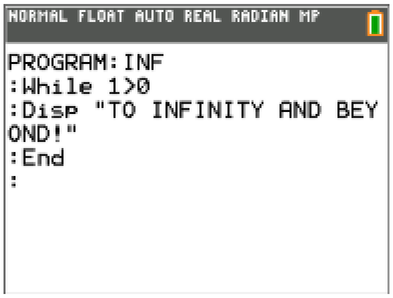
Objectives:

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

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.

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.



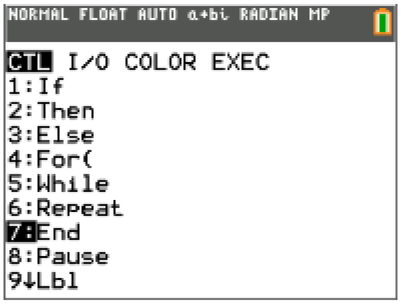
```
PROGRAM:INF
:While 1>0
:Disp "TO INFINITY AND BEY
OND!"
:End
:
```

An *infinite loop*. Why?

The Three TI-Basic Loops are:

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

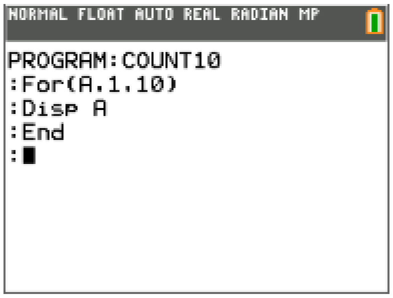
This rest of this lesson deals only with the **For(**) loop.



The For(…) Loop

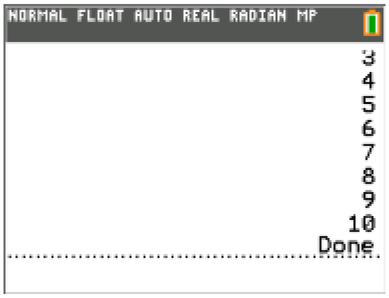
Structure: **For**(*variable*, *starting value*, *ending value*)
loop body
End

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



```
PROGRAM:COUNT10
:For(A,1,10)
:Disp A
:End
:
```

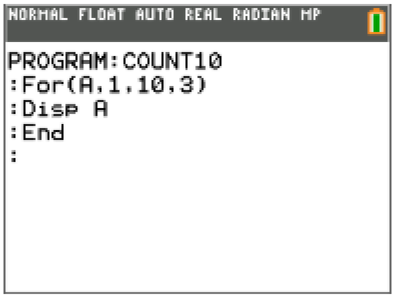
Note:
*The **For**( ) statement requires a variable (the loop control variable), a starting value and an ending value, separated by commas. The starting and ending values can be variables. The loop body can be as many statements as needed but should not change the loop control variable. The loop will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



```
                                3
                                4
                                5
                                6
                                7
                                8
                                9
                               10
                             Done
```

For Loop With Increments Other Than 1

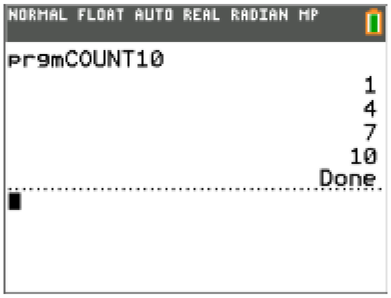
There is an optional fourth argument for the **For( )** statement: the *increment*. The increment is the value by which the loop control variable increases with each iteration of the loop. The default value is 1.



```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM:COUNT10
:For(A,1,10,3)
:Disp A
:End
:
```

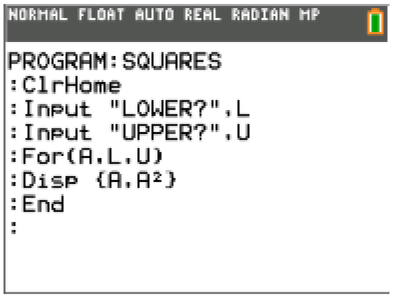
For(A,1,10,3) starts with **A**=1, then adds 3 to **A** each time the loop repeats. The loop stops when **A** is greater than 10. The increment can be a negative number.

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



```
NORMAL FLOAT AUTO REAL RADIAN MP
prgmCOUNT10
                    1
                    4
                    7
                   10
                 Done
```

Programming with For(…)



```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM:SQUARES
:ClrHome
:Input "LOWER?",L
:Input "UPPER?",U
:For(A,L,U)
:Disp {A,A²}
:End
:
```

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* <u>*values*</u> *of* **L** *and* **U**. *The list brackets are on the parentheses keys. Press* [2nd] [(] *and* [2nd] [)] *for the brackets.*

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.