



Unit 4: Loops

Skill Builder 3: The Repeat...End loop

In this third lesson for Unit 4 you will learn about the **Repeat...End** loop. We'll compare it to the **For...** loop and even show why it is more powerful.

Objectives:

- Learn the structure and logic of the **Repeat...End** loop.
- Compare it to the **While...End** loop.
- See how **Repeat...End** is used in programming the Fibonacci sequence.

Teacher Tip: Although the **Repeat** condition is written at the top of the loop it is not actually tested until the bottom of the loop! A **Repeat** loop therefore is always executed at least once.

There are both **While** and **Repeat** loops in the language because they behave slightly differently.

The Repeat... End Loop

The **Repeat...End** loop will continue looping as long as its <condition> is False. This is the exact *opposite* if the **While** loop behavior and that's not the only difference! It looks like this, which looks pretty much the same as the **While** structure:

Repeat <condition>

<loop body>

End

The programs on the right have the same output. What are the differences?

The <condition> is a logical expression such as **X>0**.

The <loop body> is any set of statements, including other loops and **If** structures. It **is processed once** and then continues **until** the <condition> is true so it behaves more like this:

Repeat

<loop body>

until <condition> **is true** **End**

But there's no 'until' keyword in TI Basic. It's implied. Even if the <condition> is true to start with, the loop body is still processed once because the condition is tested at the bottom of the loop.

In a **While** loop it's a great idea to 'initialize' the variable(s) that your loop relies upon. In a **Repeat** loop this will happen inside the loop body so initializing is not necessary.

As with **While**, somewhere in the <loop body> there should be a statement that will have an effect on the <condition> so that the loop will eventually end and statements after the loop can be processed. Usually this statement is near the bottom of the <loop body>.

```
NORMAL FLOAT AUTO REAL Radian MP
PROGRAM: REPEAT
:0→A
:Repeat A>10
:Disp A
:A+1→A
:End
:■
```

```
NORMAL FLOAT AUTO REAL Radian MP
PROGRAM: WHILE
:0→K
:While K≤10
:Disp K
:K+1→K
:End
```



10 Minutes of Code

TI-84 PLUS FAMILY

Programming the Fibonacci Sequence

Let's write a program that displays the *Fibonacci* sequence up to a certain value. You can research the *Fibonacci* sequence if you've never heard of it.

The output of the program is shown to the right. Can you write this program without peeking below?

We start with a **Prompt** statement to get an upper bound value from the user. The first two Fibonacci numbers are 1 and 1 so we'll store those values in the variables **A** and **B**. These variables are going to be used to calculate the rest of the Fibonacci numbers (up to **N**).

Then we begin the **Repeat** loop using the condition **A>N**, meaning 'until **A** is greater than **N**'. In the loop we first display the current two numbers.

Finally, we calculate the next two Fibonacci numbers and **End** the loop.

These last two loop statements show that **A+B** is being stored in both **A** and **B** and it appears that **A** and **B** are getting the same value. **This is not the case!** Try it with 1 and 1. The first statement stores 1+1 in **A**, making **A** 2. The second statement stores 2+1 in **B** making it 3. Try it yourself - 'play computer'!

Run the program with several different input values. Does it behave as planned?

Try changing the **Repeat** condition to **B>N**. What is the effect? How can we modify the program to display the 'correct' set of numbers, stopping when exactly the largest Fibonacci number less than **N** has been displayed?

UNIT 4: SKILL BUILDER 3

TEACHER NOTES

```
NORMAL FLOAT AUTO REAL Radian MP
PROGRAM: FIBONACC
N=?5
1
1
2
3
5
8
.....
Done
```

```
NORMAL FLOAT AUTO REAL Radian MP
PROGRAM: FIBONACC
:Prompt N
:1→A
:1→B
:
.....
NORMAL FLOAT AUTO REAL Radian MP
PROGRAM: FIBONACC
:Prompt N
:1→A
:1→B
:Repeat A>N
:Disp A,B
:
.....
NORMAL FLOAT AUTO REAL Radian MP
PROGRAM: FIBONACC
:Prompt N
:1→A
:1→B
:Repeat A>N
:Disp A,B
:A+B→A
:A+B→B
:End
:■
```