Unit 3: Conditional Statements | **Skill Builder 3: If…Then…Else…End; Divisibility**

In this third lesson for Unit 3 you will learn about another form of the **If** statement in TI Basic and the value of understanding numeric algorithms.

Objectives:

- See the structure of the **If…Then…Else…End** statement.
- Learn the test for 'whole-ness' (is a value an integer?).
- What is an **Algorithm**?

Teacher Tip: This lesson introduces students to the concept of an **algorithm**. This principle is critical to successful programming. While we do not delve too deeply into an algorithmic approach, the idea of having a plan before writing a program is crucial, since writing a program first requires an understanding of the task and knowledge of the programming commands and their purpose. Then we can devise an algorithm to solve the problem.

The three forms of the **If** statement are concluded in this lesson with a discussion of **If…Then…Else…End**. The syntax is important: each of the four keywords appear as separate statements in the program and only the **If** can have something follow it (the *condition*).

About If…Then…Else…End

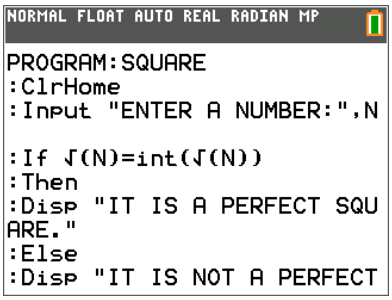
In the previous activity you learned about the **If…Then…End** statement. There are times when we'll need to take two different courses of action depending on a condition. The structure of this new **If…Then…Else…End** statement is similar:

```
If  <condition>
Then
    <True block>
Else
   <False block>
End
```

Note:
Then, **Else**, *and* **End** *are on line by themselves.*
The <True block> *is the set of statements that will be executed when the <condition> is true.*
The <False block> *is the set of statements that will be executed when the <condition> is not true.*
Therefore, one or the other of these two blocks will be executed.

Programming with If…Then…Else…End

We'll write a program to tell whether or not an entered number is a perfect square. A 'perfect square' is the square of an integer, such as 25 (5^2). The method used here is to take the square root of the number and test to see if it is an integer. The program listing is at the right.



Note:
Use the [√] key on the keypad for the square root symbol.
If, **Then**, **Else**, *and* **End** *are all on the* PRGM CTL menu.
The **int( )** *function is on the* MATH NUM *menu. The function returns an integer, so, for example,* int(6.56) → 6 int(9.999) → 9 int(-2.01) → -3

```
NORMAL FLOAT AUTO REAL RADIAN MP

PROGRAM:SQUARE

:If √(N)=int(√(N))
:Then
:Disp "IT IS A PERFECT SQU
ARE."
:Else
:Disp "IT IS NOT A PERFECT
 SQUARE."
:End
```

Try your own **int( )** examples on the HOME screen. Remember to close parentheses for the **int( )** function and the square root function. You'll only use UPPERCASE letters in your **Disp** statements on the calculator. You can enter lowercase letters through **TI-Connect CE**.

Teacher Tip: Suggest to students that they could improve the output of the program by displaying the number entered in the sentence rather than just the word "It". This would require the use of **Output( )** rather than **Disp**.

Algorithms

Techniques such as that used in this exercise are known as '**algorithms**'. An **algorithm** is a procedure or formula for solving a problem.
A recipe for baking a cake is an algorithm. If you follow the recipe you will have cake. All mathematical formulas, such as the formula of a triangle (A=B*H/2) are algorithms: they give you a method to determine a new value based on existing values. Algorithms such at the 'perfect square' technique above are important problem-solving tools. Learning common computer algorithms such as this will make your programming experience very rewarding.

Here's a box cake recipe…

1. Prepare cake batter per directions on box.
2. Bake as directed on package—two cake layers.
3. Cool in pans 10 minutes.
4. Remove from oven to wire racks.
5. Let cool completely.
6. Beat pudding mixes and milk with whisk 2 minutes.
7. Immediately spread over tops of cakes.
8. Stack cake layers.
9. Frost with whipped cream.
10. Enjoy!

As in a program, the baker follows the steps from beginning to end. At the end there's a delicious cake to enjoy. When a computer follows the steps in a program (the algorithm) the desired result is achieved. And, in fact, there's a branch of computer science that deals with PROVING that an algorithm will give the desired result. It's similar to proving mathematics theorems.

Teacher Tip: Here's an example of a computer algorithm:
To round a number to the nearest whole number use:

A = int(A+0.5)

Why does it work? When the decimal part of A is less than 0.5, adding 0.5 to the number keeps the number lower than the next higher integer. Then the int( ) function truncates the number at the decimal point, leaving just the integer part.
Here's an example:
suppose A=3.4 A+0.5 = 3.9 int(3.9) = 3 (rounds down)

suppose A=3.7 A+0.5 = 4.2 int(4.2) = 4 (rounds up)

But of course, the TI-84 also has a built-in round( ) function on the MATH NUM menu!