

Unit 1: Program Basics and Displaying on Screen

Skill Builder 1: Using program editor and syntax

This is the first of three ‘Skill Builders’ in Unit 1. At the end of this unit, you will use the skills you have learned in these Skill Builders to create a more complex program. This is your first lesson in learning to code with TI Basic. TI Basic is a programming language that can be used to program on the TI calculators. While the structure and syntax (grammar) of TI Basic is simpler than other modern languages, it provides a great starting point for learning the basics of coding. Let’s get started!

Objectives:

- Use the TI Basic Program Editor to create and run a simple program.
- Use the program menus to select and paste commands into a program.
- Run a program.

Teacher Tip: B.A.S.I.C. is one of the original programming languages that was designed for teaching and learning programming. It is an acronym of **B**eginners’ **A**ll-purpose **S**ymbolic **I**nstruction **C**ode. **TI Basic** is based upon this language.

Turn on your TI-84 Plus CE and press the **PRGM** key.

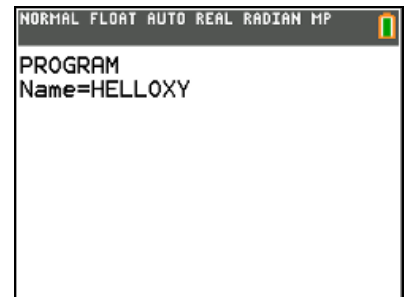
- Select **NEW** using the arrow keys.
- Select **Create New** by pressing **ENTER**.



Name your program.

Our program name will be **HELLOXY**. It can be any legal name*. Press **ENTER** after typing the name. You are now in the Program Editor. Each line begins with the colon character (:).

**A legal name must: be up to 8 characters long, start with a letter, include only uppercase letters and numbers, with no spaces; and be unique.*



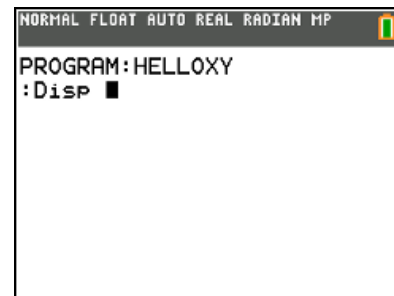
Teacher Tip: If you use a name that has already been used then you will edit that program rather than create a new one.

This program will display a simple message on the home screen of your calculator.

Selecting a programming command from the Program Menu.

The **[PRGM]** key now contains new menus that contain the commands that are used in TI Basic. If you want to use one of the commands, you *must* select it from this menu rather than type it on the screen.

1. Press the **[PRGM]** key
2. Choose the **I/O** menu using the arrow keys. This menu contains all of the commands affecting Input and Output.
3. Select **Disp**. The word will be pasted into your program at the current cursor position. The **Disp** command will display something on the HOME screen.



Teacher Tip: You cannot type in the programming commands. The commands are also not editable. All keywords in a program are selected from the menus. The text displayed is actually just a readable symbol (token) for the programming command.

Teacher Tip: Some of the other keys on the calculator behave differently while using the Program Editor: The **MATH** key allows you to select one of the math functions to use in a program. The **CATALOG** key contains a list of ALL calculator functions in alphabetical order. The **MODE** key allows you to select a mode setting so that the program will change the mode to that setting. There are other keys that exhibit similar behavior.

Teacher Tip: Some keys, such as **Y=** or **GRAPH**, will take you out of the Program Editor and into its own environment. Fear not. Just press **[PRGM]** > EDIT to get back to editing the program that you select from the list.

Type a greeting in double quotation marks.

This greeting is called a *string*, which is a group of characters that are “strung together”.

- Your string must start and end with quotation marks. Without the quotes, the program thinks you mean something completely different.
- Make your life easier: Press **[2nd]** **[ALPHA]** to turn on the alpha-lock while you type in the string.



Teacher Tip: The **[CLEAR]** key clears an entire line of code. It cannot be undone.

Teacher Tip: To insert a blank line place the cursor at the beginning or end of a line and press **[INSERT]** (**2nd DEL**) and then press **[ENTER]**.

Teacher Tip: To delete a line press **[CLEAR]** and then press **[DEL]**. There is no copy and paste.

Your program is complete! Let’s run it now. There is no need to ‘save’ with TI Basic; the program is preserved as you type it in. That’s why we named the program first.

To run the program:

Press [quit] (2nd[MODE]) to return to the HOME screen.

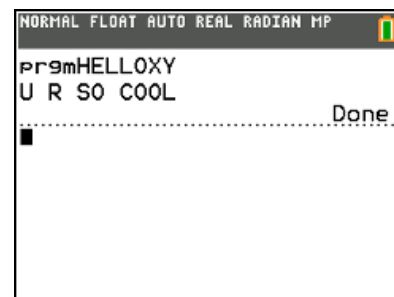
1. Press [PRGM].
2. Under the **EXEC** ('execute') menu, select your program.
3. Press [ENTER] to paste the program name on the HOME screen.
4. Press [ENTER] again to begin the run.



Your text message is displayed on the HOME screen.

You can edit your program, too:

1. Press [PRGM].
2. Choose the **EDIT** menu using the arrow keys.
3. Select your program and press [ENTER].



Teacher Tip: Pressing [ENTER] after a program finishes running will re-run the program (because [ENTER] processes the last command entered on the HOME screen).

Teacher Tip: If a program generates an **ERROR** message then there is something wrong in the program. There are two options under the error: **1:Quit** and **2:Go To**. **Quit** takes you to the HOME screen and **Go To** takes you into the Program Editor to the place in the program where the error occurred. This may or may not be the actual place that causes the error.

Teacher Tip: A common error is **SYNTAX**. **Syntax** is a synonym for grammar; there's something wrong with the *structure* of the statement.

Teacher Tip: To delete a program use the Memory Management utility (2nd + > **Memory Mgmt/Delete...**). Select **PRGM...** and press the delete key on the program you wish to remove. You will see an 'Are you sure?' warning message.

Unit 3: Conditional Statements

Skill Builder 1: Conditions and the If... Statement

In this first lesson for Unit 3 you will learn about conditions and the introduction to the **If** statements available in TI Basic.

Objectives:

- Learn about conditions.
- Use the 'simple' **If...** statement to conditionally process another statement.

If...Then statements are used to process a block of statements only when a *condition* is true or false. Before visiting the **If...Then** collection of statements, let's get an idea of just what a *condition* is.

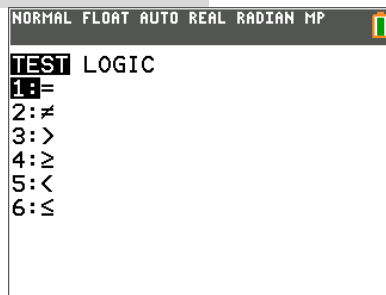
Teacher Tip: Relational and logic operators also have an order of operations and fit into the arithmetic order of operations as well.

Here's a complete list of the TI-Basic order:

Priority Level	Functions
1	Functions that precede their argument (such as $\sqrt{\quad}$ or $\sin(\quad)$, except for negation)
2	Functions that follow their argument (such as 2 or !)
3	\wedge and $\times\sqrt{\quad}$
3.5	Negation
4	nPr and nCr
5	Multiplication, division, and implied multiplication
6	Addition and subtraction
7	The relational operators =, \neq , <, >, \leq , \geq
8	The logic operator and
9	The logic operators or and xor
10	Conversions such as \blacktriangleright <i>Frac</i>

Conditions and the [TEST] Menu

Conditions are expressions that evaluate to 'true' or 'false'. Such expressions are either true or false; they cannot be both or neither. The relational operators and the logical operators are both found on the [TEST] menu (2nd [MATH]). The TEST menu contains the **relational operators**. The LOGIC menu contains the **logical operators**. The = sign is used to form a condition, not an assignment.



Examples of some conditions:

- | | | |
|-------------|---------------------|-------------------------|
| $3 > 5$ | $XY > 0$ | $X = 5$ or $Y = 5$ |
| $X + 4 > X$ | $B^2 - 4AC = 0$ | $X/2 = \text{int}(X/2)$ |
| $X \neq Y$ | $X > 0$ and $Y > 0$ | not($X > 0$) |



Conditions on the HOME Screen

You can enter conditions right on the HOME screen to see how they are computed.

Observe that 1 stands for *true* and 0 stands for *false*.

Note: when you use a variable in a condition the calculator evaluates it using the current value stored in the variable.

```
NORMAL FLOAT AUTO REAL RADIAN MP
3>5
.....0
3≠5
.....1
X+1=X
.....0
X+1>X
.....1
```

Teacher Tip: We introduce the ‘primitive’ or ‘simple’ If statement below because it is an easy way of conditionally executing only one statement (if that’s all that is needed). But the more versatile **If...Then** statement discussed after is preferred because it is clearer to a reader of the program what the programmer is trying to do.
The **:If <condition>** (without **Then**) processes only the next statement when the condition is true, otherwise it is skipped.

Programming with the ‘Simple’ If... Statement

Try this program:

```
:Prompt A
:If A>0           [If is in the [PRGM] CTL menu. '>' is in the [TEST] menu]
:Disp "A IS POSITIVE"
:Disp "A IS NOT POSITIVE"
```

```
NORMAL FLOAT AUTO REAL RADIAN MP
PRGMIFSTMT
A=?5
A is positive
A is not positive
.....Done
```

Run the program several times entering both positive and non-positive numbers and observe the output. What can you learn?

When the condition $A > 0$ is true, the statement that follows **If** is executed, otherwise it is simply skipped. But the statement that displays “A IS NOT POSITIVE” is always executed, which is not correct! See the screen at the right. We’ll fix this soon.

This ‘simple’ **If...** is a concise way of skipping **one** statement based on a condition (when it is FALSE).

Editing the If... Statement

Let’s correct the program above by adding another **If...**

1. Place the cursor on the second **Disp**.
2. Press \bar{i} and press [ENTER] to insert a blank line.
3. On that blank line add **If A<0**.

```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM: IFSTMT
:Prompt A
:If A>0
:Disp "A is positive"
:If A<0
:Disp "A is not positive"
```

Quit and run the program several times using both positive and negative numbers and 0, too!

Does your program work correctly in all cases? If, not, try to fix the problem.

In this third lesson for Unit 2 you will learn about using expressions and storing values in variables within programs.

Objectives:

- Learn about programming mathematical expressions.
- Understand order of operations.
- Realize the difference between mathematical variables and computer program variables.
- Evaluate **expressions**.
- **Store** the results of expressions in variables.

Expressions

Items such as A^2 and $A+B$ are called *expressions*. Expressions can be found in mathematical formulas. For example, the **formula** for the area of a triangle is $A = \frac{1}{2} \times B \times H$. The **expression** is $\frac{1}{2} \times B \times H$.

A program evaluates an expression using the current values of all variables and gives the result as a numeric value. Expressions are evaluated using the *algebraic order of operations*.

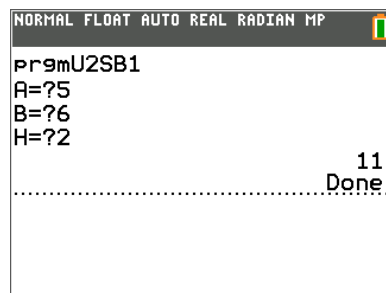
Try the program to the right which computes the area of a *trapezoid* with bases **A** and **B** and height **H**.



```
NORMAL FLOAT AUTO a+bI RADIAN MP
PROGRAM:U2SB1
:Prompt A,B,H
:Disp 1/2*(A+B)*H
:█
```

You cannot use variables such as **B1** and **B2**. The calculator computes these as the expressions $B \times 1$ and $B \times 2$ and may cause an error when used incorrectly.

You also cannot use variables with more than one letter such as **AB**. As stated earlier, this means $A \times B$. This is called *implied* multiplication because the multiplication sign between the variables is 'implied' or assumed.



```
NORMAL FLOAT AUTO REAL RADIAN MP
PrgrmU2SB1
A=?5
B=?6
H=?2
..... 11
..... Done.
```

Mathematical Expressions and Computer Expressions

While there are many similarities in the *appearance* of expressions in mathematics and computer programs there are also important differences. The most significant difference is that in a mathematical expression the variables stand for 'unknown' numbers and are replaced with numbers when needed. In a computer expression the variables are *names* for numbers.

In mathematics we use formulas to state a relationship between things such as area and lengths. In programs we use the expressions to do the calculations and the variables' values are used to compute a result. When we *write* the program we type the expression but when we *run* the program the expression is computed and creates a result to be used later.

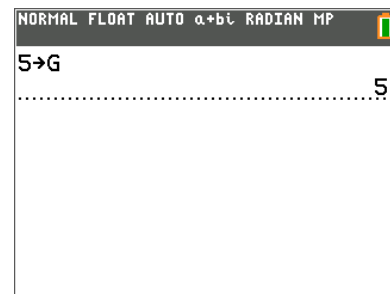
Teacher Tip: One of the more confusing statements in a program for beginners is $x+1 \rightarrow x$. It's pretty clear in the syntax of TI-Basic that 1 is being added to the variable x and then being stored in variable x, so the x on the left and the x on the right represent different values. This statement is known as a 'counter' because each time it is processed (in a loop) it increases x by 1. But in other languages such as B.A.S.I.C and Lua (and many more) the statement appears as $x=x+1$ which is clearly a false assertion in a mathematical sense but is perfectly good in a program!

Storing Values in Variables: The Assignment Statement

The **STO** operator is used to **store** (assign) the result of an expression into a variable.

Pressing the **STO** key always displays the symbol \rightarrow .

After pressing **ENTER**, the HOME screen displays the result of the expression *and* the variable **G** now contains the value **5**. In front of the arrow must be a **value** or an **expression** that produces a **value**. This is called the *assignment statement* because it assigns a value to a variable. The symbol after the arrow must be a **variable**.



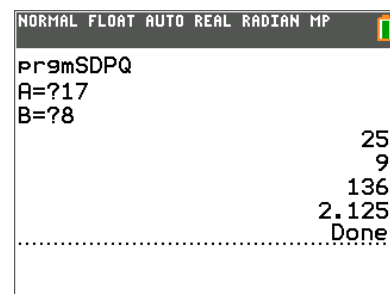
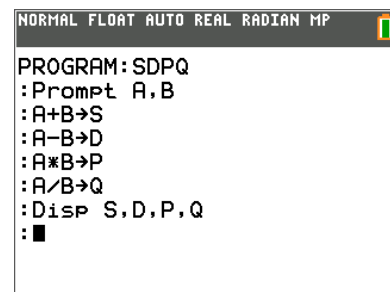
Programming with Assignment Statements

Let's write a program that asks you to enter two numbers, stores their sum, difference, product, and quotient in four variables and then displays the results.

1. Start a new program. Our program name is **SDPQ**.
2. Prompt for two variables **A** and **B**.
3. Store the four expressions in four *other* variables **S**, **D**, **P**, and **Q**.
4. Display **S**, **D**, **P** and **Q**.
5. Run the program.

Enter a value for **A** and another value for **B**.

Be sure to use numbers for which you can confirm that the calculations were performed correctly! This is known as 'testing' the program.



Note: TI Basic's assignment statement is unique in that the expression comes first, then the 'store' operator, then the variable. This makes it easy to read from left to right. In most other languages the order is the opposite, such as $S=A+B$. This is backwards because the computer evaluates the expression on the right first and then stores the result in the variable on the left.

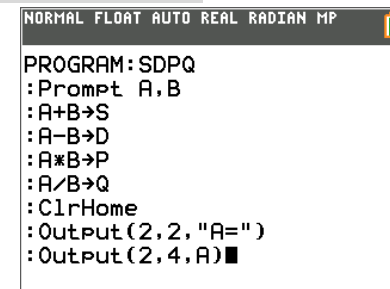
Teacher Tip: There are many advantages to storing values in variables. Using variables makes it easier to display the result. It also allows the variable to be used in subsequent calculations. If you find you are using the same number in a lot of different places in your program then you can store the number in a variable and use it everywhere that it occurs in your program.

Making Improvements

You can improve the program using **Output** (rather than **Disp** to show the original values entered *and* the four results *properly labeled*. You try it!

To the right is a snippet of code and a screen of the partially completed program running. There's still some work to do here so we'll leave it to you to complete the program.

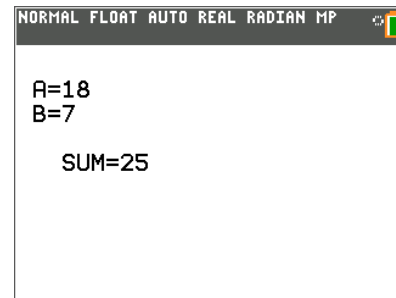
Remember to include a **Pause** statement at the end of the **Output** statements to prevent the 'Done' message from spoiling the display.



10 Minutes of Code

TI-BASIC

Remember that the **TI-84 Plus** and the **TI-84 Plus C/CE** have different HOME screen sizes (as well as different GRAPH screen sizes) so plan accordingly. The TI-84 Plus HOME has 16 characters per line and 8 lines, while TI-8 Plus C/CE HOME has 26 characters per line and 10 lines.



```
NORMAL FLOAT AUTO REAL RADIAN MP
A=18
B=7
SUM=25
```

Teacher Tip: Challenge students to experiment with other mathematical formulas and use assignment statements to store the result to make it easier to **Display** or **Output** the results. In the Application for this unit students are given three mathematical formulas to compute.

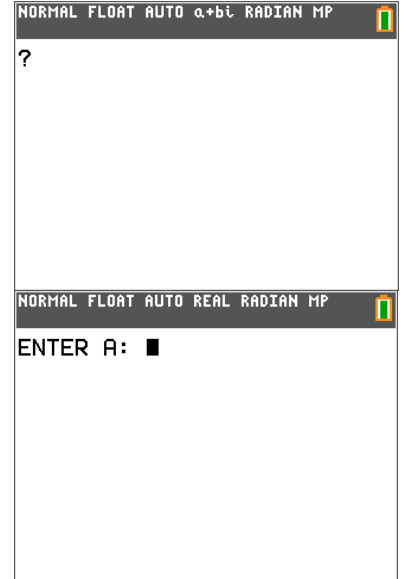
In this second lesson for Unit 2 you will learn about the different forms of the **Input** statement.

Objectives:

- Use the TI Basic **Input** statement to assign a value to a variable.
- Perform calculations within **Disp** statements.
- Use the GRAPH screen to get input to two variables at once.

The Simple Input Statement

The **Input** statement is followed by *only one* variable name to ask the user to enter a value for that variable. Unlike **Prompt**, **Input** only places a question mark on the screen as you can see in the program example at the right.



The Improved Input Statement

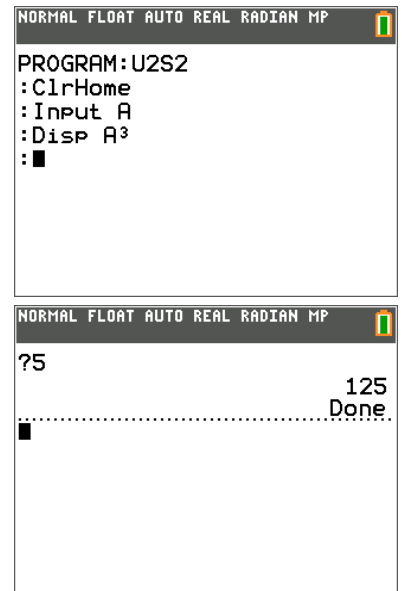
This type of **Input** statement can display a custom message that is displayed before waiting for a value for the variable. The structure of the Input statement with a message is:

Input "YOUR MESSAGE HERE",V

Note: This statement does not provide any question mark or other punctuation, so if you want one then it must be included inside the message.

Programming with Simple Input

1. Start a new program.
2. For the first statement of the program use the **Input** statement found on the **PRGM I/O** menu.
3. After the **Input** command type the name of the variable you want your program to use. Here we use the variable **A**.
4. Use the **Disp** statement to display the cube, **A³**; type the **A** then use the **MATH** menu to get the cute, small 'cubed' exponent.
5. Quit the editor and run the program.
6. After the "?" type any number and press **ENTER**.
7. The program displays the cube of the entered number and ends.



Programming with *Improved Input*

1. Edit the program you started earlier.
2. Place the cursor on the variable after the word **Input**.
3. Press [INS].
4. Type a message to display. Remember to use [A-LOCK] and quotation marks.
5. Include a punctuation mark at the end of the message (inside the quotes).
6. Place a comma after the closing quote and before the variable.
7. Keep the **Disp** statement that displays A^3 .

8. Quit the editor and run the program.
9. After the message type any number and press [ENTER].
10. The program displays the cube of that number and ends.

```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM:U2S2
:ClrHome
:Input "ENTER A: ",A
:Disp A^3
:█
```

```
NORMAL FLOAT AUTO REAL RADIAN MP
ENTER A: 25
15625
Done
:█
```

Using Input without a Variable

If you use the **Input** statement *without* a variable then the program will display the GRAPH screen with a free-floating cursor.

When you press [ENTER] the program continues and the variables **X** and **Y** contain the values that you pointed to on the GRAPH screen!

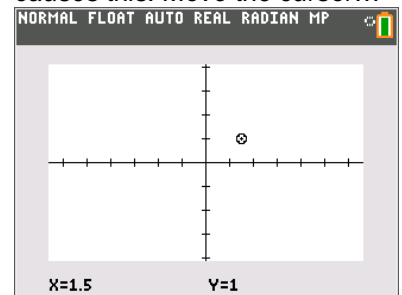
You can then use these two variables in the rest of your program.

The intent of this feature is to let you input values for **X** and **Y** 'graphically'. Cool, eh?

Running this program...

```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM:U2S2B
:Input
:Disp X,Y
:█
```

causes this. Move the cursor...



and press [ENTER] to see this...

```
NORMAL FLOAT AUTO REAL RADIAN MP
PrgrM U2S2B
1.5
1
Done
:█
```



Teacher Tip: When we get into graphics programming in Unit 5 this special feature will come in very handy!

Unit 2: Using Variables and Expressions

Skill Builder 1: Prompting for a variable

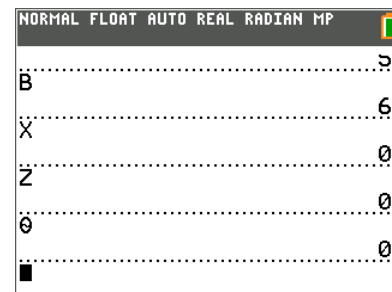
In the first lesson for Unit 2 you will learn about the **Prompt** statement to make your programs interactive, using variables to hold numeric values, evaluating and storing results of mathematical expressions, and using **Disp** and **Output** statements to show the results of stored computations.

Objectives:

- Use the TI Basic **Prompt** statement to assign a value to a variable.
- Know the difference between mathematical variables and computer variables.
- Perform calculations within **Disp** statements.
- Use **Output** statements to produce meaningful, readable results.

Real Variables

- The TI-84 Plus has 27 built-in variables that are used to store numeric values.
- The values can be *real* (decimal) numbers or *complex* numbers.
- The variable names are the letters **A** through **Z** and the letter **θ** ('theta').
- All variables contain a value. If a value is not assigned then the default value is 0 (zero).
- The values remain stored even when the calculator is turned off.
- If RAM is reset then all the values are set to 0.
- The HOME screen at the right shows some variables (on the left) and their *current* values (on the right). Yours may differ.



Teacher Tip: There are many types of variables in a TI-84 Plus. Among them are real and complex, lists, matrices, Y-vars (the graphing functions of all types) programs, apps, appvars, strings, pictures (PICs), and on the TI-84 Plus CE, background Images. This lesson deals only with the *real* variables (which can also store *complex* values). For the list of variable types look at the Mem Mgmt/Del section of the [MEM] menu.

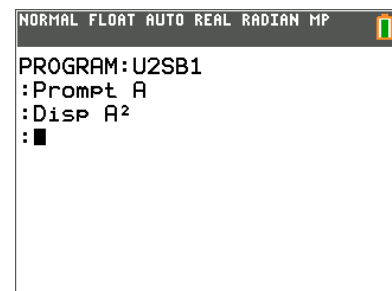
The Prompt Statement

- The **Prompt** statement is followed by one or more variable names that ask the user to enter a value for a variable.
- It is called '**Prompt**' because when you run the program, it displays the name of the variable and a question mark.



Programming with Prompt

1. Start a new program.
2. For first statement of the program use the **Prompt** statement found in the [PRGM] I/O menu.
3. After the **Prompt** command type the name of the variable you want your program to use. In this program we will use the letter **A**.
4. Use the **Disp** statement to display the square **A²**; type **A** then the x^2 key.
5. Quit the editor and run the program.
6. After the "A=?" prompt, type any number.



- The program displays the square of that number and ends.

```
NORMAL FLOAT AUTO REAL RDIAN MP
prgmU2SB1
A=?13
169
Done
```

Entering Multiple Values with Prompt

- Edit the program from above.
- Add ,**B** to the **Prompt** statement.
- Change the **Disp** statement so that it displays the sum **A+B**
- Run the program again.

Notice the two prompts? The **Prompt** statement asks for a value for each variable separately.

This is a very simple, efficient program requiring only two statements, but these two statements are doing a lot of work!

```
NORMAL FLOAT AUTO a+b RADIAN MP
PROGRAM:U2SB1
:Prompt A,B
:Disp A+B
:
```

```
NORMAL FLOAT AUTO a+b RADIAN MP
prgmU2SB1
A=?5
B=?6
11
Done
```

Using Output(Instead of Disp

Recall that you can improve the output of programs using **Output(** rather than **Disp** to show the original values entered *and* the results *properly labeled*. Just put the calculation right in the Output statement. You try it.

Example: **Output(5,7,A+B)** show the value of A+B on line 5 beginning in column 7.

- To the right are two screens of a running program, one showing the **Prompt** section and one showing the **Output** section. Can you do better?
- Remember to include **Pause** and **ClrHome** statements at the right moments in the program to keep the screen neat.

You cannot output two items with one **Output** statement. The message "SUM=" and the sum **A+B** must be output using separate statements. Screen position is important!

Note: you'll find the "=" ('equals' sign) on the Test menu (**2nd** **MATH**).

```
NORMAL FLOAT AUTO a+b RADIAN MP
prgmSDPQ
A=?13
B=?7
```

```
NORMAL FLOAT AUTO a+b RADIAN MP
A=13
B=7
SUM=20
```



Teacher Tip: Suggest trying other mathematical expressions in the **Disp** statement. We'll get into storing (assigning) values to variables later in this unit (see Unit 2, Skill Builder 3).

Remember that the TI-84 Plus (non-color) screen has 16 characters per line and 8 lines, so the choice of Output positions may vary. The TI-84 Plus C and CE (color) screens have 26 characters per line and 10 lines.

In this Application for Unit 2 you will write programs to evaluate some mathematical formulas.

Objectives:

- Use the TI Basic statements learned in Unit 2 to write a program that evaluates a formula.

The Pythagorean Theorem

In a right triangle with legs A and B and hypotenuse C,

$$A^2 + B^2 = C^2$$

Write a program that asks the user to enter the lengths of the legs then computes the length of the hypotenuse and nicely displays all three values.

Note: You first have to solve the formula above for C.

```
NORMAL FLOAT AUTO REAL Radian MP
PROGRAM:PYTHAG
:ClrHome
:Disp "THIS PROGRAM COMPUT
ES"
:Disp "THE HYPOTENUSE"
:Disp "ENTER THE LEGS..."
:Prompt A,B
:
:
:
```

Heron's Formula

Heron's Formula determines the area of any triangle using only the lengths of the three sides of the triangle, A, B, and C. It is usually stated in two parts:

$S = (A + B + C) / 2$ is the 'semi-perimeter' (half the perimeter) of the triangle

$A = \sqrt{S * (S - A)(S - B)(S - C)}$ is the area of the triangle

```
NORMAL FLOAT AUTO REAL Radian MP
PROGRAM:HERON
:ClrHome
:Disp "THIS PROGRAM COMPUT
ES"
:Disp "HERON'S FORMULA"
:Disp "ENTER THE SIDES..."
:
:Prompt A,B,C
:
:
:
```

Write a program that asks the user to enter the lengths of the three sides of a triangle and then computes the area and displays (Outputs) the sides and the area on a pretty screen.

Note: It's possible for the user to enter three numbers that cannot be the sides of any triangle. What will happen when the user enters invalid values?

Teacher Tip: Heron's formula will fail (NON-REAL answer) when the three values are impossible for a triangle (the Triangle Inequality). This can be accounted for by changing the complex MODE to a+bi.

The Quadratic Formula

If a quadratic equation is of the form $Ax^2 + Bx + C = 0$ then the roots of the equation are found by...

First, determining the discriminant:

$$D = B^2 - 4AC$$

And then the two roots are:

$$R1 = (-B + \sqrt{D}) / (2A)$$

$$R2 = (-B - \sqrt{D}) / (2A)$$

Write a program that asks the user to enter the three coefficients of the quadratic equation, A, B, and C and nicely displays the coefficients and the two roots of the equation.

```
NORMAL FLOAT AUTO REAL Radian MP
PROGRAM:QUAD
:ClrHome
:Disp "THIS PROGRAM COMPUT
ES THE"
:Disp "QUADRATIC FORMULA"
:Disp "ENTER THE COEFFICIE
NTS..."
:Prompt A,B,C
:
:
:
```

Note: You cannot use R1 and R2 as variables! Use something else.

What could possibly go wrong with this program?

Teacher Tip: Here are program listings for each assignment. The important steps are the formula calculations. The Output positions should be fine on any TI-84, but the TI-84 C/CE may use different values depending on the appearance desired.

The Pythagorean Theorem

Answer:

```
prgmPYTHAG
ClrHome
Disp "THIS PROGRAM COMPUTES"
Disp "THE HYPOTENUSE"
Disp "ENTER THE LEGS..."
Prompt A,B
 $\sqrt{(A^2+B^2)} \rightarrow C$ 

ClrHome
Output(3,5,"A= ")
Output(3,8,"A)
Output(4,5,"B= ")
Output(4,8,B)
Output(6,5," HYPOTENUSE = ")
Output(6,16,C)
Pause
ClrHome
```

Heron's Formula

Answer:

```
prgmHERON
ClrHome
Disp "THIS PROGRAM COMPUTES"
Disp "HERON'S FORMULA"
Disp "ENTER THE SIDES..."
Prompt A,B,C
 $(A+B+C)/2 \rightarrow S$ 
 $\sqrt{(S(S-A)(S-B)(S-C))} \rightarrow D$ 

ClrHome
Output(3,5,"A= ")
Output(3,8,A)
Output(4,5,"B= ")
Output(4,8,B)
Output(5,5,"C= ")
Output(5,8,C)
Output(7,5,"AREA= ")
Output(7,11,D)
Pause
ClrHome
```

The Quadratic Formula

Answer:

```
prgmQUAD
ClrHome
Disp "THIS PROGRAM COMPUTES"
Disp "THE"
Disp "QUADRATIC FORMULA"
Disp "ENTER THE COEFFICIENTS..."
Prompt A,B,C
 $B^2-4AC \rightarrow D$ 
 $(-B+\sqrt{D})/(2A) \rightarrow R$ 
 $(-B-\sqrt{D})/(2A) \rightarrow S$ 

ClrHome
Output(3,5,"A= ")
Output(3,8,A)
Output(4,5,"B= ")
Output(4,8,B)
Output(5,5,"C= ")
Output(5,8,C)
Output(7,5,"ROOT1= ")
Output(7,12,R)
Output(8,5,"ROOT2= ")
Output(8,12,S)
Pause
ClrHome
```