Unit 3: Conditional Statements | **Skill Builder 1: Request and If**

In this first lesson for Unit 3, you will learn about **Request**(ing) input from the user while the program is running, strings, and the most basic (primitive) of conditional statements, **If**.

Objectives:

- Use **Request** and **RequestStr** for input
- Investigate string variables and concatenation
- Write statements using **If** and conditions
- Use **DispAt** for better control of output

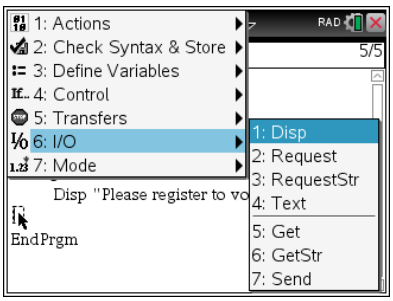
Teacher Tip: This lesson contains several unrelated but vital concepts: input, strings, concatenation, conditions, the simple **If** statement, and **DispAt**. The program example is not at all complex. It demonstrates all of these topics in a concise, one-screen application. The screens used here show the Program Editor app on a page by itself. If the Program Editor appears on a split screen you can ungroup the apps on the page by selecting **doc > Page Layout > Ungroup**, or use the shortcut **ctrl+6**. The apps will then be on separate pages. To regroup them onto a split page, use **doc > Page Layout > Group** or **ctrl+4**.

Input Overview

Up to now, we've only been able to provide values in a program or function by using arguments. In the TI-Nspire™ CX, there are two statements that allow you to enter values into a program *while it is running*. These are known as '*input'* commands:

Request "*message*", *variable* (*for numeric input*)

RequestStr "*message*", *variable* (*for string input*)



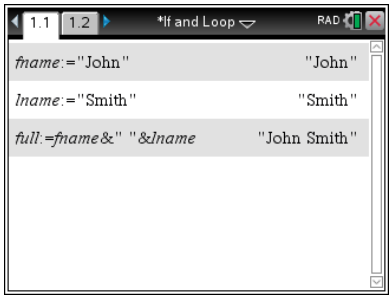
These statements are found in the **I/O** menu of the Program Editor.

Types of Variable Input

- A numeric variable can contain a real or complex number, a list, or even a matrix. It can be used in algebraic expressions, and its value is used during the computation of the expressions.
- A string variable can contain any text including letters, digits, and most punctuation marks. It cannot be used in algebraic expressions. Note that numbers are allowed in strings.
- The "message" is called the 'prompt'. It describes to the user what is supposed to be entered during the command's operation.
- The variable can be any alphabetic character, including most Greek characters or a word, such as ***hours***, that is not a reserved word. You will recognize a reserved word when the font style changes from Italic to normal as you type it.

String Concatenation

Two strings (variables or literals or a combination) can be combined into one longer string (concatenation) using the **'&**' operator. This operator is accessed by pressing **ctrl +** [📖].



Example on a Calculator app:

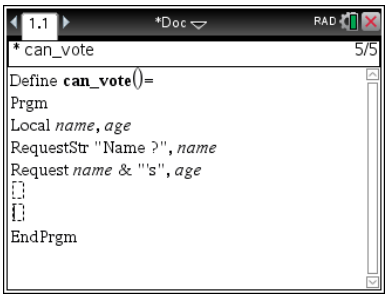
fname:="John"

lname:="Smith"

full:=*fname* & " " & *lname*

This causes the variable *full* to contain the string "John Smith"

Request and RequestStr



The program to the right contains no arguments and two **Local** variables *name* and *age*. There is no distinction between numeric and string variable names but there are two different statements for entering numeric and string data.

The **RequestStr** statement contains a prompt ("Name ?") that is displayed so that the user know what to enter.

The **Request** statement (for age) *concatenates* the name from the first statement with the text " 's age?" using the **&** symbol. A space is not needed in front of the 's.

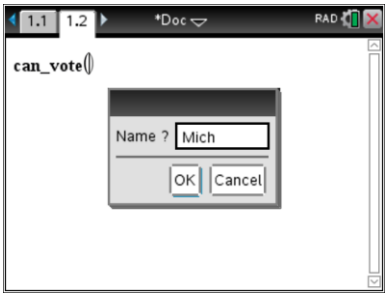
Some refer to string concatenation as string 'addition,' but it is not a mathematical operation.

Note: Any variable name can contain an underscore character (e.g., **can_vote**). The underscore character is accessed by pressing [?!▸].

RequestStr and **Request** each produce a dialog box containing the prompt and a field for entering your data.

To the right is the effect of the statement:

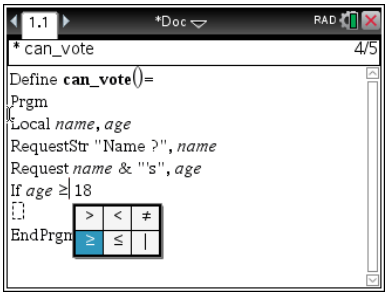
RequestStr "Name ?" , name



The user has entered 'Mich' so far.

Conditions

A *condition* is an expression that evaluates to *true* or *false. Conditions* use the relational operators. Select [ctrl] [=] to display the relational operators selection box. For the equals sign, simply press [=].



Conditions can also use the *logical operators*, **and**, **or**, **not**, and **xor.** These operators are found only in the Catalog. Also included are **nand, nor, implies (⇒), if and only if (⇔)**, and an **isPrime( )** function.

Examples of Conditions:

- **x > 0 and y > 0** (which is *true* when x is greater than 0 and y is greater than 0; note the spaces before and after the word 'and')
- **hours > 40**
- **time ≤ 0**
- **age ≥ 18**
- **$c^2 = a^2 + b^2$**
- **isPrime(n)** (a built-in function)

Teacher Tip: Beginners will need practice with forming and evaluating conditions. You can try out various conditions in the Calculator app. The logical operators such as **and**, **or**, and **not**, as with all commands in the Program Editor, can be typed rather than selected from a menu. Be sure there's a space before and after the operator. These keywords can be found in the Catalog.

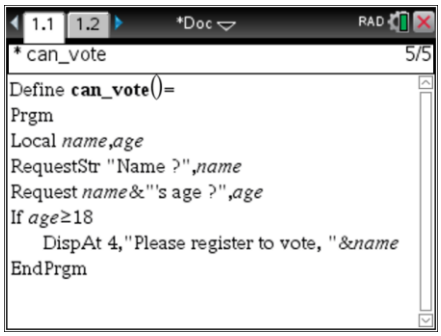
The 'Primitive' If Statement

The simplest form of all the **If** statements is

If *condition*

do only this statement

An example of this structure is found in the ***can_vote()*** program. When the value of the age variable is greater than or equal to 18, then the **DispAt** statement below it is executed; otherwise, it is skipped. This form of the **If** statement is useful for simple operations that do not require multiple statements or alternative actions (using **else**). **Then** is not used in this statement.



DispAt

The **DispAt** statement (introduced in OS version 4.5) gives programmers greater control of the output in the Calculator app. When running a program, there are 8 lines of output area with which to work. Use the structure:

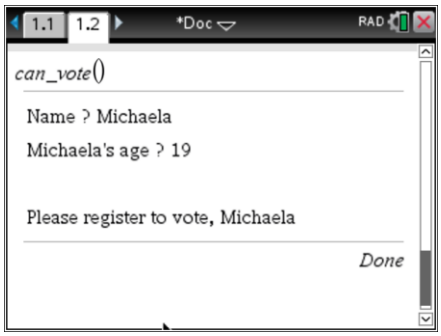
DispAt <*line number*> , <*expression to display*>

Note the comma after the line number:

DispAt 4, "Please register to vote, " & name

The result of using **DispAt** in this program is seen to the right. Concatenation is used to append the user's name to the message.

The two **Request** statements are echoed on lines 1 and 2 of the output area, and the "Please register…" message is displayed on line 4. Note the blank area, which is line 3.

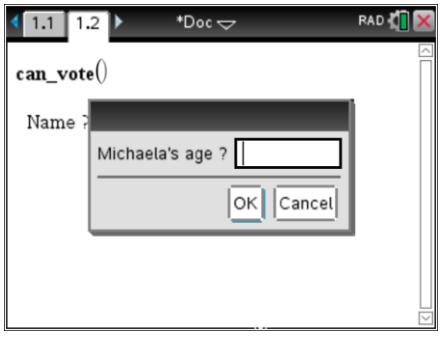


Indenting

Note that the **DispAt** statement is indented. Indenting statements is permitted on the TI-Nspire CX. Indenting helps to make the code more readable. The **DispAt** statement has to be on the line directly below the **If** statement in this program.

Running the Program

1. After entering the program, select **ctrl+R** to prepare to run the program, and then select **enter**.
2. The RequestStr statement displays a dialog box. Type a name, and select **enter** or click **OK** on the screen**.**
3. The Request statement displays a message consisting of the user's name and the word 'age' and expects a number as shown to the right. Type an age number, and select **enter** or click **OK**.



The results of the program are displayed as output in the Calculator app. If the age is greater than 18, then the text 'Please register…' is displayed. Otherwise, as seen to the right, it is not displayed.

Teacher Tip: If you don't want to display the echoing of the Request statements when the program is run, you can suppress that text in the Calculator app. Add **,0** to the end of any or all of the **Request** or **RequestStr** statements in the program:

Request "number?", n, 0

This causes the text of the input action to be hidden when the statement is finished. You can suppress some or all of the input texts.