Tools on the Edit Toolbar
Table 5-2.

Tool	Function	Keyboard Equivalent	Description
	List properties/ methods	CTRL+J	Displays a pop-up list box with the properties and methods for the object preceding the period.
	List constants	CTRL+SHIFT+J	Displays a pop-up list box with the valid constants.
	QuickInfo	CTRL+I	Gives the syntax for the procedure or method.
	Parameter info	CTRL+SHIFT+I	Provides the parameter list for the current function call (see Chapter 8).
	Complete word	CTRL+SPACEBAR	Completes the keyword or object when enough information is there (for example, msg would complete to MsgBox).
	Indent	TAB	Indents the selected text one tab stop. (Use the Editor page on the Tools|Options dialog box to change the number of spaces.)
	Outdent	SHIFT+TAB	Moves the selected text back one tab stop.
	Toggle breakpoint	F9 (Left-clicking in the left margin next to a line of code also toggles the breakpoint.)	Used for debugging (see Chapter 15).
	Comment block	None	See the following section "Comments" for more information on comments.

Tools on the Edit Toolbar (*continued*)
Table 5-2.

Tool	Function	Keyboard Equivalent	Description
	Uncomment block	None	See the following section, "Comments" for more information on comments.
	Toggle bookmark	None	The editor allows you to put bookmarks at specific places in your code. You can jump from bookmark to bookmark to more easily navigate between parts of your code.
	Next bookmark	None	Move to next bookmark.
	Previous bookmark	None	Move to previous bookmark.
	Clear all bookmarks	None	

Statements in Visual Basic

First off, let me remind you that a good way to think of a statement in a programming language is to think of it as a complete sentence—a complete thought. When you enter a statement in Visual Basic and have the Auto Syntax option that I just discussed turned on, the intelligent editor built into VB analyzes what you typed. This happens immediately after you press ENTER. VB checks to see that what you entered makes sense. It is amazing how Visual Basic can detect many typos at this stage, but *only* if you have Auto Syntax Check turned on in the Editor page of the Options dialog box. Sometimes VB can even fix the error: for example it will add a closing quote if you left it out. If a statement you entered can't be analyzed, though, a message box pops up and can often help you find out what caused the problem.

TIP: Remember the context-sensitive help feature of Visual Basic. If you need help on the syntax for functions, statements, properties, events, or methods in the Code window, move the cursor to (or type) the keyword or the property, event, or method name, and press F1. You can also select an event name in the Procedure box and press F1 for information about that event.

Note that the notation presented in the syntax statement may seem cryptic at first, but it is a good idea to get used to it. It is the notation used in both the QuickInfo feature, the manuals, and the online documentation. With this notation, anything in square brackets is optional. Notice that the parentheses are outside the square brackets, so they are required. The commas separate the optional elements (the parameters or arguments) in this function. If you skip one of the arguments, you still have to use a comma as a separator. (How else would Visual Basic know which argument belongs where? For example,

```
MyInput$ = InputBox("Example", , "A default string", 100, 200)
```

leaves the title bar as the default value (the project name) but starts the box out with the string "A default string" inside of it.

NOTE: Please see the section "Advanced Uses of Procedures and Functions" in Chapter 9 for a way around having to use a lot of empty commas.

Regardless of which type of Input Box you use, the user can type whatever he or she wants in the text box. If he or she presses ENTER or clicks on the OK button, then VB takes whatever is in the text box and sets it as the value of the string variable. Pressing ESC or clicking the Cancel button causes Visual Basic to assign the null string to the variable.f.