INPUT BOXES AND MESSAGE BOXES

INPUTBOX

Input boxes provide a simply way to gather input from the user without placing a text box on the form. An input box displays a message to the user and provides a text box for the user to enter input.

You can display input boxes through the use of the **InputBox** function, which takes the following format:

```
InputBox (Prompt, Title, Default, xPos, yPos)
```

PROMPT: The first argument is a string that is displayed to the user in the input box. Typically, the string value is used to prompt the user to enter a value.

TITLE: This is an optional argument that takes a string value which is displayed in the title bar of the input box. If you do not provide a value, the name of the project appears in the title bar.

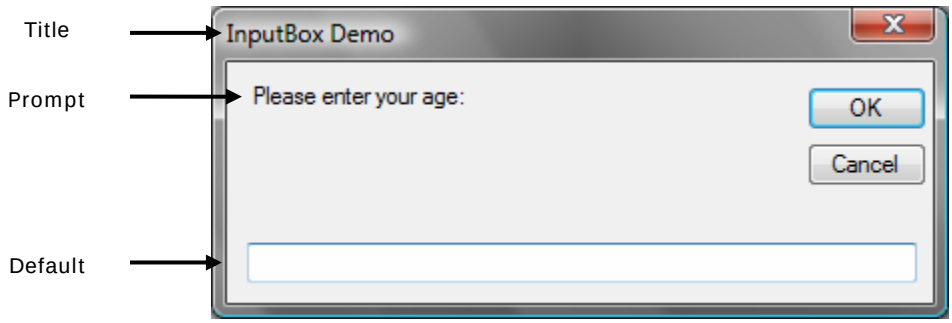
DEFAULT: This is also an optional argument that takes a string value which is initially displayed in the input box's text box. If you do not provide a value, the text box is left empty.

XPOS: This is an optional argument that takes an integer value which specifies the x-position (in pixels) of the input box. If you do not provide a value, the input box is centred horizontally on the screen.

YPOS: This is an optional argument that takes an integer value which specifies the y-position (in pixels) of the input box. If you do not provide a value, the input box is centred vertically on the screen.

Here is an example of an input box that prompts the user for his/her age:

```
InputBox("Please enter your age:", "InputBox Demo")
```



If the user clicks the **OK** button or presses the ENTER key, the function returns the string value from the input box's text box. If the user clicks the **Cancel** button, the function returns an empty string (i.e. ""). To retrieve the value returned by the input box, you will need to declare a variable and make it equal to the input box, as follows:

```
Dim age As String
age = InputBox("Please enter your age:", "InputBox Demo")
```

If you want to store the value that the user enters into an Integer variable, you would simply need to convert the value returned by the input box using the **CInt()** or **Val()** functions as follows:

```
Dim age As Integer
age = Val(InputBox("Please enter your age:", "InputBox Demo"))
```

- OR -

```
Dim age As Integer
age = CInt(InputBox("Please enter your age:", "InputBox Demo"))
```

THE MESSAGE BOX

A message box is a dialog box that displays a message to the user in the form of a pop-up window. To display a message box you will need to use the **MsgBox** function, which has the following three general formats:

1 `MsgBox(`*`Message`*`)`

This format takes one (1) parameter, which is a String that will be displayed in the message box.

EXAMPLE:

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As
      System.EventArgs) Handles Me.Load

      MsgBox("Hello world!")

End Sub
```

2 `MsgBox(`*`Message, MsgBoxStyle`*`)`

This format takes two (2) parameters: a String that will be displayed in the message box, the number and type of buttons to display and the icon style to use.

EXAMPLE:

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As
      System.EventArgs) Handles Me.Load

      MsgBox("Hello world!", MsgBoxStyle.Question)

End Sub
```

The following table outlines just some of the buttons that are available:

VALUE	DESCRIPTION
`MsgBoxStyle.AbortRetryIgnore`	Displays **Abort**, **Retry**, and **Ignore** buttons.
`MsgBoxStyle.OK`	Displays an **OK** button.
`MsgBoxStyle.OKCancel`	Displays **OK** and **Cancel** buttons.
`MsgBoxStyle.RetryCancel`	Displays **Retry** and **Cancel** buttons.
`MsgBoxStyle.YesNo`	Displays **Yes** and **No** buttons.
`MsgBoxStyle.YesNoCancel`	Displays **Yes**, **No** and **Cancel** buttons.

The following table outlines some of the icons that are available:

VALUE	ICON
`MsgBoxStyle.Information`	
`MsgBoxStyle.Critical`	
`MsgBoxStyle.Exclamation`	
`MsgBoxStyle.Question`	

3 `MsgBox(`*`Message, MsgBoxStyle, Caption`*`)`

This format takes three (3) parameters: a String that will be displayed in the message box, a value that specifies which buttons and icon to display in the message box, and a second string that will be displayed in the message box's title bar.

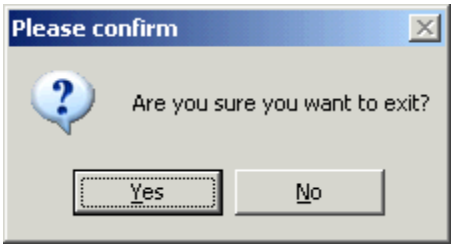
EXAMPLE:

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As
      System.EventArgs) Handles Me.Load

      MsgBox("Are you sure you want to exit?", MsgBoxStyle.YesNo,
            "Please confirm")

End Sub
```

To specify which buttons you want to appear in a message dialog and which icon you want to display, you would use an **Or** statement as follows:

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As
      System.EventArgs) Handles Me.Load

      MsgBox("Are you sure you want to exit?", MsgBoxStyle.YesNo
            Or MsgBoxStyle.Question, "Please confirm")

End Sub
```

The above code would produce a message dialog box with YES and NO buttons and the QUESTION icon:



DETERMINING WHICH BUTTON THE USER CLICKS

When the user clicks a button on a message box, the **MsgBox** function returns an integer value that indicates which button the user clicks.

The following table outlines the return values that can be returned by the **MsgBox** function:

VB CONSTANT	VALUE	DESCRIPTION
MsgBoxResult.OK	1	Indicates the user clicked the **OK** button.
MsgBoxResult.Cancel	2	Indicates the user clicked the **Cancel** button.
MsgBoxResult.Abort	3	Indicates the user clicked the **Abort** button.
MsgBoxResult.Retry	4	Indicates the user clicked the **Retry** button.
MsgBoxResult.Ignore	5	Indicates the user clicked the **Ignore** button.
MsgBoxResult.Yes	6	Indicates the user clicked the **Yes** button.
MsgBoxResult.No	7	Indicates the user clicked the **No** button.

If you want the program to execute a specific set of instructions based on the message box button that the user clicks, you will need to declare a variable as a **MsgBoxResult** to store the user's response to the question you are displaying in the **MsgBox**, and use an **If** statement to define the instructions that you want the program to execute. In the following example, the program will end if the user clicks the YES option, otherwise the program will simply proceed to the main form:

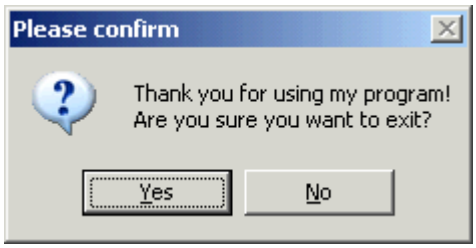
```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As
    System.EventArgs) Handles Me.Load

        'Declare local variable
        Dim choice As MsgBoxResult

        'Ask user if he/she wishes to exit the program
        choice = MsgBox("Are you sure you want to exit?",
             MsgBoxStyle.YesNo Or MsgBoxStyle.Question, "Please
             confirm")

        'User chooses to exit
        If choice = MsgBoxResult.Yes Then
             Application.Exit()
        End If

End Sub
```

DISPLAYING MULTIPLE LINES IN A MESSAGE BOX

You can display multiple lines of information in a message box by using the constant **ControlChars.CrLf**. You would just need to concatenate the constant with the string you wish to display whenever you want to begin a new line.

In the following example, the line “Thank you for using my program?” is followed by the statement “Are you sure you want to exit?” on the next line:

```
MsgBox("Thank you for using my program" & ControlChars.CrLf &
      "Are you sure you want to exit?", MsgBoxStyle.YesNo Or
      MsgBoxStyle.Question, "Please confirm")
```



When formatting output in a message box or input box, you may find it necessary to use some of the other Print and Display constants available in Visual Basic. The following are a list of some of the more common Print and Display constants:

MEMBER	DESCRIPTION
ControlChars.CrLf	Carriage-return/linefeed character combination
ControlChars.Cr	Carriage-return character
ControlChars.Lf	Linefeed character
ControlChars.NewLine	Newline character
ControlChars.Tab	Tab character
ControlChars.Back	Backspace character