

# Functions



[ [What are functions and how do I use them?](#gjdgxs) ]

[ [File](#30j0zll) | [Number](#1fob9te) | [String](#3znysh7) | [Time/Date](#2et92p0) | [Variable Conversion](#tyjcwt) ]

## ***What are functions and how do I use them?***

Functions are similar to procedures but always return a value of some sort. If a procedure does not return a value then it can just be called, but with functions the value passed back must be handled some how.

### Example 1: (No Parameters)

A useful function to start with is **Time**. This role of this function is to return the current system clock time; it does not take any parameters. To try this function out create a new label control on a suitable form. Next place a Timer control on the form as well and set its **Interval** to **1000**. Double click on the Timer control and enter the line **Label1.Caption = Time**. When you run the program you should see the label control showing the time updated every second.

### Example 2: (A Single Parameter)

The above Time function is simple to use because it does not require any parameters. However, most functions require parameters to supply key information to the module in order for it to work out the return value. The Time function does not require a parameter because all it needs to do is interogate the system clock. The function **UCase** is a simple function requiring only one parameter to be passed. To pass variables and constants as parameters they must be enclosed in brackets after the function name. Multiple parameters must be separated by commas. Create a text control on a form and then in the **KeyPress** event write: **If KeyAscii = 13 then Text1.Text = UCase(Text1.Text)**. Run the program, enter some text and press RETURN. The result should be that the text in the text box turns to upper-case.

### Example 3: (Multiple Parameters)

In example 2 the function returned different text depending on the string sent to it in the parameter. Other functions require additional parameters to tell it how to handle the other parameters. For example, the function **Mid$** takes one parameter to pass in the text to be used, one parameter to set the starting point (measured in characters), and then a third and optional parameter to specify how many characters to return. We can use this Mid$ function to return the word 'University' out of the string 'Napier University, Edinburgh'. In the **Form Activate** event enter the following code: **Label1.Caption = Mid$("Napier University, Edinburgh", 8, 10)**. When run all that will be displayed in Label1 will be the word 'University'. Try leaving out the last parameter which is optional and see what happens.

### Example 4: (Using Functions as Parameters)

Can the returning value of one function be used as the input parameter for another? The answer is yes, the example below does just that.

Suppose we need to build an input text box on a form to enter a person's name but later in the program we do not want to use the full name just the surname. An easy (or perhaps lazy) solution would be to use two separate text box controls, one for the first name and the other for the surname. However, this uses up more screen real estate and could be slower to enter. Where possible it is better to get the computer to do the hard work instead of the user. To do this we are going to create only one text input box and use a label control to output the result of the function. In the **KeyPress** event of the text box place the following code:

**Private Sub Text1\_KeyPress (KeyAscii As Integer)  
 Dim strFullName, strSurname As String  
   
 If KeyAscii = 13 Then  
 strFullName = Text1.Text  
 strSurname = Mid$(strFullName, InStr(1, strFullName, " ") + 1)  
 Label1.Caption = "Surname = " + UCase(strSurname)  
 End If  
  
End Sub**

This example uses three of Visual Basic's built in functions: Mid$, InStr, and UCase. The function **Mid$** in this case is using two parameters (the option third parameter is not being used). However, the second parameter which sets the starting character has been replaced by the function **InStr** which is used to return the position of located characters. In this case InStr is looking for the first occurrance of the character " " (space). We will use this to determine where the first name ends and the surname begins. One is added to this returning value because we do not want to return the space, we want to start one letter after the space. In Visual Basic an functions that return values that will be used as input parameters to other functions are evaluated first. The example above uses two local varaibles to try and make the code look less complicated and make it easier to read. However, it could be rewritten so that the the result from the Mid$ function is used as the input to UCase:

**Private Sub Text1\_KeyPress (KeyAscii As Integer)  
   
 If KeyAscii = 13 Then Label1.Caption = UCase(Mid$(Text1.Text, InStr(1, Text1.Text, " ") + 1))  
  
End Sub**

This shortened example will do exactly the same as the previous example except the label will not show 'Surname = ' before displaying the name. It should be noted that although both examples perform the same job the first is much more readable and thus less prone to bugs. When too many functions are used on a single line it is difficult matching closing brackets. If you find you are using too many functions on a single line split them up over several lines with the help of some temporary local varaibles.

### Example 5: (Making your own)

If Visual Basic does not have a suitable built in function then it is possible to build your own. To create a user defined function open up a code window and move the cursor down to the bottom past the last **End Sub** statement. Here, enter **Function *function\_name* ()**. Replace the word 'function\_name' with the exact name you wish to call the function. After pressing RETURN the code window should clear and display the new function on its own with the added line **End Function** at the end.

In this example we will write our own function to return the current time in New York. We will create a new form with a label control and a command button control on it (see below).



Using the method outlined above create a new function with the code shown below:

**Function NY\_time ()  
 Dim strTempTime As String  
 Dim intNY\_hour As Integer  
  
 strTempTime = Time$  
 intNY\_hour = Val(Left$(strTempTime, 2)) - 5  
  
 Rem Correct for passing back over a day  
 If intNY\_hour < 0 Then intNY\_hour = intNY\_hour + 24  
  
 intNY\_time = Str(intNY\_hour) + Right$(strTempTime, 6)  
End Function**

Next double click on the command button and enter the following code:

**Private Sub cmdGetTime\_Click ()  
 lblTime.Caption = "Time in New York is " + intNY\_time()  
  
End Sub**

This function is quite simple it just reads the system clock and takes five hours off the hour. A check to see if the hour is less than zero is require in case the time in New York is the evening before.

**Exercise:** Try expanding this function to make it more generalisable. Create a parameter to pass in the time zone difference (in hours) from Greenwich.

Below are listed some of the more important functions in Visual Basic. To find out exactly how many parameters each function requires and what type of data the parameter should be use the online help within Visual Basic.

## ***File***

* **FileAttr** - Returns file mode or operating system file information about an open file.
* **FileDateTime** - Returns a String that indicates the date and time a specified file was created or last modified.
* **FileLen** - Returns a Long integer that indicates the length of a file in bytes.
* **FreeFile** - Returns the next valid unused file number.
* **GetAttr** - Returns an integer that indicates a file, directory, or volume label's attributes.
* **Loc** - Returns the current position within an open file.
* **LOF** - Returns the size of an open file in bytes.
* **Seek** - Returns the current file position.

## ***Number***

* **Abs** - Returns the absolute value of a number.
* **Atn** - Returns the arctangent of a number.
* **Cos** - Returns the cosine of an angle (angle in radians).
* **Sin** - Returns the sine of an angle (angle in radians).

## ***String***

* **InStr** - Returns the position of the first occurrence of one string within another string.
* **LCase, LCase$** - Returns a string in which all letters of an argument have been converted to lowercase.
* **Left, Left$** - Returns the leftmost n characters of a string argument.
* **Len** - Returns the number of characters in a string expression or the number of bytes required to store a variable.
* **LTrim, LTrim$** - Returns a copy of a string with leading spaces removed.
* **Mid, Mid$** - Returns a string that is part of some other string.
* **Right, Right$** - Returns the rightmost n characters of a string argument.
* **RTrim, RTrim$** - Returns a copy of a string with trailing (rightmost) spaces removed.
* **Space, Space$** - Returns a string consisting of a specified number of spaces.
* **Trim, Trim$** - Returns a copy of a string with both leading and trailing spaces removed.
* **UCase, UCase$** - Returns a string with all letters of the argument converted to uppercase.

## ***Time/Date***

* **Date, Date$** - Returns the current date.
* **IsDate** - Returns a value indicating whether of not a Variant argument can be converted to a date.
* **Minute** - Returns an integer between 0 and 59, inclusive, that represents the minute of the hour corresponding to the time provided as an argument.
* **Month** - Returns an integer between 1 and 12, inclusive, that represents the month of the year for a date argument.
* **Now** - Returns a date that represents the current date and time according to the setting of the computer's system date and time.
* **Time, Time$** - Returns the current system time.
* **Timer** - Returns the number of seconds that have elapsed since 12:00 a.m. (midnight).
* **Weekday** - Returns an integer between 1 (Sunday) and 7 (Saturday) that represents the day of the week for a date argument.
* **Year** - Returns an integer between 100 and 9999, inclusive, that represents the year of a date argument.

## ***Variable Conversion***

* **CCur** - Explicitly converts expressions to the Currency data type.
* **CDbl** - Explicitly converts expressions to the Double data type.
* **CInt** - Explicitly converts expressions to the Integer data type.
* **CLng** - Explicitly converts expressions to the Long data type.
* **CSnd** - Explicitly converts expressions to the Single data type.
* **CStr** - Explicitly converts expressions to the String data type.
* **CVar** - Explicitly converts expressions to the Variant data type.
* **CVDate** - Converts an expression to a Variant of VarType 7 (Date).

 [Menu](http://docs.google.com/dvbhome.html)