Built in Functions in VBA

Objectives: At the end of this lesson you shall be able to
- **describe the math functions in VBA**
- **describe the logical functions in VBA**
- **describe the date/time functions in VBA**
- **describe the conversion functions in VBA.**

Introduction

VBA has a rich collection of built in functions that perform a variety of tasks and calculations for you. There are functions to convert data types, perform calculations on dates, perform simple to complex mathematics, make financial calculations, manage text strings, format values, and retrieve data from tables, among others. Using the VBA Built in Functions will help coding much easier for the user. We have already used many built in functions in our earlier lessons like the msgbox function and many string manipulation functions just to name a few.

MS Excel: VBA Functions (VBA Formulae) - Category wise

The commonly used VBA functions in Excel, sorted by Category are shown here.

String Functions: The string functions were already discussed in the related theory for Ex. 2.2.09

Math Functions

Table 1 : Lists some of the common Built in Functions in the Mathematical category.

Table 1

Function Name	Description
Abs	Returns the absolute value of a number.
Cos	Returns the cosine of the specified angle.
Cosh	Returns the hyperbolic cosine of the specified angle.
Exp	Returns e (the base of natural logarithms) raised to the specified power.
Fix	Returns the integer portion of a number.
Format	Takes a numeric expression and returns it as a formatted string.
Int	Returns the integer portion of a number.
Log	Returns the natural (base e) logarithm of a specified number or the logarithm of a specified number in a specified base.
Rnd	Generates a random number (integer value)
Round	Returns a Decimal or Double value rounded to the nearest integral value or to a specified number of fractional digits.
Sign	Returns an Integer value indicating the sign of a number.
Sin	Returns the sine of the specified angle.
Sqr	Returns the square root of a specified number.
Tan	Returns the tangent of the specified angle.
Val	Accepts a string as input and returns the numbers found in that string.

1 Dim a, b as integer

```
a=81

debug.print sqr(a)
```

This will display the square root of 81 ie. 9

Logical Functions

Table 2. lists some of the common Built in Functions in the Logical category.

Table 2

Function Name	Description
ISDATE	Returns TRUE if the expression is a valid date. Otherwise, it returns FALSE.
ISERROR	Checks for error values.
ISNULL	Returns TRUE if the expression is a null value. Otherwise, it returns FALSE.
ISNUMERIC	Returns TRUE if the expression is a valid number. Otherwise, it returns FALSE.

Examples:

1 Sub Button1_Click()

```
N = TextBox1.Text
If IsNumeric(N) = True Then
MsgBox "correct"
Else
MsgBox "Insert only numbers"
End If
```

This checks if the data entered in the textbox is a number or not.

2 Sub Button1_Click()

```
N = TextBox1.Text
If IsDate(N) = True Then
MsgBox "correct"
Else
MsgBox "Insert only dates"
End If
End Sub
```

This checks if the data entered in the textbox is a valid date or not.

Date / Time Functions

Table 3. lists some of the common Built in Functions in the Date / Time category.

Table 3

Function	Return Value
DATE	Returns the current system date.
DATEADD	Returns a date after which a certain time/date interval has been added.
DATEDIFF	Returns the difference between two date values, based on the interval specified.
DATEPART	Returns a specified part of a given date.
DATESERIAL	Returns a date given a year, month, and day value.
DATEVALUE	Returns the serial number of a date.
DAY	Returns the day of the month (a number from 1 to 31) given a date value.
FORMAT Dates	Takes a date expression and returns it as a formatted string.
HOUR	Returns the hour of a time value (from 0 to 23).
MINUTE	Returns the minute of a time value (from 0 to 59).
MONTH	Returns the month (a number from 1 to 12) given a date value.
MONTHNAME	Returns a string representing the month given a number from 1 to 12.
NOW	Returns the current system date and time.
TIMESERIAL	Returns a time given an hour, minute, and second value.
TIMEVALUE	Returns the serial number of a time.
WEEKDAY	Returns a number representing the day of the week, given a date value.
WEEKDAYNAME	Returns a string representing the day of the week given a number from 1 to 7.
YEAR	Returns the year portion of the date argument.

Examples

1 DateDiff() Function

Syntax for the DateDiff function is :

DateDiff (interval, date1, date2, [firstdayofweek], [firstweekofyear])

Parameters or Arguments

Interval is the interval of time to use to calculate the difference between date1 and date2. Below is a list of valid interval values as in Table 4

Table 4

Interval	Explanation
yyyy	Year
q	Quarter
m	Month
y	Day of year
d	Day
w	Weekday
ww	Week
h	Hour
n	Minute
s	Second

Date1 and Date2 are the two dates to calculate the difference between.

first day of week is optional. It is a constant that specifies the first day of the week. If this parameter is omitted, Excel assumes that Sunday is the first day of the week.

first week of year is optional. It is a constant that specifies the first week of the year. If this parameter is omitted, Excel assumes that the week containing Jan 1st is the first week of the year.

```
Sub test()

Debug.PrintDateDiff("yyyy", "1/12/1999", "31/1/2000")

Debug.PrintDateDiff("q", "1/12/1999", "31/1/2000")

Debug.PrintDateDiff("m", "1/12/1999", "31/1/2000")

End Sub
```

The result will be

```
1

4

12
```

2 Format Date

Syntax

The syntax for the Microsoft Excel FORMAT function is:

Format (expression, [format, [firstdayofweek, [firstweekofyear]]])

Parameters or Arguments

Expression is the value to format.

Format is optional. It is the format to apply to the expression. You can either define your own format or use one of the named formats that Excel has predefined such as shown in Table 5.

Table 5

Format	Explanation
General Date	Displays date based on your system settings
Long Date	Displays date based on your system's long date setting
Medium Date	Displays date based on your system's medium date setting
Short Date	Displays date based on your system's short date setting
Long Time	Displays time based on your system's long time setting
Medium Time	Displays time based on your system's medium time setting
Short Time	Displays time based on your system's short time setting

First day of week is optional. It is a value that specifies the first day of the week. If this parameter is omitted, the FORMAT function assumes that Sunday is the first day of the week. This parameter can be one of the following values as shown in Table 6.

First week of year is optional. It is a value that specifies the first week of the year. If this parameter is omitted, the FORMAT function assumes that the week that contains January 1 is the first week of the year. This parameter can be one of the following values as shown in Table 7.

```
Sub test()

Debug.Print Format(#1/1/1990#, "Short Date")

Debug.Print Format(#1/1/1990#, "Long Date")

Debug.Print Format(#1/1/1990#, "yyyy/mm/dd")

End Sub
```

Table 6

Constant	Value	Explanation
vbUseSystem	0	Uses the NLS API setting
VbSunday	1	Sunday (default, if parameter is omitted)
vbMonday	2	Monday
vbTuesday	3	Tuesday
vbWednesday	4	Wednesday
vbThursday	5	Thursday
vbFriday	6	Friday
vbSaturday	7	Saturday

The result will be

1/1/1990

Table 7

Constant	Value	Explanation
vbUseSystem	0	Uses the NLS API setting
vbFirstJan1	1	The week that contains January 1
vbFirstFourDays	2	The first week that has at least 4 days in the year
vbFirstFullWeek	3	The first full week of the year

Monday, January 01, 1990

1990/01/01

Data Type Conversion Functions

Table 8. below lists some of the common Built in Functions in the Data Type Conversion category.

Table 8

Function	Return Type	Range for expression argument
CBool	Boolean	Any valid string or numeric expression.
CByte	Byte	0 to 255.
CCur	Currency	-922,337,203,685,477.5808 to 922,337,203,685,477.5807.
CDate	Date	Any valid date expression.
CDbl	Double	-1.79769313486231E308 to -4.94065645841247E-324 for negative values; 4.94065645841247E- 324 to 1.79769313486232E308 for positive values.
CDec	Decimal	+/-79,228,162,514,264,337,593,543,950,335 for zero-scaled numbers, that is, numbers with no decimal places. For numbers with 28 decimal places, the range is +/-7.9228162514264337593543950335. The smallest possible non-zero number is 0.0000000000000000000000000001.
CInt	Integer	-32,768 to 32,767; fractions are rounded.
CLng	Long	-2,147,483,648 to 2,147,483,647; fractions are rounded.
CSng	Single	-3.402823E38 to -1.401298E-45 for negative values; 1.401298E-45 to 3.402823E38 for positive values.
CStr	String	Returns for CStr depend on the expression argument.
CVar	Variant	Same range as Double for numerics. Same range as String for non-numerics.

Example

CDate function

```
Sub test()
Dim lNum As Long
Dim a As String
a = 12345
Debug.PrintCDate(a)
b = "January 1, 1990"
Debug.PrintCDate(b)
c = "1:23:45 PM"
Debug.PrintCDate(c)
End Sub
```

This will display

```
10/18/1933
1/1/1990
1:23:45 PM
```

User defined functions in VBA

Objectives: At the end of this lesson you shall be able to
- **create user defined functions**
- **describe passing values to functions byval and byref**
- **describe using arrays with functions**
- **describe the scope of variables**
- **describe the access specifiers public and private.**

Introduction

In Excel Visual Basic too, like in most programming languages, a set of commands to perform a specific task is placed into a procedure, which can be a function or a subroutine. The main difference between a VBA function and a VBA subroutine is that a function (generally) returns a result, whereas a subroutine does not.

Therefore, if you wish to perform a task that returns a result (ex. summing of a group of numbers), you will generally use a function, but if you just need a set of actions to be carried out (ex. formatting a set of cells), you might choose to use a subroutine.

User Defined Functions

One of the most power features of Excel VBA is that you can create your own functions or UDFs. A UDF (User Defined Function) is simply a function that you create yourself with VBA for your own defined tasks. UDFs are often called "Custom Functions". A UDF can remain in a code module attached to a workbook, in which case it will always be available when that workbook is open. Alternatively you can create your own add-in containing one or more functions that you can install into Excel. Here the user-defined functions can be entered into any cell or on the formula bar of the spreadsheet just like entering the built-in formulas of the MS Excel spreadsheet.

Custom functions, like macros, use the Visual Basic for Applications (VBA) programming language. They differ from macros in two significant ways. First, they use function procedures instead of sub procedures. They start with a Function statement instead of a Sub statement and end with End Function instead of End Sub. Second, they perform calculations instead of taking actions. Certain kinds of statements (such as statements that select and format ranges) are generally excluded from custom functions.

A simple function may look like this:

```
Function area()

Dim l, b

l = 10

b = 20

Debug.Print "area Is " & l * b

End Function
```

When executed from the immediate window this function displays the area.

Alternately this function can be called by another subroutine, for ex.

```
Sub test_fn()

Call area

End Sub
```

Returning a value from the procedures

In the example given below, the area() function calculates l*b.

The subroutine that calls this function is returned this value.

```
Sub test_fn()

Debug.Print "The function has returned the value " & area

End Sub

Function area()

Dim l, b, A

l = 10

b = 20

area = l * b

End Function
```

The result will be:The function has returned the value 200

Passing Arguments to functions

We can pass the arguments in two different ways: