Function findArea()

Dim Length As Double

Dim Width As Double

Length = InputBox("Enter Length ", "Enter a Number")

Width = InputBox("Enter Width", "Enter a Number")

findArea = Length \* Width

End Function

Variable is a named memory location used to hold a value that can be changed during the script execution.

Following are the basic rules for naming a variable.

You must use a letter as the first character.

You can't use a space, period (.), exclamation mark (!), or the characters @, &, $, # in the name.

Name can't exceed 255 characters in length.

You cannot use Visual Basic reserved keywords as variable name.

Data Types

There are many VBA data types, which can be divided into two main categories, namely numeric and non-numeric data types.

Numeric Data Types

Following table displays the numeric data types and the allowed range of values.

Type Range of Values

Byte 0 to 255

Integer -32,768 to 32,767

Long -2,147,483,648 to 2,147,483,648

Single

-3.402823E+38 to -1.401298E-45 for negative values

1.401298E-45 to 3.402823E+38 for positive values.

Double

-1.79769313486232e+308 to -4.94065645841247E-324 for negative values

4.94065645841247E-324 to 1.79769313486232e+308 for positive values.

Currency -922,337,203,685,477.5808 to 922,337,203,685,477.5807

Decimal

+/- 79,228,162,514,264,337,593,543,950,335 if no decimal is use

+/- 7.9228162514264337593543950335 (28 decimal places).

Non-Numeric Data Types

Following table displays the non-numeric data types and the allowed range of values.

Type Range of Values

String (fixed length) 1 to 65,400 characters

String (variable length) 0 to 2 billion characters

Date January 1, 100 to December 31, 9999

Boolean True or False

Object Any embedded object

Variant (numeric) Any value as large as double

Variant (text) Same as variable-length string

Private Sub say\_helloworld\_Click()

Dim password As String

password = "Admin#1"

Dim num As Integer

num = 1234

Dim BirthDay As Date

BirthDay = DateValue("30 / 10 / 2020")

MsgBox "Passowrd is " & password & Chr(10) & "Value of num is " & num & Chr(10) & "Value of Birthday is " & BirthDay

End Sub

constant

Constant is a named memory location used to hold a value that CANNOT be changed during the script execution.

If a user tries to change a Constant value, the script execution ends up with an error. Constants are declared the same way

the variables are declared.

Private Sub Constant\_demo\_Click()

Const MyInteger As Integer = 42

Const myDate As Date = #2/2/2020#

Const myDay As String = "Sunday"

MsgBox "Integer is " & MyInteger & Chr(10) & "myDate is "

& myDate & Chr(10) & "myDay is " & myDay

End Sub

operators:-

An Operator can be defined using a simple expression - 4 + 5 is equal to 9. Here, 4 and 5 are called operands and

+ is called operator. VBA supports following types of operators −

Arithmetic Operators

Comparison Operators

Logical (or Relational) Operators

Concatenation Operators

Operator Description Example

+ Adds the two operands A + B will give 15

- Subtracts the second operand from the first A - B will give -5

\* Multiplies both the operands A \* B will give 50

/ Divides the numerator by the denominator B / A will give 2

% Modulus operator and the remainder after an

integer division B % A will give 0

^ Exponentiation operator B ^ A will give 100000

The Comparison Operators

There are following comparison operators supported by VBA.

Assume variable A holds 10 and variable B holds 20, then −

Show Examples

Operator Description Example

= Checks if the value of the two operands are equal or not. If yes, then the condition is true. (A = B) is F

<> Checks if the value of the two operands are equal or not.

If the values are not equal, then the condition is true. (A <> B) is True.

> Checks if the value of the left operand is greater than the value of the right operand. If yes, then the condition is true. (A > B) is False.

< Checks if the value of the left operand is less than the value of the right operand. If yes, then the condition is true. (A < B) is True.

>= Checks if the value of the left operand is greater than or equal to the value of the right operand. If yes, then the condition is true. (A >= B) is False.

<= Checks if the value of the left operand is less than or equal to the value of the right operand. If yes, then the condition is true. (A <= B) is True.

The Logical Operators

Following logical operators are supported by VBA.

Assume variable A holds 10 and variable B holds 0, then –

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| AND | Called Logical AND operator. If both the conditions are True, then the Expression is true. | a<>0 AND b<>0 is False. |
| OR | Called Logical OR Operator. If any of the two conditions are True, then the condition is true. | a<>0 OR b<>0 is true. |
| NOT | Called Logical NOT Operator. Used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make false. | NOT(a<>0 OR b<>0) is false. |
| XOR | Called Logical Exclusion. It is the combination of NOT and OR Operator. If one, and only one, of the expressions evaluates to be True, the result is True. | (a<>0 XOR b<>0) is true. |

The Concatenation Operators

Following Concatenation operators are supported by VBA.

Assume variable A holds 5 and variable B holds 10 then −

[Show Examples](https://www.tutorialspoint.com/vba/vba_concatenation_operators.htm)

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Adds two Values as Variable. Values are Numeric | A + B will give 15 |
| & | Concatenates two Values | A & B will give 510 |

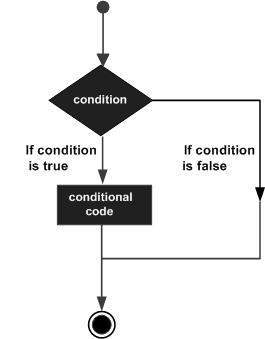
Assume variable A = "Microsoft" and variable B = "VBScript", then −

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Concatenates two Values | A + B will give MicrosoftVBScript |
| & | Concatenates two Values | A & B will give MicrosoftVBScript |

**VBA - Decisions**

Decision making allows the programmers to control the execution flow of a script or one of its sections. The execution is governed by one or more conditional statements.

Following is the general form of a typical decision making structure found in most of the programming languages.



VBA provides the following types of decision making statements. Click the following links to check their details.

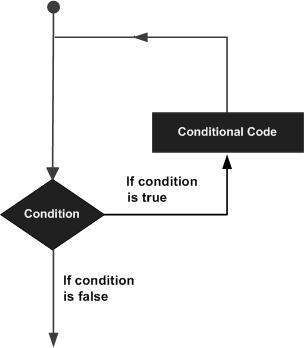
|  |  |
| --- | --- |
| **Sr.No.** | **Statement & Description** |
| 1 | [**if statement**](https://www.tutorialspoint.com/vba/vba_if_statement.htm)  An **if** statement consists of a Boolean expression followed by one or more statements. |
| 2 | [**if..else statement**](https://www.tutorialspoint.com/vba/vba_if_else_statement.htm)  An **if else** statement consists of a Boolean expression followed by one or more statements. If the condition is True, the statements under **If** statements are executed. If the condition is false, the **Else** part of the script is executed. |
| 3 | [**if...elseif..else statement**](https://www.tutorialspoint.com/vba/vba_if_elseif_else_statement.htm)  An **if** statement followed by one or more **ElseIf** statements, that consists of Boolean expressions and then followed by an optional **else statement**, which executes when all the condition become false. |
| 4 | [**nested if statements**](https://www.tutorialspoint.com/vba/vba_nested_if_statements.htm)  An **if** or **elseif** statement inside another **if** or **elseif** statement(s). |
| 5 | [**switch statement**](https://www.tutorialspoint.com/vba/vba_switch_statement.htm)  A **switch** statement allows a variable to be tested for equality against a list of values. |

**VBA - Loops**

There may be a situation when you need to execute a block of code several number of times. In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.

Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or group of statements multiple times. Following is the general form of a loop statement in VBA.



VBA provides the following types of loops to handle looping requirements. Click the following links to check their detail.

|  |  |
| --- | --- |
| **Sr.No.** | **Loop Type & Description** |
| 1 | [**for loop**](https://www.tutorialspoint.com/vba/vba_for_loop.htm)  Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable. |
| 2 | [**for ..each loop**](https://www.tutorialspoint.com/vba/vba_foreach_loop.htm)  This is executed if there is at least one element in the group and reiterated for each element in a group. |
| 3 | [**while..wend loop**](https://www.tutorialspoint.com/vba/vba_while_wend_loop.htm)  This tests the condition before executing the loop body. |
| 4 | [**do..while loops**](https://www.tutorialspoint.com/vba/vba_do_while_loop.htm)  The do..While statements will be executed as long as the condition is True.(i.e.,) The Loop should be repeated till the condition is False. |
| 5 | [**do..until loops**](https://www.tutorialspoint.com/vba/vba_do_until_loop.htm)  The do..Until statements will be executed as long as the condition is False.(i.e.,) The Loop should be repeated till the condition is True. |

## Loop Control Statements

Loop control statements change execution from its normal sequence. When execution leaves a scope, all the remaining statements in the loop are NOT executed.

VBA supports the following control statements. Click the following links to check their detail.

|  |  |
| --- | --- |
| **S.No.** | **Control Statement & Description** |
| 1 | [**Exit For statement**](https://www.tutorialspoint.com/vba/vba_exit_for_statement.htm)  Terminates the **For loop** statement and transfers the execution to the statement immediately following the loop |
| 2 | [**Exit Do statement**](https://www.tutorialspoint.com/vba/vba_exit_do_statement.htm)  Terminates the **Do While** statement and transfers the execution to the statement immediately following the loop |

**Kickstart Your Career**

# VBA – Strings

# Strings are a sequence of characters, which can consist of either alphabets, numbers, special characters, or all of them. A variable is said to be a string if it is enclosed within double quotes " ".

## Examples

str1 = "string" ' Only Alphabets

str2 = "132.45" ' Only Numbers

str3 = "!@#$;\*" ' Only Special Characters

Str4 = "Asc23@#" ' Has all the above

## String Functions

There are predefined VBA String functions, which help the developers to work with the strings very effectively. Following are String methods that are supported in VBA. Please click on each one of the methods to know in detail.

|  |  |
| --- | --- |
| **Sr.No.** | **Function Name & Description** |
| 1 | [**InStr**](https://www.tutorialspoint.com/vba/vba_instr_function.htm)  Returns the first occurrence of the specified substring. Search happens from the left to the right. |
| 2 | [**InstrRev**](https://www.tutorialspoint.com/vba/vba_instrrev_function.htm)  Returns the first occurrence of the specified substring. Search happens from the right to the left. |
| 3 | [**Lcase**](https://www.tutorialspoint.com/vba/vba_lcase_function.htm)  Returns the lower case of the specified string. |
| 4 | [**Ucase**](https://www.tutorialspoint.com/vba/vba_ucase_function.htm)  Returns the upper case of the specified string. |
| 5 | [**Left**](https://www.tutorialspoint.com/vba/vba_left_function.htm)  Returns a specific number of characters from the left side of the string. |
| 6 | [**Right**](https://www.tutorialspoint.com/vba/vba_right_function.htm)  Returns a specific number of characters from the right side of the string. |
| 7 | [**Mid**](https://www.tutorialspoint.com/vba/vba_mid_function.htm)  Returns a specific number of characters from a string based on the specified parameters. |
| 8 | [**Ltrim**](https://www.tutorialspoint.com/vba/vba_ltrim_function.htm)  Returns a string after removing the spaces on the left side of the specified string. |
| 9 | [**Rtrim**](https://www.tutorialspoint.com/vba/vba_rtrim_function.htm)  Returns a string after removing the spaces on the right side of the specified string. |
| 10 | [**Trim**](https://www.tutorialspoint.com/vba/vba_trim_function.htm)  Returns a string value after removing both the leading and the trailing blank spaces. |
| 11 | [**Len**](https://www.tutorialspoint.com/vba/vba_len_function.htm)  Returns the length of the given string. |
| 12 | [**Replace**](https://www.tutorialspoint.com/vba/vba_replace_function.htm)  Returns a string after replacing a string with another string. |
| 13 | [**Space**](https://www.tutorialspoint.com/vba/vba_space_function.htm)  Fills a string with the specified number of spaces. |
| 14 | [**StrComp**](https://www.tutorialspoint.com/vba/vba_strcomp_function.htm)  Returns an integer value after comparing the two specified strings. |
| 15 | [**String**](https://www.tutorialspoint.com/vba/vba_string_function.htm)  Returns a string with a specified character for specified number of times. |
| 16 | [**StrReverse**](https://www.tutorialspoint.com/vba/vba_strreverse_function.htm)  Returns a string after reversing the sequence of the characters of the given string. |

# VBA - Date-Time Function

VBScript Date and Time Functions help the developers to convert date and time from one format to another or to express the date or time value in the format that suits a specific condition.

## Date Functions

|  |  |
| --- | --- |
| **Sr.No.** | **Function & Description** |
| 1 | [**Date**](https://www.tutorialspoint.com/vba/vba_date_function.htm)  A Function, which returns the current system date. |
| 2 | [**CDate**](https://www.tutorialspoint.com/vba/vba_cdate_function.htm)  A Function, which converts a given input to date. |
| 3 | [**DateAdd**](https://www.tutorialspoint.com/vba/vba_dateadd_function.htm)  A Function, which returns a date to which a specified time interval has been added. |
| 4 | [**DateDiff**](https://www.tutorialspoint.com/vba/vba_datediff_function.htm)  A Function, which returns the difference between two time period. |
| 5 | [**DatePart**](https://www.tutorialspoint.com/vba/vba_datepart_function.htm)  A Function, which returns a specified part of the given input date value. |
| 6 | [**DateSerial**](https://www.tutorialspoint.com/vba/vba_dateserial_function.htm)  A Function, which returns a valid date for the given year, month, and date. |
| 7 | [**FormatDateTime**](https://www.tutorialspoint.com/vba/vba_formatdatetime_function.htm)  A Function, which formats the date based on the supplied parameters. |
| 8 | [**IsDate**](https://www.tutorialspoint.com/vba/vba_isdate_function.htm)  A Function, which returns a Boolean Value whether or not the supplied parameter is a date. |
| 9 | [**Day**](https://www.tutorialspoint.com/vba/vba_day_function.htm)  A Function, which returns an integer between 1 and 31 that represents the day of the specified date. |
| 10 | [**Month**](https://www.tutorialspoint.com/vba/vba_month_function.htm)  A Function, which returns an integer between 1 and 12 that represents the month of the specified date. |
| 11 | [**Year**](https://www.tutorialspoint.com/vba/vba_year_function.htm)  A Function, which returns an integer that represents the year of the specified date. |
| 12 | [**MonthName**](https://www.tutorialspoint.com/vba/vba_monthname_function.htm)  A Function, which returns the name of the particular month for the specified date. |
| 13 | [**WeekDay**](https://www.tutorialspoint.com/vba/vba_weekday_function.htm)  A Function, which returns an integer(1 to 7) that represents the day of the week for the specified day. |
| 14 | [**WeekDayName**](https://www.tutorialspoint.com/vba/vba_weekdayname_function.htm)  A Function, which returns the weekday name for the specified day. |

## Time Functions

|  |  |
| --- | --- |
| **Sr.No.** | **Function & Description** |
| 1 | [**Now**](https://www.tutorialspoint.com/vba/vba_now_function.htm)  A Function, which returns the current system date and time. |
| 2 | [**Hour**](https://www.tutorialspoint.com/vba/vba_hour_function.htm)  A Function, which returns an integer between 0 and 23 that represents the hour part of the given time. |
| 3 | [**Minute**](https://www.tutorialspoint.com/vba/vba_minute_function.htm)  A Function, which returns an integer between 0 and 59 that represents the minutes part of the given time. |
| 4 | [**Second**](https://www.tutorialspoint.com/vba/vba_second_function.htm)  A Function, which returns an integer between 0 and 59 that represents the seconds part of the given time. |
| 5 | [**Time**](https://www.tutorialspoint.com/vba/vba_time_function.htm)  A Function, which returns the current system time. |
| 6 | [**Timer**](https://www.tutorialspoint.com/vba/vba_timer_function.htm)  A Function, which returns the number of seconds and milliseconds since 12:00 AM. |
| 7 | [**TimeSerial**](https://www.tutorialspoint.com/vba/vba_timeserial_function.htm)  A Function, which returns the time for the specific input of hour, minute and second. |
| 8 | [**TimeValue**](https://www.tutorialspoint.com/vba/vba_timevalue_function.htm)  A Function, which converts the input string to a time format. |

arrays

We know very well that a variable is a container to store a value. Sometimes, developers are in a position to hold more than one value in a single variable at a time. When a series of values are stored in a single variable, then it is known as an **array variable**.

## Array Declaration

Arrays are declared the same way a variable has been declared except that the declaration of an array variable uses parenthesis. In the following example, the size of the array is mentioned in the brackets.

'Method 1 : Using Dim

Dim arr1() 'Without Size

'Method 2 : Mentioning the Size

Dim arr2(5) 'Declared with size of 5

'Method 3 : using 'Array' Parameter

Dim arr3

arr3 = Array("apple","Orange","Grapes")

* Although, the array size is indicated as 5, it can hold 6 values as array index starts from ZERO.
* Array Index cannot be negative.
* VBScript Arrays can store any type of variable in an array. Hence, an array can store an integer, string, or characters in a single array variable.

## Assigning Values to an Array

The values are assigned to the array by specifying an array index value against each one of the values to be assigned. It can be a string.

### Example

Add a button and add the following function.

Private Sub Constant\_demo\_Click()

Dim arr(5)

arr(0) = "1" 'Number as String

arr(1) = "VBScript" 'String

arr(2) = 100 'Number

arr(3) = 2.45 'Decimal Number

arr(4) = #10/07/2013# 'Date

arr(5) = #12.45 PM# 'Time

msgbox("Value stored in Array index 0 : " & arr(0))

msgbox("Value stored in Array index 1 : " & arr(1))

msgbox("Value stored in Array index 2 : " & arr(2))

msgbox("Value stored in Array index 3 : " & arr(3))

msgbox("Value stored in Array index 4 : " & arr(4))

msgbox("Value stored in Array index 5 : " & arr(5))

End Sub

# VBA - User Defined Functions

A **function** is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code over and over again. This enables the programmers to divide a big program into a number of small and manageable functions.

Apart from inbuilt functions, VBA allows to write user-defined functions as well. In this chapter, you will learn how to write your own functions in VBA.

## Function Definition

A VBA function can have an optional return statement. This is required if you want to return a value from a function.

For example, you can pass two numbers in a function and then you can expect from the function to return their multiplication in your calling program.

**Note** − A function can return multiple values separated by a comma as an array assigned to the function name itself.

Before we use a function, we need to define that particular function. The most common way to define a function in VBA is by using the **Function** keyword, followed by a unique function name and it may or may not carry a list of parameters and a statement with **End Function** keyword, which indicates the end of the function. Following is the basic syntax.

Hide:-

Sub HideWorksheet()

Dim ws As Worksheet

For Each ws In ThisWorkbook.Worksheets

If ws.Name <> ThisWorkbook.ActiveSheet.Name Then

ws.Visible = xlSheetHidden

End If

Next ws

End Sub

Unhide sheets

Sub UnhideAllWorksheet()

Dim ws As Worksheet

For Each ws In ActiveWorkbook.Worksheets

ws.Visible = xlSheetVisible

Next ws

End Sub

Delete sheets

Sub DeleteWorksheets()

Dim ws As Worksheet

For Each ws In ThisWorkbook.Worksheets

If ws.name <> ThisWorkbook.ActiveSheet.name Then

Application.DisplayAlerts = False

ws.Delete

Application.DisplayAlerts = True

End If

Next ws

End Sub

### Protect all Worksheets Instantly

Sub ProtectAllWorskeets()

Dim ws As Worksheet

Dim ps As String

ps = InputBox("Enter a Password.", vbOKCancel)

For Each ws In ActiveWorkbook.Worksheets

ws.Protect Password:=ps

Next ws

End Sub

Resize charts

Sub Resize\_Charts()

Dim i As Integer

For i = 1 To ActiveSheet.ChartObjects.Count

With ActiveSheet.ChartObjects(i)

.Width = 300

.Height = 200

End With

Next i

End Sub.

Insert multiple sheets

Sub InsertMultipleSheets()

Dim i As Integer

i = \_

InputBox("Enter number of sheets to insert.", \_

"Enter Multiple Sheets")

Sheets.Add After:=ActiveSheet, Count:=i

End Sub

Sub lockCellsWithFormulas()

With ActiveSheet

.Unprotect

.Cells.Locked = False

.Cells.SpecialCells(xlCellTypeFormulas).Locked = True

.Protect AllowDeletingRows:=True

End With

End Sub