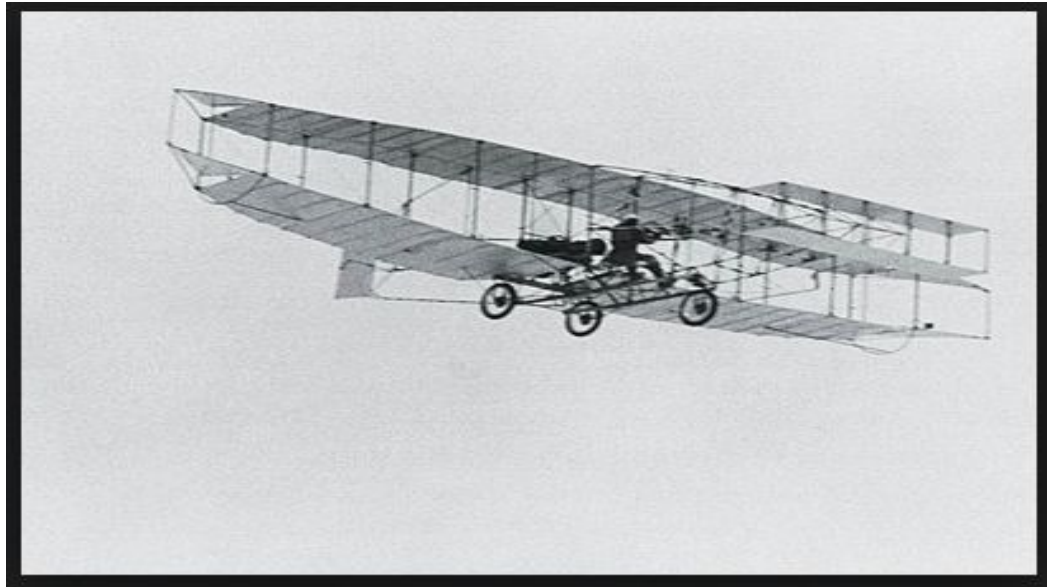


VBScript and UFT



In this playhouse of infinite forms I have had my play, and here have I caught sight of him that is formless.

- Rabindranath Tagore (from *Gitanjali*)

In this chapter we will discuss the VBScript programming and the way to enhance the test scripts to provide more robust automation implementation. We will discuss VBScript, Library and Debugging in this chapter

1.0 Magic of Programming

VBScript is the soul of UFT automation. VBScript was born to automate system administration tasks and designing web applications. Newer technologies replaced VBScript for web-designing but it is still a powerful tool in the hand of Windows system administrator to do system configuration and automation tasks. Internet Explorer one of leader in web browsers arena.

Windows machines used majorly in regression and system testing during 2000s era. QTP, UFT's predecessor, proved very efficient with VBScript to manage Windows as well as Web applications. Now QTP, re-skinned as UFT, provides features to access many technologies and browsers. But this is also a main reason for UFT not to have a cordial relation with UNIX and Linux systems till now. UFT with help of VBScript can do many automation tasks including

1. Managing and scheduling System tasks
2. Window scripting host programming (WSH)
3. Registry Manipulation
4. Managing system configuration, Time and settings
5. Interacting with Browser DOM
6. Interacting with other COM applications APIs, Databases, Filesystem and custom applications
7. Programming of WMI, ADSI, Active Directory, Microsoft Office

A single book will be insufficient to list down on VBScript capabilities. Our goal is to focus on the most efficient VBScript programming ways for automation. Automation accomplish a repetitive task and we will focus on most used VBScript facilities for automation.

2.0 What is Common in All Programing language

Every language provides a way to define variables with different type of datatypes, operators and computation capabilities, decision controls, looping controls, storage and computation mechanism for arrays, strings, memory management, FileSystem management, I/O operations etc. UFT automation engineer needs to understand the VBScript function, capabilities and limitations to accomplish automation task.

2.1 VBScript Programming

Exercise 7.1 First Program in VBScript

Step1. Write a VBScript code to take user input from InputBox.



Open Notepad | Write code as given below | File | Save | HelloUFT.vbs in "C:\Test" | Close

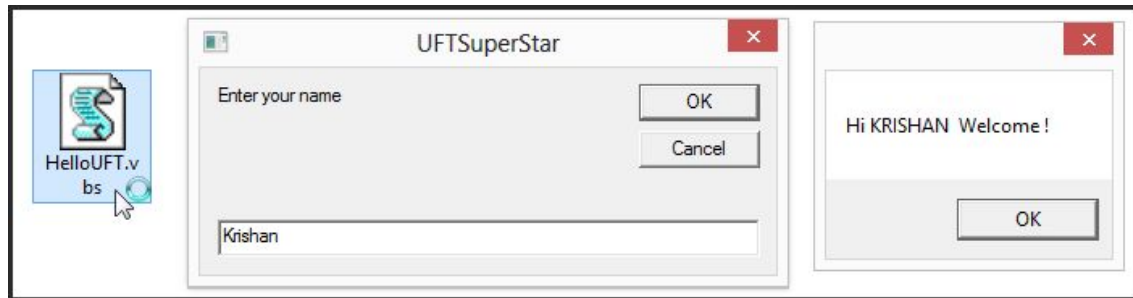
```

HelloUFT.vbs - Notepad
File Edit Format View Help
'To extract a number from alphanumeric text string
Dim strMyName
strMyName=InputBox("Enter your name","UFTSuperStar","Name")
msgbox "Hi " & UCase(strMyName) & " Welcome !" 'String Ucase function

```

Step2. Execute program and provide user input

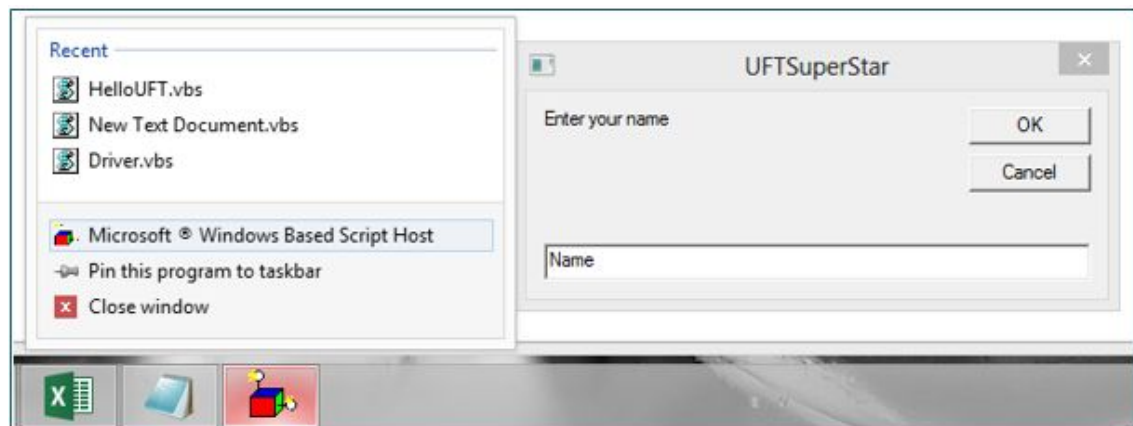
 *Navigate to "C:\Test" | double click on HelloUFT.vbs | Provide "Krishan" in box | OK*



Concept of Exercise 7.1 First Program in VBScript


In STEP1, we created VBS program to take user input and show in message box. In STEP2, we run the program by double clicking on the .vbs file. VBScript run by Microsoft ® Windows Based Script Host (WSH). WSH engine can interpret and run many type of files including .vbs (and many more e.g. plain-text [JScript](#) (.JS and .JSE files) and [VBScript](#) (.VBS and .VBE files)).

Programmer can also write a single program in many scripting language and save/run them in .WSF files. WSH is out of scope of this book but we need to understand simple fact that UFT use VBScript and VBScript programmes run by WSH on windows machine.



***ConfuSense** – You can store vbs program in .vbs files and run them. Also you can write VBS program in UFT. UFT also use the same WSH engine to run the VBScript program.

Internet Explorer understands VBScript. You may find many examples of VBScript code inside html code. HTML file can contain script language e.g. JavaScript and VBScript.

<HTML>		<HTML>
<Body>		<Body>
<script type="text/vbscript">		<script type="text/javascript">
'VBScript code here		'JavaScript code here
</script>		</script>
</Body>		</Body>
</HTML>		</HTML>




***ConfuSense** – Please refer MSDN for complete VBScript language Reference
[http://msdn.microsoft.com/en-us/library/t0aew7h6\(v=vs.84\).aspx](http://msdn.microsoft.com/en-us/library/t0aew7h6(v=vs.84).aspx)

VBScript 1 ▸ VBScript User's Guide - Explain the concept of Visual Basic Scripting Edition (VBScript) usage. 2 ▸ VBScript Language Reference - Information about the elements that comprise the VBScript language.	
<div>MSDN Library</div>	

Understanding VBScript

VBScript is Object Oriented Base language (not Object Oriented language i.e. do not support polymorphism and inheritance) and very similar to procedural language e.g. C-Language. Most of procedural programming language have following features:

 Features	Example
1 Comment	Single quote ' used for comment
2 Variables	Dim a - Variant type , only one datatype
3 operators	Math, comparison and logic (Or,and)
4 Statements and Expressions	any executable valid syntax (a = 5)
5 Conditionals	Loops(For, For-each) etc.
6 Flow Control	Statements e.g. If, If else and If elseif
7 Functions and Subroutines	Function and Sub
8 Array	Variable holds multiple value of same type
9 Strings	Variable holds multiple character string
10 Accessing External Source Files	FilesystemObject
11 Classes	A template of a logical entity
12 System Objects	WMI , Wscript etc.

Let's explore all these features in the following topics.

Escape a Statement – Comment

Single quote will force VBS Engine “not to read” the statement during run-time. Unfortunately, there is no multi-line comment in VBS though UFT provides feature to comment multiple lines by selecting them and insert (un)comment. Comments seems a trivial things to write during time consuming programming efforts. But comments are very vital to make program readable. VBScript is not case sensitive language and you may find yourself in a complete mess if comments are not inserted appropriately.

```
'*****
' File:      Hello.vbs (VBScript finding UFTSuperStar)
' Author:    (c) Krishan Shukla
' Purpose:   Depict User name in message box
' Inputs:    InputBox(): User name string
'            strMyName: the name of the user in InputBox
' Returns:   NA
'Date of Creation - 21/10/2010
'*****
```

Variable

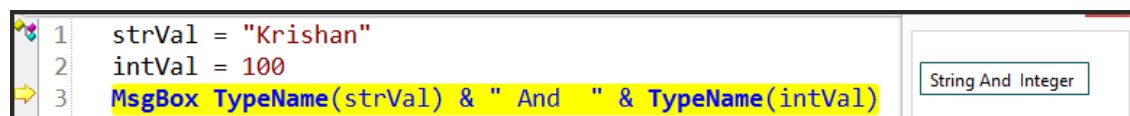
VBScript program is a collection of statements that drive logic. Variable statement can hold the data during the execution e.g. x, y and z in statements in x=5, y=6, z=x+y. So variable represents value in the memory at run-time.

Variable declaration is a step to creating variable in memory i.e. Dim a will create variable “a” in memory. When variable start holding the value it called as initialization i.e. a=5 is initialization.

Dim a	Variable Declaration
a = 5	Variable Initialization

Array declaration Dim <arrayvariablename> (<length>) or Dim arrKrish(10)

Datatype tells the type of data variable contains e.g. integer, string, boolean and currency. VBScript has only one datatype “Variant” but vbscript support sub datatype and process the variable based on the situation. We can find the variable type during run-time by using TypeName



```
1 strVal = "Krishan"
2 intVal = 100
3 MsgBox TypeName(strVal) & " And " & TypeName(intVal)
```

String And Integer

VBS datatype i.e. Variant behaves the type of data it contains. There are following subtype of Variant datatype.

Subtype	Description
Empty	Variant is uninitialized. Value is 0 for numeric variables or a zero-length string ("") for string variables.
Null	Variant intentionally contains no valid data.
Boolean	Contains either True or False .
Byte	Contains integer in the range 0 to 255.
Integer	Contains integer in the range -32,768 to 32,767.
Currency	-922,337,203,685,477.5808 to 922,337,203,685,477.5807.
Long	Contains integer in the range -2,147,483,648 to 2,147,483,647.
Single	Contains a single-precision, floating-point number
Double	Contains a double-precision floating-point number
Date (Time)	Contains a number that represents a date It is between January 1, 100 to December 31, 9999.
String	Contains a variable-length string. It can be up to approximately 2 billion characters in length.
Object	Contains an object.
Error	Contains an error number.

Scope and LifeTime of Variable:

The scope of variable is determined by where you declare it. When you declare variable in function or procedure then variable can be accessed/change in the function or procedure block only. So, effectively, it is bound with local and procedure-level scope. If variable declare outside all the functions then it can be accessed/changed by all the function's code. So, effectively, it is bound with script-level scope.

Lifetime of variable depends on how long it exists. The life span of script-level variable encompass start to end of the script. Variable with procedural level scope die as soon as procedure ends. When the procedure ends the memory become available from procedure level variable. So this is the reason that you can define same variable name in multiple functions.

Operators

Every programming language supports operators. Common operators examples such as "*" for multiplication and "<" for "less than". VBScript have following categories of operators. There is a precedence of operator to dictate which part will evaluate first in the expression e.g.

Operators				
Math	Comparison	Logical	String Concatenation	Assignment
$^$ ($2 \wedge 3 = 8$)	= > <	NOT	&	=
* division(/)	InEquality <>	AND	+	
Integer division (\)	< >	OR		
Mod	>= <=	Xor		
+ -	Is	Eqv		

Mod- return reminder after division. **MsgBox 10 MOD 3 will return "1".**

Is- applicable on objects. It does not compare values but check the reference of two objects that whether two objects are referring same memory location or not.

Logical operators-

If **Not** a= 10 will go in if-condition if a is not 10

If a=10 **AND** b=20 then will go in if-condition if both of a and b returns true

If a=10 **OR** b=20 then will go in if-condition if any of a or b returns true

XOR- It returns false if both of expression remain same otherwise returns true. Truth table of XOR:

A=true xor B= true -> False; A=false xor B= false -> False

A=true xor B= false -> true; A=false xor B= true -> true

Eqv - It returns true if both of expression remain same otherwise returns false. Truth table of EQV:

A=true eqv B= true -> True; A=false eqv B= false -> True

A=true eqv B= false -> False; A=false eqv B= true -> False

String concatenation: & and + used for string concatenation. Operator & converts both expression into string before joining them.

	Variable 1	+	Variable 2	Result		Variable 1	&	Variable 2	Result
Integer	2	+	2	4		2	&	2	22
String	a		B	ab		a	&	b	ab
String/Integer	a	+	2	ERROR		a	&	2	a2

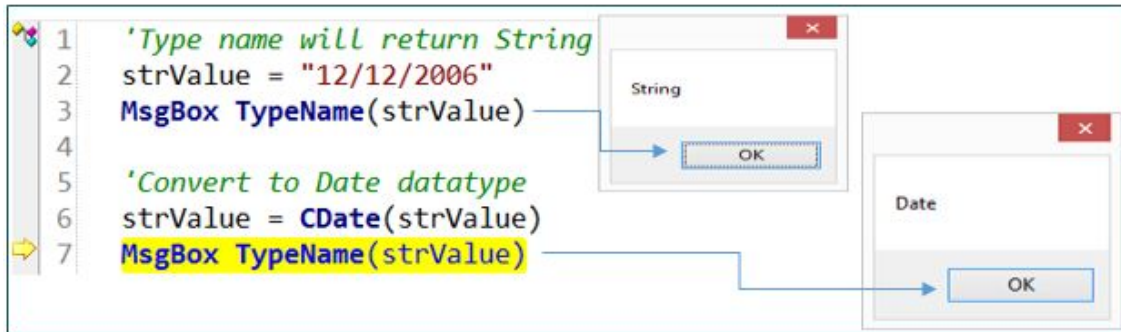
Conversion Function

Conversion functions convert the variable from one datatype to other datatype. Datatype tells the kind of data variable can hold e.g. integer, string, boolean or currency.

VBScript has only one datatype "Variant". Variant, the chameleon, behaves according to the usage

MsgBox 2 & "A" ' 2 and A behave as string, returns 2A
MsgBox 2 + 2 ' 2 behaves as integer, returns 4

This is implicit conversion based on operator. Sometime we need explicit datatype conversion functions to convert one datatype to another before using them in any statement.



The following are the mainly conversion functions available in VBScript

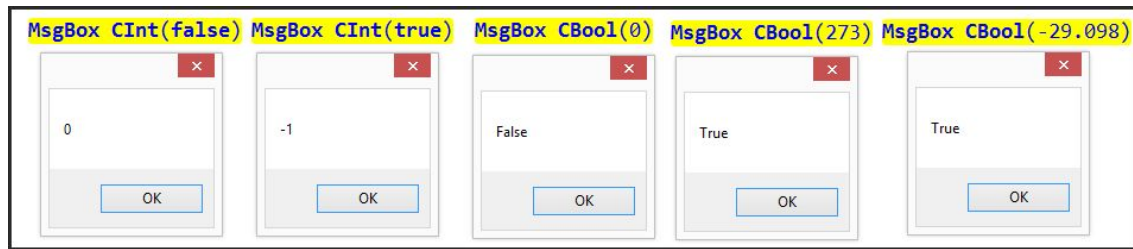
Function	Let's DO	Converts
Asc	Msgbox Asc("A") ' It will display integer 65	First letter in a string to ANSI code
CBool	MsgBox CBool(false) ' Returns zero	An expression to a variant of subtype Boolean
CCur	MsgBox CCur("£10") ' Returns 10 by converting string	An expression to a variant of subtype Currency
CDate	MsgBox CDate("10/OCT/2012") 'Returns 10/10/2012	Any valid date and time expression to the variant of subtype Date
CDBl	MsgBox CDBl("10.23") 'Returns 10.23	An expression to a variant of subtype Double
Chr	MsgBox Chr(65) 'Returns A	Specified ANSI code to a character
CInt	MsgBox CInt("10.23") 'Returns 10	An expression to a variant of subtype Integer
CLng	MsgBox CLng("10.23") 'Returns 10	An expression to a variant of subtype Long
CStr	MsgBox CStr(10.23) 'Returns "10.23"	An expression to a variant of subtype String
Hex	MsgBox Hex(65) 'Returns 41	hexadecimal value of a specified number



***ConfuSense** – Point to remember, string always enclose in "" and integer never enclose with "". If you enclose integer (e.g. "2") it will become string.

Relation of Boolean and Real number conversion

Unlike few other popular language any non-zero value in VBScript consider as true and zero is consider as false.

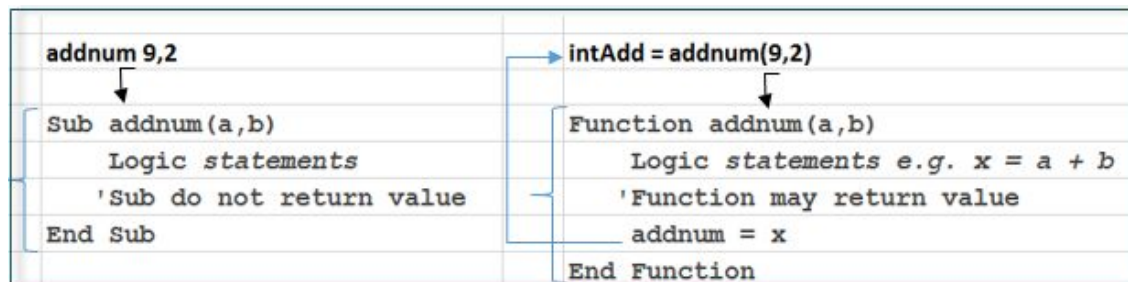


Functions and Sub Procedure

VBScript provides Sub and Function to modularize the code. Sub and Function comprises user logic. Sub cannot return the value but Function may return the value. Return statement is little different in VBS compare to other languages. We need to assign function name with the return value.

FunctionName = Return Value ' *Return Statement in VBScript*

If function returns the value then function must be in parenthesis. This is the reason that sub do not have parenthesis but function statement has parenthesis in figure below.

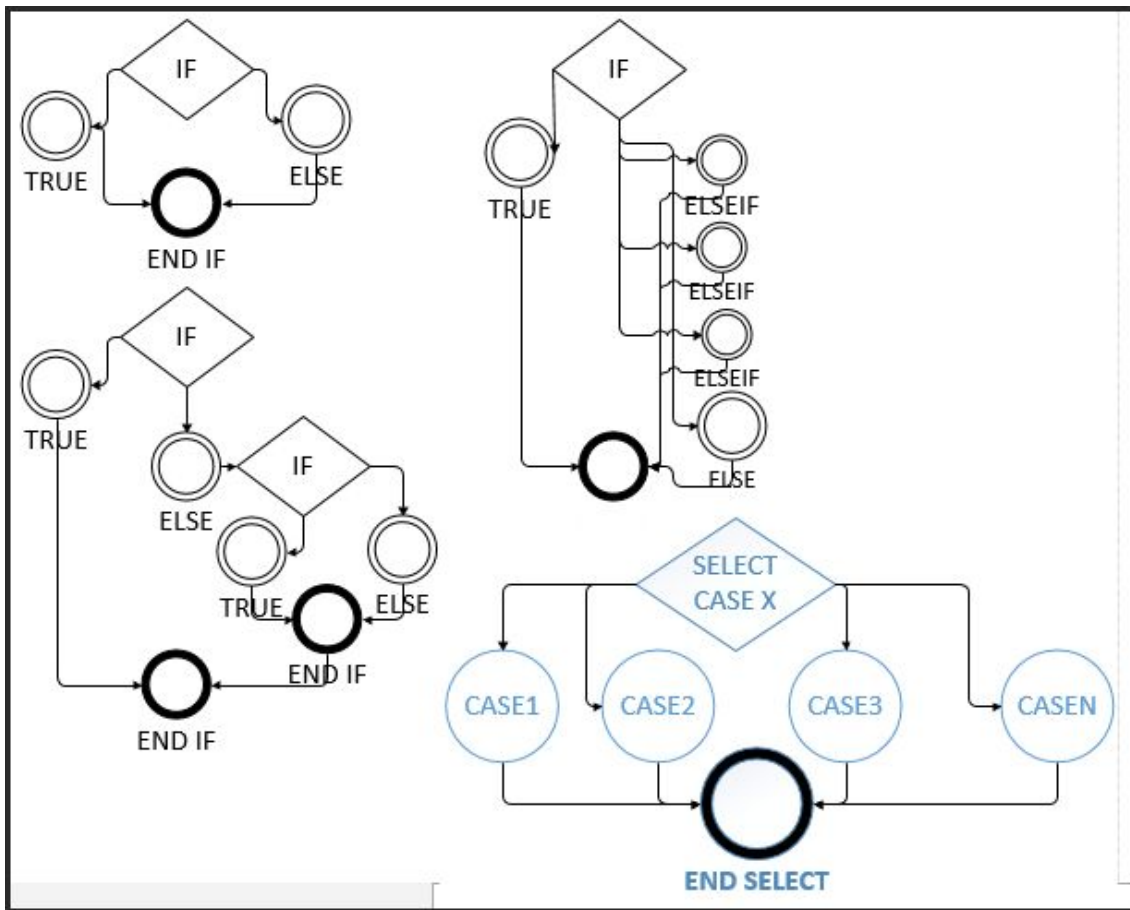


Conditional Statements – Decision and Control statements

Programmer need decision statement to drive the logic. These statement decide the flow of the program. VBScript support following statements

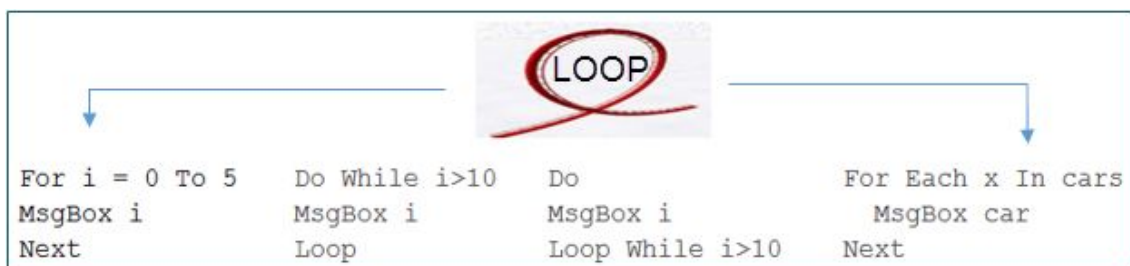
If...Then...Else If...Then...ElseIf...Else Select Case	If x < 5 Then msgbox "x less then 5" ElseIf x =5 Then msgbox "x is 5" ElseIf x =6 Then msgbox "x is 6" Else msgbox "x less then 6" End If	Select Case x Case 5 msgbox "x is equal to 5" Case 6 msgbox "x is equal to 6" Case 7 msgbox "x is equal to 7" Case else msgbox "x != 5,6 or 7" End Select
---	---	--

Decision Statements in VBScript



Loops

VBScript provide the loops to iterate same logic for number of time. There are four types of loops. Except Do loop, all loops first check condition to enter inside the loop for first iteration.



Loops sometime prove very performance intensive particularly performing string based operations. So they must iterate with proper boundaries. Loop need not to be increment by 1. We can use Step (e.g. For i = 0 To 5 Step 2) with loop that will increase loop with specified steps. Infinite loop is common logical error that prevent loop to come out from the block. Loops need to use judiciously as they can hamper performance. Also if any object creation happens in loop then it may lead to memory leak if memory management is not addressed properly. For-each is special loop that works with collection, objects or arrays.

Array and String in VBScript

Array stores fixes size data with same datatype. Array Index starts with zero.

Index	0	1	2	3	4
Value	2	4	6	8	10
arr(index)					

```
arr = Array(2,4,6,8,10)
For i = 0 To 4
    MsgBox arr(i)
Next
```

OR

```
arr = Array(2,4,6,8,10)
For each val in arr
    MsgBox val
Next
```

Array can have multiple dimensions

Declaration	Intialization		
Dim country(2)	country(0) = "USA"	country(1) = "UK"	country(2) = "PAK"
Dim Location(1,2)	Location(0,0)="Jhansi"	Location(0,1)="Bentonville"	Location(0,2)="London"
(row,col)	Location(1,1)="New York"	Location(1,1)="Delhi"	Location(1,2)="Sydney"

There are static and dynamic arrays in VBScript. Array use contiguous memory space in RAM. Static arrays are very efficient and each element is OLE VARIANT type and take 16 byte memory. Discussing about OLE VARIANT is out of scope of this book but in nutshell, it is a complex data structure contains memory pointer to represent run time data location. Dynamic arrays and Dictionaries (also term as Collection) are two ways to add value at runtime in VBScript. Array use Redim statement and dictionary use Add/Remove {key,value} methods to manipulate the data at run time.

Resizing Array

Exercise of Array Resizing

STEP1: Create following program



Open Notepad | Write code as given below in PROGRAM-1 | File | Save | DynamicArray.vbs in "C:\Test" | Close | Run

1	'Declares array without dimensions	'Declares array without dimensions
2	'Interpreter doesn't know size now	'Interpreter doesn't know size now
3	Dim Country()	Dim Country()
4	' Allocate 4 elements	' Allocate 4 elements
5	ReDim country(3)	ReDim country(3)
6	country(0) = "USA"	country(0) = "USA"
7	country(1) = "UK"	country(1) = "UK"
8	country(2) = "PAK"	country(2) = "PAK"
9	country(3) = "AUS"	country(3) = "AUS"
10	For each val in country	For each val in country
11	MsgBox val	MsgBox val
12	Next	Next
13	'Clear old array & create new copy	'Resize old array but keep old values
14	ReDim country(4)	ReDim Preserve country(4)
15		
16	For each val in country	For each val in country
17	MsgBox Val	MsgBox Val
18	MsgBox TypeName(val)	MsgBox TypeName(val)
19	Next	Next
	PROGRAM-1	PROGRAM-2

STEP2: Modify the following program

 Open Notepad | Write code as given in PROGRAM-2 | File | Save | DynamicArrayPreserve.vbs in "C:\Test" | Close | Run

Concept of Exercise of Array Resizing

Size of array can be modified during run time. The **ReDim** statement is used to change the size of an already declared array. Neither **ReDim** can change dimensions of the array nor the data type of an array. **ReDim** releases the existing array from memory and creates a new array. The elements of the new array are initialized to the default value for their data type.

Steps	Statement	Memory Allotment
Unknow length array	Dim Country()	
		1 2 3 4
Redfine array	Redim Country(3)	USA UK PAK AUS
		Copy at new location. Resetting all elements
Redifine Again	Redim Country(4)	Free this Memory
		1 2 3 4 5
		Default all Empty

The **ReDim Preserve** statement is used to change the size of an already declared array but it keeps the values of array. If we make an array smaller than it was originally, we are bound to loose data in in the eliminated elements.

Array filter

```

a=Array("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")
b=Filter(a,"S",True,1) 'b contains
sunday,tuesday,wednesday,thursday,saturday
for each value in b
    MsgBox (value)
next

b=Filter(a,"S",False,1) 'b contains sunday,monday,tuesday,friday
for each value in b
    MsgBox (value)
next

```

Exercise 7.2 VBScript Programming

STEP1: Create a factorial program



Open Notepad | Write code as given below | File | Save | FactorialPro.vbs in "C:\Test" | Close | Double click on FactorialPro.vbs

```

1  intFVal=InputBox("User Input", "First Value", "5")
2  msgbox Factorial(intFVal)      ' Return cathed by MsgBox
3
4  Function Factorial(intNum)
5      factReult=1
6      For i = 1 To intNum
7          factReult = factReult * i
8      Next
9
10     Factorial = factReult      'Return from function
11 End Function

```

STEP2: Create a Fibonacci Series program



Open Notepad | Write code as given below | File | Save | FibonacciPro.vbs in "C:\Test" | Close | Double click on FibonacciPro.vbs

Fibonacci Series																		
0	+	1	+	1	+	2	+	3	+	5	+	8	+	13	+	21	+	...

```

1
2
3
4
5 intFVal=InputBox("User Input", "Fibonacci serise for ", "7")
6 arrFab = Fibonacci(intFVal)
7 For i = 0 To intFVal
8     MsgBox arrFab(i)
9 Next
10
11 Function Fibonacci(intElement)
12     ' Colon : variable declaration and initialization in same line
13     Dim fab: redim fab(intElement)
14     fab(0) = 0
15     fab(1) = 1
16
17     For i = 2 To intElement
18         fab(i) = fab(i-1) + fab(i-2)
19     Next
20     Fibonacci = fab    'Return from function
21
22 End Function

```

VBScript Statements

VBScript contains six types of statements-

- Declaration statements
- Conditional statements for decision-making
- Loop statements for repetition
- Error handling statements
- Classes
- Subroutines

Below table list few of the statements given for example

Statement	Description	Statement	Description
Call	Invokes a subroutine	For/Next	Executes a loop while iterating a variable
Const	Declares a constant value	For Each/Next	Executes a loop while iterating through a collection of objects
Dim	Declares variables	Function/End Function	Declares a routine that will return a value
Do/Loop	Executes a loop until a condition or while a condition is True	If/Then/Else/End If	Conditionally executes one set of statements or another
Erase	Reinitializes the contents of a fixed-size array and frees all of the memory allocated to a variable-sized array	On Error	Takes the specified action if an error condition arises
Select Case/End Select	Chooses a single condition from a list of possible conditions	Option Explicit	Requires that all variables must be declared before their use
Set	Assigns a reference to an object or creates a new object	Private	Declares private variables
Sub	Declares a subroutine	Public	Declares public variables
While/Wend	Executes a loop while a condition is True	Randomize	Initializes the random-number generator
'	Comment	ReDim	Changes the size of an array

VBScript Functions

VBScript provides many in-built functions [REF1.0].

VBScript Functions						
Date/Time	Conversion	Format	Math	String	Validation	Other Function
Date	CBool	FormatCurrency	Abs	Join	VarType	CreateObject
DateAdd	CByte	FormatDateTime	Atn	Filter	TypeName	Eval
DateDiff	CCur	FormatNumber	Cos	InStr	IsNumeric	GetLocale
DatePart	CDate	FormatPercent	Exp	InStrRev	IsArray	GetObject
DateSerial	CDbl		Hex	LCase	IsDate	GetRef
DateValue	CInt		Int	Left/Right	IsEmpty	InputBox
Day	CLng		Fix	Len	IsNull	LoadPicture
Hour	CSng		Log	Ltrim	IsObject	MsgBox
Minute	CStr		Oct	RTrim		RGB
Month			Rnd	Trim		Round
MonthName			Round	Mid		ScriptEngine
Now			Sgn	Replace		ScriptEngineBuildVersion
Second			Sin	Space		ScriptEngineMajorVersion
Time			Sqr	Split		ScriptEngineMinorVersion
TimeSerial			Tan	StrComp		SetLocale
TimeValue				String		
Weekday				StrReverse		
				UCase		

Why so many



String Functions and VBScript

String manipulation is one of the most critical aspect of test automation to drive user logic. VBScript datatype, Variant, mostly behaves as string. String operations and manipulations are the most used statement

In script. UFT use string datatype in nearly everything e.g. validations, handling object properties, decision and comparison statements, exception handling and reporting. It is correct to say that String manipulations are the building blocks of automation.

VBScript provides many string related in-built functions e.g. Ucase("Krishan") will return the "KRISHAN" string.

Exercise 7.2 String Function in VBScript Programming

STEP1: Create a Test



Navigate to UFT | New | Test | GUI Test | "StringTest" | Click Create | Record | OK |
<http://Google.co.uk> | OK | Search with "krishna wikipedia" | Google Search button | Click Image link
 | Close Browser | Save | Stop recording | Save

STEP2: Insert function in script



Navigate to "StringTest" | Modify script as given below | Save | Run

```

1  SystemUtil.CloseProcessByName "iexplore.exe"
2  SystemUtil.Run "iexplore.exe"
3  Browser("Browser").Navigate "http://google.co.uk/"
4  Browser("Browser").Page("Google").WebEdit("q").Set "Krishan Wikipedia"
5  Browser("Browser").Page("Google").WebEdit("q").Submit
6  Browser("Browser").Page("Krishan Wikipedia - Google").WebButton("btnG").Click
7  wait 2
8  strLink = Browser("Browser").Page("Krishan Wikipedia - Google_2"). _
9  Link("Krishna - Wikipedia, the").GetROProperty( "text")
10
11 strLinkU = UcaseFunction(strLink) ' or can write MsgBox UcaseFunction(strLink)
12 MsgBox strLinkU
13
14 Browser("Browser").Page("Krishan Wikipedia - Google_2"). _
15 Link("Krishna - Wikipedia, the").Click
16 Browser("Browser").Page("Krishna - Wikipedia, the").Sync
17 Browser("Browser").CloseAllTabs
18
19 Function UcaseFunction(linkneedInUpper)
20     'return UCase(linkneedInUpper) X
21     UcaseFunction = UCase(linkneedInUpper) ✓
22 End Function
  
```

STEP3: Create Other String functions



Navigate to "StringTest" | Insert steps before the step clicking link | Save | Run

```

14  MsgBox InStrFunction(strLink, "free")
15  Function InStrFunction(strMain, strSearch)
16
17      isStringfound = Instr(strMain, strSearch)
18      InStrFunction = isStringfound
19  End Function
  
```

Zero; If strSearch is not in strMain

Position; If strSearch is in strMain

Concept of Exercise 7.2 String Function in VBScript Programming

Following are the string functions:

Example of Important String functions:

Images	Function Name		Result/Return
Pic a man with inchytape	Len	intLenghtOfString = Len ("UFTStar")	7
Pic a girl to compare two dimond	StrComp	StrComp("star","star")	0
		StrComp("star","STar")	Non- Negative
	InStr	Instr("InStr star return index of found string" , "star")	6
	InStr	Instr("InStr star return 0 for not-founded string" , "Star")	0
	InStr	Instr("UFT star in this" , "Star",vbTextCompare)	4
	InStr	Instr(1, "this is star string" , "star")	8
	InStr	Instr(9, "this is star string" , "star")	0
a car with big and small tyre	LCase	Lcase("sTar")	Star
	Ucase	Ucase("sTar")	STAR
A pic of keep right	Right	Right("Right return digits from right",7)	m right
	Left	Right("Right return digits from right",7)	Right r
Pick a man from in between	Mid	Mid("Dnt confuse in MID, return 1 num to 2 num",6,2)	On
		Mid("Dnt confuse in MID, return 1 num to 2 num",36)	2 num
http://babajidesalu.wordpress.com/2010/01/26/nigeria-a-problem-with-choice/			
UIta Khada man	StrRevers e	StrReverse("TFU")	UFT
Tukde aadmi kaat raha	Split	arrName= Split("John,Ram,Javed,Aaron,,Gautam")	arrName(0)="jo hn" arrName(1)="ra m" arrName(2)="Ja

			ved" arrName(3)="A aron" arrName(4)="" arrName(5)="G autam"
--	--	--	--

String Constants

Constant never changes its value during run-time. There are two type of constant in VBScript.

Constant Type		Example
User-Defined	1	<code>Const Site = "QuickTestPro.co.uk"</code>
	2	<code>Const Site = #3-4-99# 'Const Date use # not "</code>
Intrinsic (in-built)	1	<code>vbWednesday</code> returns 4, <code>msgbox vbWednesday</code>

Date and time are enclosing them in number signs (#) with string use quotation marks (" ").

Since these constants are built into VBScript, you don't have to define them before using them. Use them anywhere in your code to represent the values shown for each.

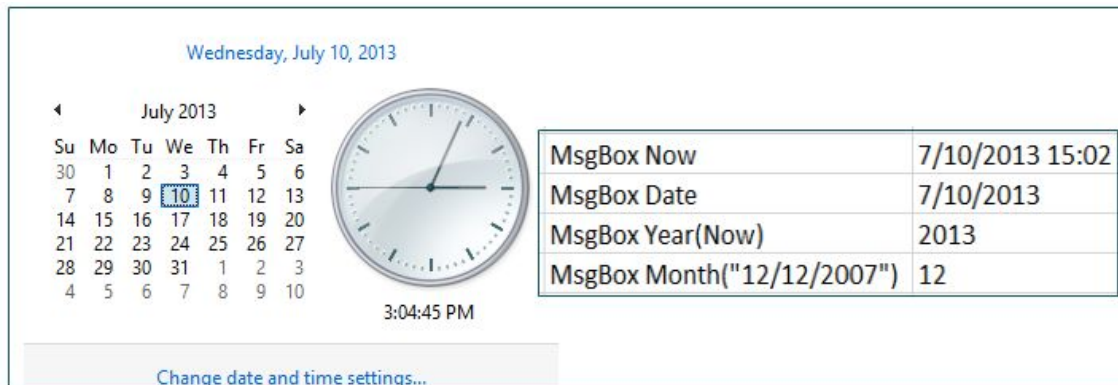
Constant	Return Value	Detail
<code>vbCr</code>	<code>Chr(13)</code>	Carriage return
<code>vbLf</code>	<code>Chr(10)</code>	Line feed
<code>vbYellow</code>	<code>&hFFFF</code>	Yellow
<code>vbBinaryCompare</code>	0	Perform a binary comparison.
<code>vbWednesday</code>	4	Wednesday
<code>vbObjectError</code>	-2147221504	User-defined error numbers should be greater than this value

Following are the internal constant type in VBScript [Ref-1]. CONST are very useful in framework designing to assert the non-changing state of any variable and should be used appropriately.

VBScript Constants	
Color Constants	Date Format Constants
Comparison Constants	Miscellaneous Constants
Date and Time Constants	MsgBox Constants
VarType Constants	Tristate Constants

Date and Time Functions

There are many VBScript date & time functions available e.g. function "Now" returns the system time

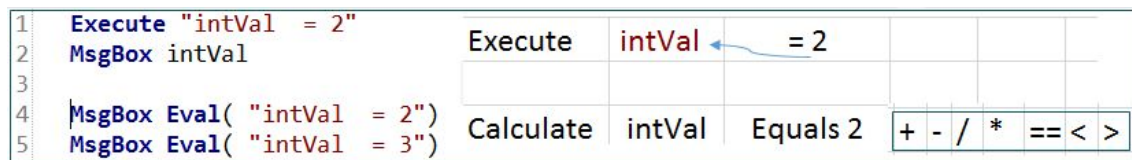


The screenshot shows the Windows system clock and date settings interface. At the top, it displays "Wednesday, July 10, 2013". Below this is a calendar for July 2013, with the 10th highlighted. To the right of the calendar is a clock showing the time as 3:04:45 PM. Below the clock is a button labeled "Change date and time settings...". To the right of the clock is a table showing the results of several VBScript date and time functions:

MsgBox Now	7/10/2013 15:02
MsgBox Date	7/10/2013
MsgBox Year(Now)	2013
MsgBox Month("12/12/2007")	12

Execute and Eval Statement

Execute and Eval are two very useful statement. You can think in this way, Eval calculate the expression and return result but Execute executes the statement.



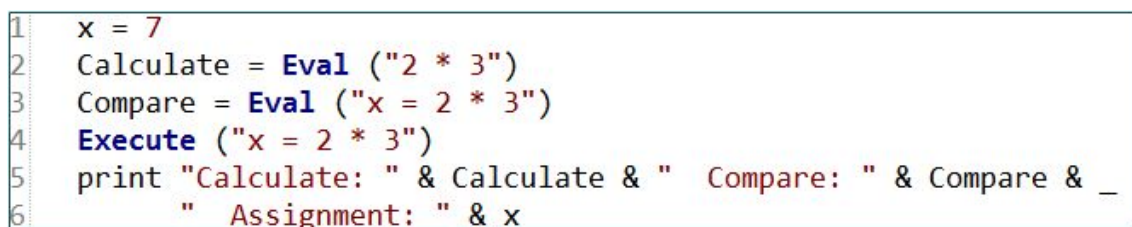
The screenshot shows a VBScript editor with the following code:

```
1 Execute "intVal = 2"
2 MsgBox intVal
3
4 MsgBox Eval( "intVal = 2")
5 MsgBox Eval( "intVal = 3")
```

To the right of the code is a table showing the results of the Execute and Eval statements:

Execute	intVal	= 2
Calculate	intVal	Equals 2

You can also test the following to see the behaviour of execute and eval.



The screenshot shows a VBScript editor with the following code:

```
1 x = 7
2 Calculate = Eval ("2 * 3")
3 Compare = Eval ("x = 2 * 3")
4 Execute ("x = 2 * 3")
5 print "Calculate: " & Calculate & " Compare: " & Compare & _
6 " Assignment: " & x
```

Exercise 7.3 Execute and Eval in Test Script

STEP1: Create Test



Navigate to UFT | New | Test | GUI Test | "In_LineCodeTest" | Click Create | Record | Select First radio button | OK | <http://Google.co.uk> | OK | Search with "krishna wikipedia" | Google Search button | Click Image link | Close Browser | Save | Stop recording | Save

STEP2: Modify script with following line

```

5 Browser("Browser").Page("Google").WebEdit("q").Set "Krishan Wikipedia"
6
7
8 'objGoogleEdit is String here
9 objGoogleEdit = "Browser(""Browser"").Page(""Google"").WebEdit(""q"")"
10
11 'objGoogleEdit converted to Object
12 Execute("Set objGoogleEdit = " & objGoogleEdit )
13 objGoogleEdit.Set "Krishan Wikipedia"

```

↕ Can Be Converted INTO ↕

Variable – Scope and call by

Call by Ref and Call by Val in VBScript

```

fval =10
sval =20
Calling fval , sval
MsgBox fval & " " & sval

Function Calling(ByVal fV, ByVal sV)
fV =20
sV = 10
MsgBox fV & " " & sV
End Function

```

By Default it is - Call byRef

fval	10
sval	20
fV	10
sV	20

Copy of value
passed in s in
Call "ByVal"

fval	10
sval	20
fV	
sV	

Memory Location
passed in Call
"ByRef"

```
fval =10  
sval =20  
Calling fval , sval  
MsgBox fval & " " & sval
```



```
Function Calling(ByRef fV, ByRef sV)  
fV =20  
sV = 10  
MsgBox fV & " " & sV  
End Function
```

```
fval =10  
sval =20  
Calling fval , sval  
MsgBox fval & " " & sval
```



```
Function Calling(fV, sV)  
fV =20  
sV = 10  
MsgBox fV & " " & sV  
End Function
```

ByRef is default. So if we don't specify anything then it will not pass by Value

```
fval =10  
sval =20  
Calling fval , sval  
MsgBox fval & " " & sval
```

= IS SAME AS =

```
Function Calling(ByRef fV, ByRef sV)  
fV =20  
sV = 10  
MsgBox fV & " " & sV  
End Function
```

```
fval =10  
sval =20  
Calling fval , sval  
MsgBox fval & " " & sval
```

```
Function Calling(fV, sV)  
fV =20  
sV = 10  
MsgBox fV & " " & sV  
End Function
```

Classes in VBScript

VBScript is Object Oriented Base language. Class is a logical representation of practical concept. Think about the concept of car. The things come in mind are behaviour for example start, stop, calculate miles etc. or state for example color, height , weight etc. Car can be considered as a class to represent the concept here

Exercise Understanding Class in VBScript

STEP1: Create a Class with a Function



[Navigate to UFT | File | New | "MyClassTest" | Create | write following code | Run](#)

```

1  Class Arithmetic
2
3  Function add(intFirstNum, intSecondNum)
4      add = intFirstNum + intSecondNum 'Return Val
5  End Function
6
7  End Class
8
9  Set objArithmetic = New Arithmetic ' Object Createion
10 intAddedVal = objArithmetic.add (10,20)
11 MsgBox intAddedVal

```

STEP2: Write more functions under class



Navigate "MyClassTest" in UFT | write following code inside class statement | Run

Function subtract(intFirstNum, intSecondNum)
add = intFirstNum - intSecondNum
End Function
Function multiply(intFirstNum, intSecondNum)
add = intFirstNum * intSecondNum
End Function

Concept of Exercise Understanding Class in VBScript

In STEP1, we created a class. Class is a concept and consider as a template. But program control cannot go inside the class block. We need to create object to use the class. Object is the physical realization of class or instance of a class. "Set" keyword use to define the the object and "New" keyword initialize the object in VBScript (Line-9). Object use dot operator "." to access functions and property of class (Line-10).

In Step2, we created other many function and modularize them by keeping inside class. Now a single object can call any function (or known as behaviour) of the class.

More on VBScript Class

Exercise Encapsulation and Access Specifiers in Class

STEP1: Create a Class with a Property



Navigate to UFT | File | New | "ClasswithPropertyTest" | Create | write following code | Run


```

1  Class User
2      Private m_userName 'Cannot directly access it
3
4      Public Property Get UserName 'Get returns variable
5          UserName = m_userName
6      End Property
7      'Let set the variable
8      Public Property Let UserName (strUserName)
9          m_userName = strUserName
10     End Property
11 End Class
12 Set objUser = New User
13 MsgBox objUser.UserName
14 objUser.UserName = "Krishan"
15 MsgBox objUser.UserName

```

Understanding Exercise Encapsulation and Access Specifiers in Class

Exercise Object Initialization by Class_Initialize

STEP1: Create a Class with a Property



Navigate to UFT | File | New | "ObjectInitializationTest" | Create | write following code | Run

Automate Other Applications from VBScript by CreateObject

Create Object method is provided so that other applications can be automated from VBScript.

STEP1: Create an Internet Explorer object and program the logic



Navigate to UFT | File | New | "CreateObjectTest" | Create | write following code | Run

```

1  'Create IE Object Instance
2  Set objBrowser = CreateObject("InternetExplorer.Application")
3  objBrowser.Visible = True
4  objBrowser.Navigate "QuickTestPro.co.uk"
5  wait 3
6  objBrowser.Quit
7  Set objBrowser = Nothing

```





Supported Method/Property

Very Important - Destroy Object to release memory

In Step1, we have created Internet explorer object, set properties e.g. visible="ture" and call methods. Creation of object requires memory, so it is programmer's responsibility to release the memory by setting object to "Nothing". If we will not release the object then it may lead to memory leak. Explicit releasing the memory only applies on the object created by CreateObject. VBS engine implicit take care memory management of the system objects e.g. err object, regular expression object etc.

Specification of CreateObject

CreateObject takes three parameters and return object with initial null value.

1. Class/Type name of Application i.e. Application or sometime referred as server. So you should not confuse with the parameter server.
2. Class or type of object is second parameter. So one application can expose multiple classes that can render different type of objects. In the above example, server would be "InternetExplorer" while class name would be "Application". So we will get the Object of server. Class i.e. "InternetExplorer. Application".
3. Location is the machine name where we want to create the object. By default it is local machine i.e. localhost. We can specify the IP or machine name to create object on remote machine. To use this argument, the internet security must be turned off.

CreateObject(Application.ClassName[,location])		
	Parameter	Description
1	Application	Required. The name of the application that provides the object
2	ClassName	Required. The type/class of the object
3	location	Optional. Where to create the object. Default Localhost i.e. local machine

Practical use of CreateObject

Well, we just explored the IE automation by vbscript using CreateObject. Testing need to interact various application e.g. word,excel,text files, databases, browsers and so on. Automation of these application is possible by the same way as we did for Internet explorer. So, what will be the "application.classname" for these applications?

Following are the few example of applications that can be automate. The supported methods and property can be found in the hel/reference file of these applications. This list is not exhaustive and there are many more applications can be automate by creating object of the application

OLE Automation To Automate Applications	
Application Name	Application.Classname
Unified Functional Testing	QuickTest.Application
MS Excel	Excel.Application
QuickTestPro	QuickTest.Application
MS Word Document	Word.Application
File System	Scripting.FileSystemObject
MS Outlook	Outlook.Application
NewMail	CDONTS.Newmail
Windows Script	WScript.Shell
ADODB for Database etc.	ADODB.Connection
DAO for Database etc.	DAO.DBEngine.35
Mercury - for Copy & paste	Mercury.Clipboard
Quality Center	TDApiOle.TDConnection
Mercury - for Keyboard and Mouse	Mercury.DeviceReplay
Vbscript Dictionary	Scripting.Dictionary
XML DOM	Microsoft.XMLDOM
Windows Network	WScript.Network
Adobe Acrobat	AcroExch.App

Decrypt the all buzzwords - OLE, COM and Automation

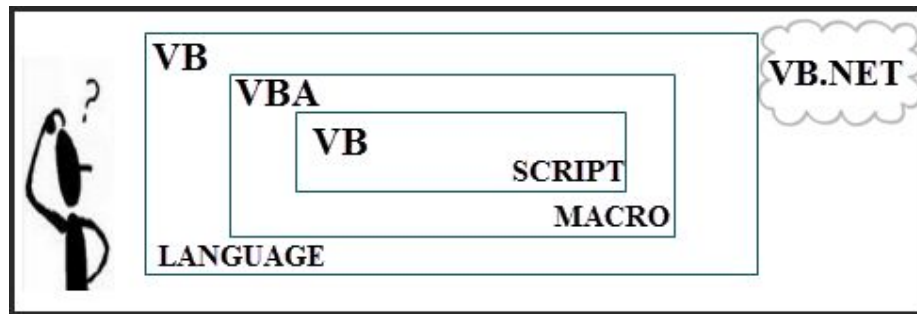
Confusion of V-Matrix

Visual Basic is 3rd generation language developed by Microsoft in 1991. Visual Basic was designed to be relatively easy to learn and use. VB provide complete development environment comparing to languages like C++, Visual C++ etc.

Visual Basic .NET (VB.NET) is an object-oriented computer programming language that can be viewed as the successor of the classic Visual Basic (VB), implemented on the .NET Framework. Microsoft's extended support ended in March 2008 for Visual Basic and now Visual Basic referred simply to Visual Basic .NET.

A subset of VB, Visual Basic for Applications (VBA), is used as a macro or scripting language. VBS use to automate Microsoft applications including but not limited to Microsoft office e.g. MS Excel, MS Word and MS Access etc.

VBScript is a "light-weight" subset of Visual Basic for Applications. Many features (but NOT all) of VBA supported by VBScript. Most of the time you may confuse by encountering the VBA code written in VBA that looks very similar to VBScript.



This is an example that show how we may convert from VBA to VBS.

VBScript	VB and VBA
Dim i	Dim i As Integer
i = 12	i = 12

	VBSCRIPT	VBA
Variables	Dim A	Dim A as Integer
System Objects	Err Object	Screen, Printer, App, Debug, Err, and Clipboard objects
File Operations	Supports FileSystemObject	Supports I/O commands Open, Read, Line Input etc.
Debugging	VBScript does not provide any debugging features. There are no breakpoints, watches, or debug windows. Do not confuse with UFT. UFT enable program to debug VBScript program	Support extensive debugging.
Jump Statements	Not Supported	Supports Goto, GoSub, line numbers and line labels
Like Operator	Instr can search substring in string e.g Instr(1, "Uma", "Um")	Support Like statement e.g. MsgBox "rmaLaI" Like "SharmaLaI"
Financial functions.	VBScript does not support financial functions.	Supported
Performance	Comparably slow	Comparably fast
embedded	Web Pages for Internet Explorer	MS Applications e.g. MS Office (MS Excel etc.)

VBScript Coding Conventions

VBS Coding convention [REF] is important but most overlooked phenomena. Programmer must follow the common standard and convention that lead clarity in code and minimize maintenance efforts.

Coding convention can include the standardization of the following:

1. Naming convention for Variable, Objects and functions(or known as procedures)
2. Comment conventions

3. Indent standard

Variable	Prefix	Example		Object type	Prefix	Example
Boolean	bln	blnExist	1	Check box	chk	chkStatus
Byte	byt	bytData	2	Combo box, drop-down	cbo	cboGurha
Date (Time)	dtm	dtmToday	3	Command button	cmd	cmdExit
Double	dbl	dblWeight	4	Common dialog	dlg	dlgFileOpen
Error	err	errIssue	5	Frame	fra	fraHome
Integer	int	intAge	6	Image	img	imgFavicon
Long	lng	lngDistance	7	Label	lbl	lblHelpMessage
Object	obj	objExcel	8	Line	lin	linBreak
Single	sng	sngDigit	9	List Box	lst	lstUFTVersion
String	str	strKrishID	10	Text box	txt	txtFirstName

All function should begin with appropriate comment section. Comment need not to describe the implementation details but provide the purpose, author, creation/updating date etc. Whenever possible, parameters passed in function and return values should be described properly.

Indent make program understandable. Nested blocks should be indented with four spaces while comment should start after one space.

Fact-Sheet of VBScript

Fast and Fun Fact of VBScript - "The END" of Confusion	
Dim a Vs. dlm A	VBScript is case Insensitive language e.g. "dim A" and "DiM a" are same
Option Explicit	Writing "Option Explicit" force variable to declare with optional Dim statement
Kill VBS	VBScript execution can be killed in task manager by Wscript or Window Script host process
VB/VBS/VBS	VBS looks like VB and VBA but they are different in Syntax
OOB	VBS is Object Oriented Base language and does not support Inheritance
Class	We can write classes in VBScript
Object	Set oObject = new ClassName will create a oObject of class in VBScript
Err Objects	Err is internal VBScript objects Err stores information of any error in VBScript execution.
Single Thread	VBS do not have multi-threading
VBS Born For	VBS born to build dynamic pages and Windows System administration tasks.
Interpreter	VBS has interpreter instead of compiler
WMI/Wscript	VBS can work with windows specific WMI, Wscript etc to perform system related task
Wscript in UFT	We cannot create object of Wscript in UFT but call a library that can create and use that.

Summary:

References:

[REF1.0] VBS Functions [http://msdn.microsoft.com/en-us/library/3ca8tfek\(v=vs.84\).aspx](http://msdn.microsoft.com/en-us/library/3ca8tfek(v=vs.84).aspx)

[REF2.0] Constants (VBScript) [http://msdn.microsoft.com/en-us/library/ydz4cfk3\(v=vs.84\).aspx](http://msdn.microsoft.com/en-us/library/ydz4cfk3(v=vs.84).aspx)

[REF] VBS Coding Conventions [http://msdn.microsoft.com/en-us/library/ektke1b0\(v=vs.71\).aspx](http://msdn.microsoft.com/en-us/library/ektke1b0(v=vs.71).aspx)